University of La Laguna
Department of Computer Science and Systems Engineering

PROBE ARM MOTION TECHNIQUES FOR MIRADAS MULTI-OBJECT SPECTROGRAPH

JOSEP SABATER MORROS

Supervisors
Dr. Santiago Torres Álvarez
Prof. Francisco Garzón López

Tutor
Dr. Santiago Torres Álvarez

A thesis submitted for the degree of Doctor of Philosophy in Industrial, Computer Science and Environmental Engineering.

June 2019

Eixordats pel brogit sorollós de les reixes
no escoltem l'esperança de levíssim trepig.
Car tothom sent la pedra quan esquinça els verds pàmpols
però qui pot veure obrir-se la llavor del forment
o mesurar quant creix l'infant de l'alba a l'alba?
Talment és una flor de desclosa suau
l'esperança: segur, tenaç com el blat tendre
vindrà el seu just esplet quan la neu dansarà
lleugera desnuant-se per les arbrades clares.

Maria Àngels Anglada: Primavera Lliure

El tiempo transcurre con frenesí, unos acontecimientos suceden a otros en caótico torbellino. ¡Qué cosa tan extraña la naturaleza humana! Antes se te cortaba la respiración ante semejante celeridad y, sin embargo, ahora es como si comenzases a acostumbrarte y cada vez resultase más complicado alcanzar la perplejidad y que el horror te domine.

Veniamín A. Kaverin: Ante el Espejo

There is no future to fear, no past to regret, only this, only a series of moments strung along, like lit globes on a string.

Anna Hope: Expectation

Abstract

Since ancient times, humans have been captivated by the bright objects in the sky. Despite being around for ages, the astronomers still have to unravel many mysteries about the Milky Way. How was it formed, or has it changed through time? These are only two of the multiple unanswered questions in Astrophysics. Scientists have been constructing different models of the evolution of our galaxy. Still, more observations are needed to refine the constraints present in all of them. By way of illustration, quality data disentangling the chemical composition of the primary components of our galaxy, the stars, would be of great help. In the last decade, many efforts have been focused on the development of new multiple-object spectroscopy facilities that will provide invaluable information to update properly those models.

The Mid-resolution InfRAreD Astronomical Spectrograph (MIRADAS) is a near-infrared multi-object echelle spectrograph for the 10.4-meter Gran Telescopio Canarias (GTC) being built by an international consortium. Due to its resolution and its multiplexing capabilities, MIRADAS will help astronomers to address some of the leading scientific challenges of the following decades. With its 12 deployable integral field units based on cryogenic probe arms with pick-off mirrors, this astronomical instrument will allow scientists to observe concurrently up to 12 user-defined celestial targets. The structure of each of these opto-mechanical devices is conceived for optical simplicity as well as for a higher degree of stability when working upside down. However, the cost of this design is the highly complex behavior of these arms.

In this thesis, we provide a comprehensive study of the probe arm employed in MIRADAS. A mathematical model, as well as solutions of the forward and inverse kinematics problems, is presented. Furthermore, relevant particularities constraining the motion of the arm are also analyzed. First, we study the configuration space and the regions that are physically unachievable because of the closed-loop nature of the arm. Second, we pay special attention to the inability to interpolate of the controllers commanding the arm actuators. In fact, this imposes restrictions to the motion algorithms presented in this manuscript. Finally, we also discuss different arm patrolling strategies, putting emphasis on their advantages and weak points.

On the other hand, the room in the circular plate where the arms are attached is scarce. Therefore, to acquire the required celestial targets, all arms in the system need to be carefully moved from their current locations to their destinations. This task in MIRADAS has been divided into three different processes, all thoroughly discussed in this work. The first of them, field

segmentation, adequately arranges the distinct targets in scatter fields to be observed by the instrument. In particular, it computes several clusters, whose elements are posteriorly employed in target allocation, the second processing step. Moreover, we also determine a geometric center for these groups of targets, information required for the telescope's pointing procedure. Regarding target allocation, which is in charge of determining the most suitable assignment of arms to targets, we present two different solutions: one based in linear programming and other in a greedy-like metaheuristic. The former, as demonstrated by the tests, obtain better assignment plans in terms of the different metrics utilized. However, it becomes impractical for large fields. In these scenarios, the metaheuristic returns good results in a reduced amount of execution time. Finally, the assignment plans delivered by the allocator are then passed to the third and last step of the processing process. The motion planner individually analyzes each assignment plan, yielding coordinated collision-free high-level motions for all arms involved in the allocation. These movements, once accurately translated to low-level commands and executed by a piece of software controlling the real hardware, will position the pick-off mirror of each arm in the corresponding sky locations. Experimental tests show that the motion planner delivers successful trajectories for a typical scenario presenting at least one instance of every known conflict in MIRADAS. The proposed solution also returns collision-free motions for a series of cases involving real science targets.

Acknowledgements

Almost every thesis begins with acknowledgments to the many individuals who offered valuable help, support, suggestions and constructive criticism. I confess that I never fully appreciated how true this was until doing my own research. The list of people to whom I am in debt is long.

First and foremost, I would like to thank my supervisors, Dr. Santiago Álvarez Torres and Prof. Francisco Garzón López, for their advice, guidance, support and encouragement throughout a rather prolonged process. I am also especially indebted to them for being so kind, attentive, and for making me feel at home in my multiple short stays at San Cristóbal de la Laguna.

I wish to express my deepest gratitude to Dr. José María Gómez Cama and Prof. Jordi Torra Roca for giving me the once in a lifetime opportunity to be involved in a project combining such interesting fields as robotics and astronomy. Unfortunately, Jordi has left us too soon. However, every time I gaze up into the night sky and see the stars in all their glory, I know that Jordi is there encouraging us with his engaging and perpetual smile. I would also like to extend a special thanks to Prof. Stephen Eikenberry for the confidence he has shown in me over the years.

Many people helped me to understand some of the concepts employed in this thesis. Dr. Jorge Riera-Ledesma has greatly informed and influenced my thinking in linear programming. Jorge enlightened me when I felt lost and made it possible to experience the joy of collaboration. Dr. Rafael Arnay del Arco, whose fruitful comments and conversations provided invaluable insights into several topics. And, finally, I am very grateful to Dr. Alexandre Perera-Lluna for being always available and kind enough to give useful answers to my multiple questions.

My thanks to those that have been my officemates in the last few years. While it has been a pleasure to work with all of them, I really appreciate the fun I have had with Albert Casas, Pablo Oriol and David Roma. In addition, I am very thankful to the staff at the Instituto de Astrofísica de Canarias and at Universidad de La Laguna that had with me all those enjoyable dinners. I am in debt with them for showing me such nice hidden restaurants. Likewise, I was very fortunate to share very good moments with many at the University of Florida, especially with Verónica Donoso, Claudia Vega, Dr. Craig Warner and Dr. Alan Garner.

Finally, I thank my wife, Anna, and my son, Sasha, for their unconditional love and patience while I was immersed in my research as well as in writing this manuscript. My gratitude also goes to my parents, who devoted all their energies to provide me with the education necessary to face this life-changing challenge, and to my brother for always standing by my side.

CONTENTS

List of Tables

List of Figures

Figure **Page**

CHAPTER 1

INTRODUCTION

The Sun, the Moon, and the thousands of lights in the sky aroused people's interest a long time ago. Although observations of celestial bodies took place in many pre-historic societies, it is still unclear when and where people exactly started to be aware of the heavens. The archaeological earliest evidence dates back to the Aurignacian Culture of Europe, c.32 000 B.C. Alexander Marshack thoroughly studied several ancient bone fragments discovered in French caves. He concluded that some of the carvings found in the form of crescent depicted the phases of the moon (Marshack, 1964). These first lunar calendars, indeed, predated writing, which was not invented until almost 25 000 years later.

Astronomy has changed man's idea of the world from anthropocentric conceptions to the modern view of a vast universe where we, humans, play an insignificant role. As any fundamental science, the ultimate goal of this field of knowledge is to validate or evolve through direct observation the theoretical models constructed by logical reasoning. Up to the end of the Middle Ages, the most important means of observation of the sky was the human eye. The first mention to a telescope is usually credited to the Dutch spectacle-maker Hans Lippershey. In 1608, he submitted to the government in the Netherlands a patent related to an instrument "for seeing things far away as if they were nearby" (Watson, 2007). Although Lippershey failed to receive the patent, the word about such a device rapidly spread all over Europe. A year later, Galileo Galilei, then Professor of Mathematics at the University of Padua (Italy), enhanced the invention and made his first systematic observations of sky objects (Drake, 2003). His startling discoveries were described in the treatise *Sidereus Nuncius*[1], published in March 1610.

Before the 19th century, astronomical research was primarily focused on measuring the positions and computing the motions of celestial objects. However, then a breathtaking step on

[1] The Sidereal Messenger

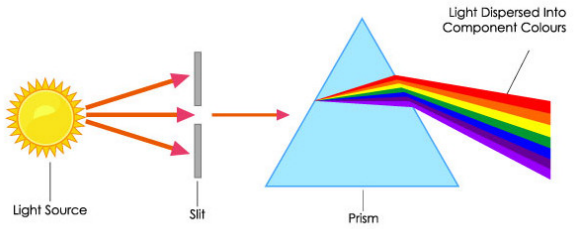


Figure 1.1: The prism, a triangular block of glass with smooth polished surfaces, disperses the incident light beam into its constituent wavelengths.

the way to modern astrophysics was done. Astronomers, thanks to the adoption of new techniques such as spectroscopy and photography, began to apply the laws of chemistry and physics to ascertain the nature of cosmos. Developed initially to analyze materials on Earth and study the nature of color, spectroscopy was first used for astronomical purposes in 1814 by Joseph von Fraunhofer. The Bavarian physicist mounted a prism -see fig. 1.1- in front of a telescope and discovered that, when decomposing the light from the Sun, several hundred dark (absorption) lines were observed in the spectrum[2]. But it was not until 1859 that Gustav Kirchoff and Robert Bunsen identified the cause of this phenomenon. The two scientists demonstrated that every chemical element produces a unique spectrum and noticed that the Fraunhofer lines coincided with emission lines identified in the spectra of known gases (Hearnshaw, 2009). In the last few decades, astronomical instrumentation has experimented another revolution due to the embrace of more and more sensitive semiconductor detectors. Nowadays, a 60 cm telescope can perform observations similar to those made with the 5 m Hale telescope at Mount Palomar Observatory (USA) when it was set in operation in 1948 (Wilson, 2013).

Spectrographs have become a fundamental part of astronomy to study the behavior, physical characteristics, and dynamic processes of celestial bodies. Currently, many astronomical observations use telescopes as, essentially, spectrographs. These instruments split or disperse the light[3] from a source into its component wavelengths or frequencies so that the resulting spectrum can be recorded in some multichannel detector. The sources can be, for instance, a star or a galaxy. Regarding detectors, they can be photographic plates or silicon charge coupled devices (CCDs) if we are interested in acquiring optical spectra, or, on the other hand, more finicky arrays if we observe in the near-infrared (NIR).

Many astronomical spectrographs basically contain the same essential elements. In fig. 1.2, a schematic diagram of a single-slit spectrograph is shown. We enumerate these elements in the

[2]Fraunhofer was able to map over 570 absorption lines. He designated the main features with the letters A through K, a system of identification still in use. These bands nowadays are known as the Franhoufer lines.
[3]More precisely electromagnetic radiation.

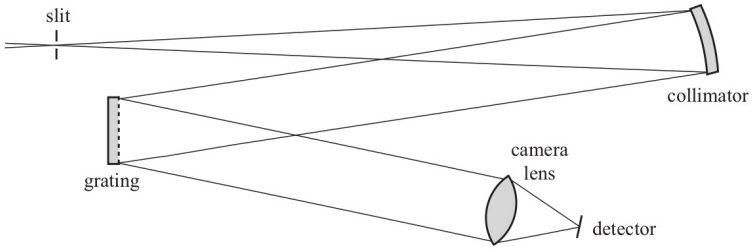


Figure 1.2: A schematic diagram of a slit spectrograph. *Image Credit:* Massey and Hanson (2013).

following list:

- A slit, sitting in the focal plane, limits the spatial extent of the incoming radiation so that the spectra from different locations do not overlap in the instrument focal plane.
- A collimator, which would convert the diverging light beam into parallel light.
- A reflection grating separating the incident beam into its constituent wavelengths.
- A camera that would then focus the spectrum onto the detector.

These instruments are usually characterized by their *spectral resolution* and *resolving power*. The former determines the ability to resolve features in the electromagnetic spectrum. This measure is expressed by $\Delta\lambda$ and is frequently related to the resolving power, denoted by R, as follows:

$$R = \frac{\lambda}{\Delta\lambda} \tag{1.1}$$

where $\Delta\lambda$ is the smallest difference in wavelengths that can be distinguished at a wavelength of λ.

Although in the 1660s Newton split light into a spectrum with the help of a prism, most of the spectrographs nowadays employ diffraction gratings as dispersers. These are planar glass surfaces ruled with a certain number of grooves per mm. Although being more expensive than prisms, gratings are more efficient because (i) the dispersed light is free from any non-linear distortions over the wavelengths dispersed and (ii) the resolving power is higher than those of prisms of comparable sizes. Additionally, there exist fixed and rotating gratings. The first disperse the beam in a definite pattern, whereas the second ones spread the diffraction pattern to cover broad wavelength bands. Finally, it is worth mentioning that there exists a particular grating in which some high-resolution spectrographs in development or in operation are based on. We refer to the *echelle* grating[4], which was devised by George Harrison in 1949 at the Massachusetts Institute of Technology (Harrison, 1949).

[4]Its name comes from French *échelle*, meaning ladder.

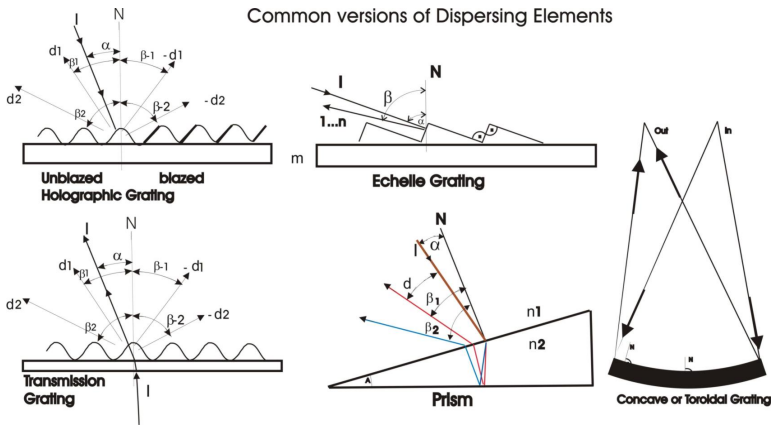


Figure 1.3: A look at several kinds of dispersers. *Image Credit: Neumann (2014).*

The *echelle* is a coarsely ruled grating with a specific groove shape allowing the incident light beam to arrive at a large angle of incidence to the normal; see fig. 1.3. This grating provides a high spectral dispersion at the detector and resolving power, but with small free spectral range (FSR) per order so that multiple orders are needed to cover a standard spectral range (Eversberg and Vollmann, 2015). Using multiple high orders, echelle spectrographs require after the echelle (but sometimes before) a cross-dispersing element, which is often a prism or a conventional low-dispersion grating, or the light is cross-dispersed twice by a prism in a double-pass mode, both before and after (Hearnshaw, 2014). As an example, the optical layout of the Magellan Echellette (MagE), deployed in 2007 on the 6.5 m Magellan II telescope located at Las Campanas Observatory in Chile, is shown in fig. 1.4 (Marshall et al., 2008). As can be appreciated, after passing a slit, the diverging beam is collimated by a mirror. Then, cross dispersion is provided by two prisms, the first of which is used in double pass mode, whereas the second has a single pass.

1.1 Multi-Object Spectroscopy

Multi-object spectroscopy (MOS) refers to the simultaneous recording of the spectra of more than one discrete object over a wide field. Indeed, MOS is actually an old astronomical technique. Although objective prisms had been used previously by Fraunhoffer and Secchi, it was not until later that they were employed for MOS purposes. In 1885, Edward Pickering from the Harvard College Observatory placed a small angle prism over the objective of an 8-in. diameter photographic refractor and recorded the spectra of sixth magnitude stars in as little as five minutes (Pickering, 1912). In this same program, part of the Henry Draper Memorial, the spectra of 10 351 stars were acquired on 633 different photographic plates. This catalog was published

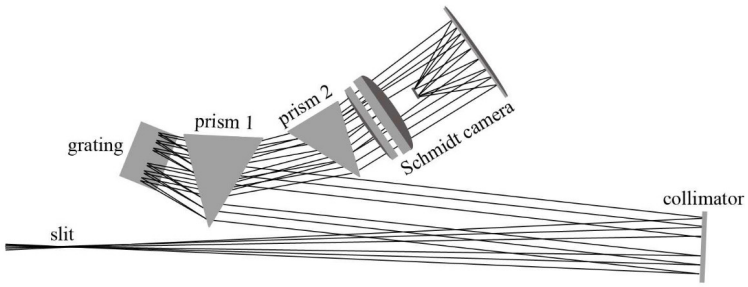


Figure 1.4: The optical layout of MagE deployed in Magellan II telescope. *Image Credit: Massey and Hanson (2013).*

in 1890 by Pickering, being the most extensive and detailed spectral classification of its age (Hearnshaw, 2014). However, since the 1960s, due to its limitations caused by overlapping spectra and sky brightness, as well as variable resolving power, the utility of this *slitless* approach for ground-based observations is very limited.

1.1.1 MOS Techniques

Most MOS capabilities in operation or under construction employ one of the three following techniques: multi-slits, fibers or image slicers based on pick-off mirrors. In this section, we introduce them as well as provide extra resources where further details can be found.

1.1.1.1 Multi-slits and Fiber-based MOS

If we have around tens of objects per square arcmin, which is a relatively high density, and the FOV is in the range of several arcmin, then a multi-slit MOS could be employed. The approach of these instruments is conceptually simple. A typical optical layout can be appreciated in fig. 1.5. Like conventional single-object spectrographs, they have a slit mask but now containing dozens, hundreds or even a few thousands of short narrow slits (or slitlets) at specific FOV locations. Although these MOS offer a large multiplexing[5] advantage, they present some drawbacks as well. First, the design of these masks tends to be quite challenging since many constraints apply (Bacon and Monnet, 2017; Massey and Hanson, 2013). For instance, the slits cannot overlap spatially, and they have to have a minimum length to effectively allow sky subtraction. Besides, positioning more than one slit at any given X position (perpendicular to dispersion) runs the risk of obtaining overlapping spectra. Second, the wavelength coverage may be variable, depending on the location of the slits within the field of interest. Finally, these instruments have traditionally

[5]Multiplexing is the ability to observe a large number of targets in one exposure, at a given pointing.

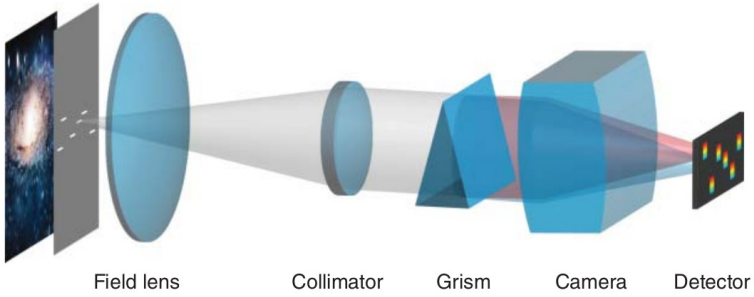
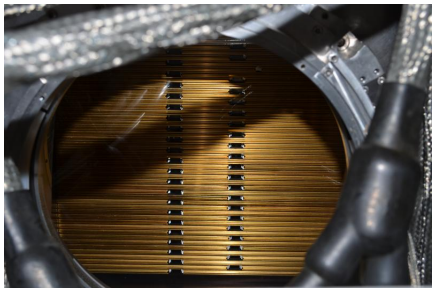
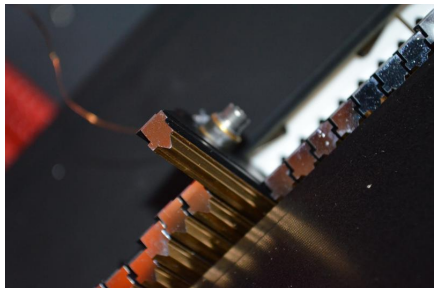


Figure 1.5: The optical layout of a typical multi-slit spectograph. *Image Credit:* *Bacon and Monnet (2017)*.



(a) The EMIR CSU, which is composed of metallic sliding bars. *Image Credit: IAC*.



(b) Detail of the bars in the EMIR CSU. *Image Credit: IAC*.

Figure 1.6: The EMIR configurable slit unit (CSU) ready for integration at Instituto de Astrofísica de Canarias (IAC).

employed for each concrete observation machined masks built well in advance. Therefore, there is no real flexibility while observing at the telescope. Fortunately, this problem has been overcome lately when these pre-formatted masks have been replaced with configurable slit-arrays, which can operate at cryogenic temperatures as well. Such is the case of EMIR, a multi-slit NIR MOS for the 10 m class Gran Telescopio Canarias (GTC) (Garzón et al., 2006, 2014). This instrument offers multiplexing capabilities through 110 different slits, formed by moving opposable bars from both sides of the focal plane and configured with the help of a graphical application; see fig. 1.6. Nowadays, essentially all 8-10 m class telescopes have one or more multi-slit spectrographs in operation, as per table 1.1 and table 1.2 for respectively the optical and NIR domains.

On the other hand, if the density of objects is less than 10 per square arcmin, which is relatively low, but the FOV is in the range of many arcmin, a fiber-based MOS might be suitable. The overall concept of this MOS flavor is to feed a spectrograph with a bundle of fibers arranged

Table 1.1: Optical multi-slit spectrographs with their main characteristics: host telescope, field-of-view (FOV) in arcmin, maximum multiplex (M), spectral resolution (R), spectral range ($\Delta\lambda$) in µm and date of first light. Some of these instruments have been upgraded, while others have been decommissioned (*).

Instrument	Telescope	FOV	M	R	$\Delta\lambda$	First Light	Ref.
LDSS++ (*)	3.5 m AAT	–	40-60	400	–	1986	Robinson (1988)
MOS-SIS (*)	3.58 m CFHT	–	–	1000	–	1992	LeFevre et al. (1994)
DOLORES	3.5 m TNG	$7.8' \times 6'$	–	600 - 6000	–	2000	INAF (2019)
LRIS-B	10 m Keck 1	$7.8' \times 6'$	60	3000 - 5000	0.31 - 0.52	1993	McCarthy et al. (1998)
LRIS-R	10 m Keck 1	$7.8' \times 6'$	60	3000 - 5000	0.52 - 1.00	1993	Oke et al. (1995); Rockosi et al. (2010)
FORS2	8.2 m VLT	$6.8' \times 6.8'$	470	260 - 2600	0.33 - 1.10	1999	Anderson et al. (2017)
FOCAS	8.2 m Subaru	6' diam.	50	250 - 2500	0.34 - 1.00	2000	Kashikawa et al. (2000)
DEIMOS	10 m Keck 2	$16.3' \times 8'$	130	3000 - 6000	0.41 - 1.10	2002	Faber et al. (2003)
VIMOS	8.2 m VLT	$4x7' \times 8'$	1000	200 - 2500	0.36 - 1.00	2002	LeFevre et al. (2003, 2000)
GMOS-N	8.1 m Gemini-N	$5.5' \times 5.5'$	60	630 - 4400	0.36 - 0.94	2003	Davies et al. (1997)
GMOS-S	8.1 m Gemini-N	$5.5' \times 5.5'$	60	630 - 4400	0.36 - 0.94	2003	Davies et al. (1997)
RSS	9.2 m SALT	8' diam.	60	800 - 6000	0.32 - 0.90	2005	Romero (2018); Sheinis et al. (2006)
MODS 1-2	8.4 m LBT	$6' \times 6'$	500	100 - 2000	0.32 - 1.00	2014	Pogge et al. (2010); Wagner et al. (2014)
OSIRIS	10.4 m GTC	$7.5' \times 6'$	250	360 - 2500	0.36 - 1.05	2014	Sánchez et al. (2012)

Table 1.2: NIR multi-slit spectrographs with their main characteristics: host telescope, field-of-view (FOV) in arcmin, maximum multiplex (M), spectral resolution (R), spectral range ($\Delta\lambda$) in µm and date of first light. Some of these instruments have been upgraded, while others have been decommissioned (*).

Instrument	Telescope	FOV	M	R	$\Delta\lambda$	First Light	Ref.
LIRIS	4.2 m WHT	$4.27' \times 4.27'$	Pref-Masks	700 - 2500	0.8 - 2.50	2002	Manchado-Torres et al. (1998)
MOIRCS	8.2 m Subaru	$4' \times 7'$	40	1300 - 3000	0.9 - 2.50	2007	Ichikawa et al. (2006)
LUCI 1-2	8.4 m LBT	$4' \times 4'$	> 100	1900 - 8500	0.90 - 2.50	2008	Buschkamp et al. (2012)
MOSFIRE	10 m Keck 1	$6.1' \times 3'$	46	3500	0.97 - 2.50	2008	McLean et al. (2010)
Flamingo 2	8.1 m Gemini-S	$2' \times 6.1'$	> 100	1200 - 3000	1 - 2.40	2009	Eikenberry et al. (2006b)
EMIR	10.4 m GTC	$6' \times 4'$	110	4000	0.90 - 2.50	2016	Garzón et al. (2006, 2014)

in the focal plane so that they gather light from the different astronomical targets of interest. Then, the fibers are properly regrouped at the entrance slit of a classical long-slit spectrograph; see fig. 1.7. Initially, plug-board fiber systems like Medusa (Hill et al., 1980) were developed. They had a number of holes pre-drilled at specific locations in which the fibers were glued. This approach presents three essential disadvantages: (i) the time-consuming task of preparing the plates, (ii) the lack of operation flexibility at the telescope and (iii) the lack of real-time coarse fiber positioning to maximize throughput. Luckily, in most modern multi-object spectrographs, the fibers are robotically positioned in real-time immediately prior to each observation of a new field.

One of the first automatic positioners, seen in fig. 1.8a, was the 2dF [6] pick and place gripper claw (Lewis et al., 2002) developed by the Australian Astrophysical Observatory (AAO). The fibers are inserted on magnetic buttons, and the robotic positioner sequentially places them into their positions on a metal FOV plate. This system has two field plates, each having a total of 392 fibers. A tumbling mechanism allows the acquisition of up to 392 simultaneous sky targets

[6] For 2-degree field

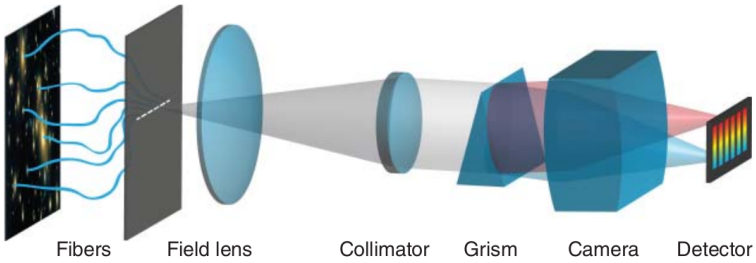
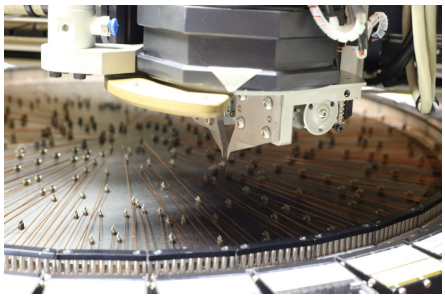


Figure 1.7: Layout of a typical multi-fiber spectrograph. Optical fibers are put on selected targets in the FOV and arranged in a pseudo long-slit at the entrance of a classical long-slit spectrograph. *Image Credit: Bacon and Monnet (2017).*

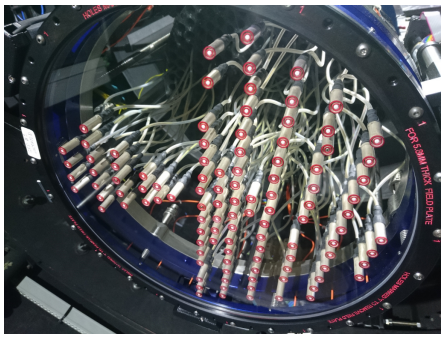
configured on one of the two plate sides, while the robot works to position the fibers for the next field on the other one. Each plate takes about 45 min to complete. Currently, there are two AAO multi-object instruments mounted upon this system: AAOmega (Sharp et al., 2006) and HERMES (Sheinis, 2016), commissioned in 2006 and in 2013 respectively.

In 2012, fiber handling autonomous mini-robots were proposed (Gilbert et al., 2012). Each of these holders, carrying a single fiber, consists of a pair of concentric piezo-ceramic actuators freely moving on a glass plate located on the instrument FOV. Since all fibers can be deployed in parallel, set-up time is in the order of a few minutes. Moreover, by utilizing this positioner, in principle, atmospheric refraction effects due to telescope elevation changes could be compensated by slightly moving the fibers during an exposure. This kind of positioners have been recently demonstrated in the 150-fiber based TAIPAN spectograph installed on the 1.2 m UK-Schmidt telescope at AA0 (Lorente et al., 2018); see fig. 1.8b. This MOS instrument acts as a proof-of-concept for the starbugs technology, proposed for future use in the MANIFEST positioner (Lawrence et al., 2014b). This fiber-based subsystem will be part of the Giant Magellan Telescope, a ground-based extremely large telescope planned for completion around 2025 (McCarthy et al., 2018).

Finally, for more extensive surveys, especially for multiplexing degrees in the order of hundreds and thousands, some instruments have adopted arrays of custom fiber positioner assemblies. These positioners tend to be based on two piezo-electric rotary motors moving a fiber optic anywhere in a limited circular patrol area, often in the order of a few millimeters. That is the case of MEGARA, an optical MOS for GTC, with 92 of its positioners available for MOS observations (Márquez et al., 2018; Pérez-Calpena et al., 2014). There are instruments with a more significant number of fibers. For instance, the Dark Energy Spectroscopic Instrument (DESI) for the 4 m Nicholas U. Mayall telescope on the Kitt Peak National Observatory (KPNO), expected to start operation at the end of 2019 (Flaugher and Bebek, 2014). This instrument, designed to measure the effect of dark energy on the expansion of the universe, will possess about 5000 fiber positioners. Finally, the multiplexing system for the Prime Focus Spectrograph (PFS) instrument, installed at

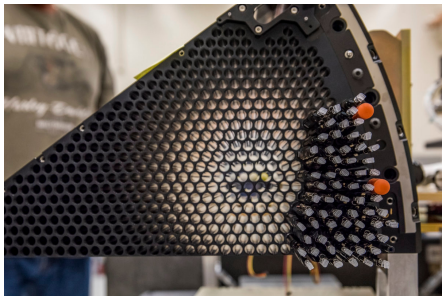


(a) The 2dF fiber positioner with its grasp on the upper side of the picture. Fibers, inserted on a metallic clip, are magnetically attached to the plate. *Image Credit: AAOmega+2dF User Manual (AA0)*.

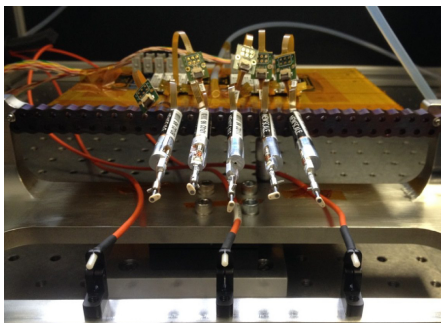


(b) Starbug fiber positioners attached to the the field plate of TAIPAN. *Image Credit: David Brown (AA0)*.

Figure 1.8: Two different fiber positioners in use at the AA0.



(a) DESI focal plane assembly with a few fiber positioners placed in the array. *Image Credit: Stanford Linear Accelerator Center (SLAC)*.



(b) Several Cobra fiber positioners for PFS instrument. *Image Credit: Fisher et al. (2014)*.

Figure 1.9: Fiber assemblies for DESI and PFS.

the 8.2 m Subaru Telescope on Mauna Kea, Hawaii, consists of 2400 optical fibers. Each fiber is 65 m in length and is placed by a two-rotary motor positioner known as Cobra (Fisher et al., 2014). These positioners patrol circular areas of 9.5 mm in diameter and are packed in a hexagonal pattern with 8 mm separation. In fig. 1.9, details of the positioners and fiber assemblies for DESI and PFS are shown.

In general, and regardless of the positioning system utilized, the fiber-based MOS approach produces much more straightforward and practical systems than the multi-slit approach. Besides, fiber-fed instruments present several benefits. First, the cost and construction time of this kind

Table 1.3: Optical multi-fiber spectrographs with their main characteristics: host telescope, patrol field area in square degree in square degree, maximum multiplex (M), spectral resolution (R), spectral range ($\Delta\lambda$) in µm, planned survey, and date of first light. Some of these instruments have been upgraded, while others have been decommissioned (*).

Instrument	Telescope	Field	M	R	$\Delta\lambda$	Survey	First Light	Ref.
AutoFib2	4.2 m WHT	–	150	100 - 9500	–	–	1994	Lewis et al. (1997)
2dF	3.9 m AAT	3.1	400	1300	0.47 - 0.85	5×10^5 galax.	1997	Lewis et al. (2002)
SDSS	2.5 m Apache	3.1	640	1000	0.38 - 0.91	1.5×10^6 galax.	2000	Owen et al. (1994)
Hectospec	6.5 m MMT	–	300	500 - 3000	0.35 - 1.0	–	2004	Fabricant et al. (1994)
SDSS	2.5 m Apache	3.1	300	22500	1.51 - 1.70	1×10^5 stars	2011	Ge et al. (2009)
SDSS	2.5 m Apache	3.1	1000	2000	0.36 - 1.04	1.5×10^6 galax.	2011	Ge et al. (2009)
LAMOST	4 m LAMOST	20	4000	1800	0.37 - 0.91	1×10^6 galax.	2011	Su et al. (1998)
HERMES	3.9 m AAT	3.1	400	28000	0.47 - 0.79	1×10^6 galax.	2013	Raskin and Van Winckel (2008)
VIRUS	9.1 m HET	–	400	850	0.48 - 0.93	–	2015	Hill et al. (2016)
MEGARA	10.4 m GTC	3.5	100	5500 - 12000 - 20000	0.37 - 0.98	–	2017	Márquez et al. (2018)
WEAVE	4.2 m WHT	3.1	1000	5000	0.37 - 0.95	2×10^7 galax.	2018	Dalton et al. (2012)
WEAVE	4.2 m WHT	3.1	1000	28000	0.37 - 0.95	1×10^6 stars	2018	Dalton et al. (2012)
DESI	4 m Mayall	8	5000	2000 - 5500	0.36 - 0.98	30×10^6 stars	2019	Flaugher and Bebek (2014)
PFS	8.2 m Subaru	1.5	2400	3000	0.37 - 0.85	Legacy Surv.	2020	Tamura et al. (2016)
DESpec	4 m Blanco	3.8	4000	3000	0.55 - 0.95	8×10^6 galax.	2020	Saunders et al. (2012)
4MOST	4.1 m VISTA	7	2400	5000	0.37 - 0.95	2×10^7 galax.	2021	de Jong et al. (2016)
4MOST	4.1 m VISTA	7	2400	2000	0.37 - 0.95	1×10^6 stars	2021	de Jong et al. (2016)

Table 1.4: NIR multi-slit spectrographs with their main characteristics: host telescope, patrol field area in square degree in arcmin, maximum multiplex (M), spectral resolution (R), spectral range ($\Delta\lambda$) in µm and date of first light.

Instrument	Telescope	Field	M	R	$\Delta\lambda$	First Light	Ref.
Hydra	3.5 m WIYN	0.8	93	1000 - 10000	0.40 - 1.80	2005	Barden and Armandroff (1995)
FMOS	8.2 m Subaru	0.2	400	600 - 2200	0.90 - 1.80	2012	Maihara et al. (2000); Tamura et al. (2012)
PFS	8.2 m Subaru	1.5	2400	3000	0.90 - 1.45	2020	Tamura et al. (2016)
MOONS	8.2 m VLT	0.14	1000	4000 - 20000	0.65 - 1.80	2020	Cirasuolo et al. (2014)

of instruments can be significantly reduced, since the spectrograph can be placed in a fixed, controlled environment only by increasing the length of the fibers. Second, fiber outputs can be configured along a curved convex slit, which is an ideal input to a spherical mirror acting as the collimator. However, these advantages are neutralized by three notable downsides. First, since fibers tilt to address their respective targets, a significant light fraction is lost at the slit end. These losses can be as high as a factor of 3 or more compared to a conventional slit spectrograph. Second, these systems present *etendue loss* if the fibers are long or/and bent. Fortunately, this effect can be frequently compensated by adding a microlens to the fiber entrance (Bacon and Monnet, 2017). Finally, sky subtraction cannot be resolved locally, and its associated accuracy depends on the degree of precision achieved when removing the fiber-to-fiber transmission variations by flat-fielding (Massey and Hanson, 2013). In table 1.3 and table 1.4, we have grouped the most representative multi-fiber MOS facilities working in optical as well as in NIR wavelengths.

1.1.1.2 Image Slicers based on Pick-off Mirrors

Back in the 1980s, the scientific community was looking for ways to acquire larger FOV in a single exposure, enabling, this way, the study of a given extensive spatial object. This could not be

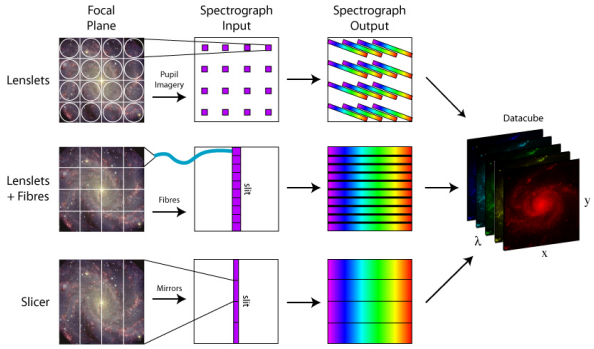
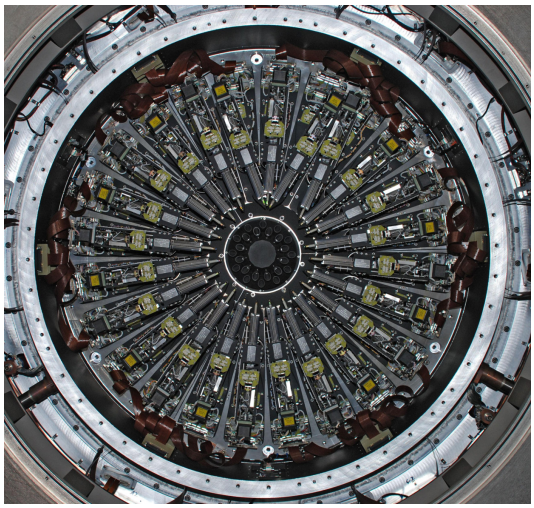


Figure 1.10: The three different techniques employed by IFUs: lenslets, fibers and image slicers. Each IFU present in the instrument FOV directs its beam to a dispersive element producing spectroscopic information for each slice. Pipeline processing algorithms combine the dispersed spectra into a 3D data cube. *Image Credit: M. Westmoquette*.

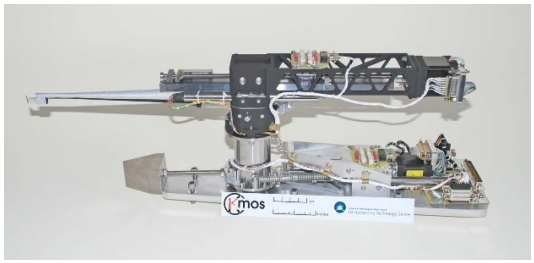
done in a cost efficient manner with traditional long-slit spectrographs; consequently, the need for integral field spectroscopy (IFS) naturally arose. This technique, also known as 3D spectroscopy, consists in dividing a field into multiple (X, Y) spatial components and individually dispersing each component with a spectrograph in order to obtain its associated spectrum. Every integral field spectrograph is formed by several integral-field units (IFU), each sampling individual spatial elements of the astronomical scene. There exist several approaches for decomposing a field and reformatting it into one or more slits (Calcines et al., 2018). However, regardless of the technique used, the final product is frequently a data-cube containing spatial information as well as wavelengths.

In broad terms, IFUs can gather light using one of three following technologies: lenslets, fibers (with and without lenslets) or image slicing mirrors. Figure 1.10 illustrates these main methods. By way of example, MIRADAS probe arms are all-mirror image slicers that can be used for MOS purposes. The multi-object effect can be achieved by deploying the available IFUs over a large patrol field, each of which centered on a promising celestial target. Image slicers mainly present three advantages: (i) they are the only IFU mode to preserve all spatial information, (ii) they are also the most compact at reformatting the focal plane onto the detector and (iii) they can work in cryogenic systems and at long wavelengths. However, they suffer from scattered-light from the slicing mirrors.

Before MIRADAS, deployable IFUs based on pick-off arms were designed for IRMOS (Eikenberry et al., 2006a) as well as for KMOS (Sharples et al., 2006). The latter is a second-generation instrument operating on ESO's Very Large Telescope (VLT) that is able to perform IFS in the NIR bands for 24 targets simultaneously. Each of these targets is sampled by a different deployable image-slicing IFU. The 24 arms are evenly distributed around two different planes, one above



(a) A bird's eye view of the 24 probe arms of KMOS arranged in two different horizontal planes. *Image Credit: KMOS's User Manual Period 104*.



(b) The probe arm based IFU of KMOS. The mechanism is composed of a vertical pivot and an extensible horizontal bar, in one of whose ends there is a pick-off mirror to collect light. *Image Credit: KMOS's User Manual Period 104*.

Figure 1.11: ESO's KMOS bench with its 24 arms.

and other below the telescope focal plane; see fig. 1.11b. The KMOS arm, as shown in fig. 1.11a, is an open kinematic chain of two degrees of freedom (DoF), each driven by one motor capable of operating in cryogenic temperatures. The T-shaped mechanism consists of two tubes and two independent joints providing to the structure a polar (r,ϕ) motion. A revolute joint in the arm base is responsible for the angular (ϕ) motion. A prismatic (sliding) joint, above the revolute joint and connected to it by a vertical tube, controls the translational or radial (r) motion. The pick-off mirror which diverts the input beam along the arm is located at the end of a horizontal bar sliding along the prismatic joint. This variable path length of the beam needs to be compensated by an optical trombone. In table 1.5, the most notable MOS employing IFUs based on pick-off

Table 1.5: MOS capabilities using deployable IFUs based on image-slicers: host telescope, patrol field in square degrees, maximum multiplex (M), spectral resolution (R), spectral range ($\Delta\lambda$) in µm and date of first light.

Instrument	Telescope	Field	M	R	$\Delta\lambda$	First Light	Ref.
KMOS	8.2 m VLT	0.011	24	4200	0.80 - 2.5	2014	Bennett et al. (2008); Sharples et al. (2006, 2004)
MIRADAS	10.4 m GTC	0.006	12	20000	1 - 2.5	2020	Eikenberry et al. (2014, 2016)

probe arms are shown.

The interested reader is referred to Bacon and Monnet (2017), Bershady (2010), and Calcines et al. (2018) for a comprehensive discussion of all mentioned techniques and technologies.

1.1.2 Science Perspectives

Since the late eighties, the demand for astronomical MOS capabilities has expanded significantly and, nowadays, MOS has become a major path for future instrumentation. Indeed, the need for wide-field, high-multiplex spectroscopic facilities has been identified in a number of strategic documents (Bode, 2012; de Zeeuw and Molster, 2007; Turon et al., 2008). More specifically, the community has shown a great interest in a follow-up spectroscopic analysis to the impressive imaging surveys carried out during the last decade. In order to make the most of all the targets delivered, investing in new MOS facilities is crucial. Three clear policies appear in that direction. First, the conversion of existing 4 m class telescopes for long-term multi-fiber optical surveys. Second, development of multi-fiber instruments on common-user 8 m class telescopes providing extensions in the NIR spectral region. Finally, construction of new panoramic integral field units on 8-10 m telescopes offering the capability of undertaking 'blind' spectroscopic searches. In the following subsections, we mention the fundamental science drivers supporting the current trend for multi-object spectrographs. It is impossible to discuss all science cases as well as the many MOS instruments in operation or coming soon. For further details, see Skillen et al. (2016).

1.1.2.1 Milky Way and Stellar Physics

Many unsolved questions in astronomy could be addressed by massive MOS surveys. For instance, the archaeology of the Milky Way, especially how our galaxy formed and evolved to adopt the present appearance or how many pieces were combined to constitute its halo. More in general, the Milky Way can be employed as a model since astronomers have recognized that our galaxy and its immediate environment offers a unique opportunity to understand the laws governing the formation of disc galaxies (Freeman and Bland-Hawthorn, 2002). Knowing the ages, masses, element composition and orbits of the stars in the Milky Way system will respond to several questions concerning: (i) the galactic gravitational potential and the properties of dark matter, (ii) the formation history and memory of a prototypical large galaxy, and (iii) stellar physics and the origin of the chemical elements (Ellis et al., 2017).

Gaia (Lindegren et al., 1994), the space observatory of the European Space Agency (ESA) launched in 2013, is already providing accurate distances, luminosities, and proper motions for one billion stars, and stellar spectra for 60 million stars (Skillen et al., 2016). The ultimate goal is to have measures for the joint distribution of kinematics, physical properties, and abundances for all stars in the Galaxy and its satellites. This will provide a complete overview of the star formation and chemical evolution history of the Milky Way and its nearest satellites. Several massive ground-based MOS instruments are currently in operation or being built to make progress in the computation of radial velocities with accuracy 1-5 $\mathrm{km\,s^{-1}}$ as well as abundance determinations for many celestial bodies. The most relevant related instruments are the following multi-fiber spectrographs: (a) HERMES at the Anglo-Australian telescope (b) WEAVE at 4.2 mm William Herschel telescope (c) MOONS, mounted at a Nasmyth focus at the ESO's VLT, and (d) 4MOST at the ESO's VISTA telescope. The specifications and features of these instruments are described in detail in de Jong et al. (2016), Sheinis (2016), Cirasuolo et al. (2014), and Dalton et al. (2012).

On the other hand, a homogeneous spectroscopic survey of several million stars at R ~ 5000 and hundreds of thousands at R ~ 20 000 would create opportunities for many studies of stellar physics. The key would basically be to exploit the quality and homogeneity of the data as well as the ability to tap on exotic phases of stellar evolution thanks to the large sizes of the surveys.

1.1.2.2 Galaxy Evolution

Although there is consensus in the community about some aspects of galaxy formation, many questions about evolution remain without an answer. As an example, how baryons made it from intergalactic gas clouds to the stars that populate galaxies today. Roughly, there is a growing interest in performing extensive studies about hydrodynamics in the cosmic web, gas cooling, shock heating and star formation in a range of epochs and interplay between dark and luminous matter, among others.

In the past two decades, the power of MOS capabilities for addressing several aspects of galaxy evolution was made evident by surveys on 10 m class telescopes. Nowadays, MOS galaxy evolution studies are expanding into new paths of discovery, especially with the employment of cryogenic instruments such as MOSFIRE, FMOS, KMOS, EMIR, and MIRADAS. With these MOS instruments, the old dream of observing galaxies at NIR wavelengths has started to come true. In parallel, 3D-spectroscopy will provide a new set of clues into galaxy formation coming from spatially-resolved dynamical and stellar-population information (Skillen et al., 2016).

1.1.2.3 Cosmology

The big questions in this area, nowadays, primarily focus on the nature of dark energy and matter, and the physics of inflation. In this direction, MOS can provide vital information on cosmological parameters. New instruments such as DESI and HETDEX are currently being built

to address the main wish of cosmologists: higher multiplexing power as well as wider fields. One of the challenges of such instruments is to increment the accuracy with which probes constrain cosmological parameters, and, specifically, to increase the constraints on the dark energy equation of state (Skillen et al., 2016).

1.2 Thesis Overview

The overall aim of this thesis is to propose techniques enabling the determination of a global solution for the observation of the multiple astronomical targets selected by a concrete MIRADAS user. In particular, these techniques, once correctly combined, should be able, given a set of celestial objects, to compute one or more plans to safely place the pick-off mirrors of the MIRADAS MXS probe arms in the targets of interest. This task is common to all MOS having opto-mechanicals devices moving in a shared environment; however, the specific algorithms resolving it chiefly depend on the characteristics of each different instrument. Not only that, although similar in concept, most MOS instruments differ in the mechanisms employed to acquire light from the telescope. Precisely, these devices are the ones constraining and imposing restrictive requirements to the design of solutions. Therefore, it is crucial to pay attention and understand how these opto-mechanical elements work.

From a bird's eye perspective, the problem to address is simple. Consequently, at first sight, it is tempting to try to tackle it by following a global, compact, and integrated approach. It is more than that since common sense says that a global solution would be the most beneficial in terms of performance. However, after careful analysis, one becomes aware of the complexity and the vast number of variables involved in the problem to solve. Considering this, a *divide and conquer* strategy has been adopted in this thesis. The original problem has been broken down into three smaller sub-problems, which are simple enough to be solved directly. This work makes contributions in the three following areas: (i) segmentation of scattered fields, (ii) target allocation and (iii) motion planning.

The algorithms proposed for each of the three previous topics once properly interconnected should provide a solution to the overall problem of efficiently observing a given set of user-defined sky locations. This high-level piece of software gluing together the techniques introduced in the following chapters is out of the scope of this thesis. Among other tasks, this upper layer computational logic should be able to prioritize and adequately schedule observations to make rational use of the telescope time. This might result in successive calls to either all or some of the algorithms proposed with different parameters or subsets of the original target set. A very general overview of one possible processing flow is introduced in one of the chapters of this manuscript. We believe it can help the reader to get a better picture of the global problem. A brief outline of the content of this document is provided below.

Chapter 2 provides an overview of the MIRADAS instrument for GTC. Although the instru-

ment has many subsystems, the chapter only summarizes those aspects that are fundamental to understand the motion techniques presented in this work. Moreover, we provide a short description of the MIRADAS trajectory generation data flow as well as the necessary connections between the distinct components to successfully accomplish the task of safely moving the arms of the multiplexing system (MXS).

Chapter 3 focuses on the characteristics of the MXS probe arm, an image-slicer IFU with a pick-off mirror collecting light from a celestial target of interest. Although this mechanism has several optical elements to relay the beam to the spectrograph, we discuss only those aspects directly impacting the design of the motion algorithms proposed in this thesis. Concretely, one contribution of this thesis consists in the proposal of a mathematical model and an in-depth study of the arm behavior. Also, other important issues are also addressed. First, we present different arm patrolling strategies. Second, we discuss the particularities of the controller employed to govern the probe arm actuators. They, as we will see, impose several constraints to be fulfilled by the explored motion techniques.

Chapter 4 delves into the first component of the process generating trajectories for MIRADAS probe arms to reach a given set of user-defined targets. This initial step has been identified as characteristic of this instrument since, to the best of our knowledge, only one other MOS instrument requires it. In particular, this chapter presents how a scattered field can be appropriately segmented to be sequentially observed by MIRADAS. First, it gives insights about the different methods employed in data science to decompose a large data set into several families, each possessing different features. Then, a solution is proposed that, in addition to gathering together similar data, it also delivers telescope pointing information for each group of targets.

Chapter 5 turns to the sub-problem of target allocation. This task, given a telescope pointing location as well as a set of targets fitting in the MIRADAS FOV, assigns an arm of the MXS system to every target according to some concrete criteria. More specifically, two distinct approaches are provided. The first employs a popular metaheuristic, whereas the second applies a mathematical method to obtain an optimal assignment plan. Indeed, the two methods are complementary. Being an exact method, the mathematical formulation, although returning the best possible plan, results impractical with very large data-sets. In these situations, as it will be discussed, the approximated approach will provide a good enough solution in a decent time.

Chapter 6 is devoted to motion planning, the final stage in trajectory generation. In this last element of the data flow, synchronized collision-free motions are generated for a given target allocation plan. Once adequately converted to the particular controller commands and delivered to the low-level software, the arms will parallelly execute these motions. In this chapter, first, an overview of related work in the field is provided, especially for fiber-based MOS as well as for probe arm based instruments. Although we are treating motion planning in the context of astronomical instrumentation, the problem can be fundamentally addressed as a multi-robot system problem. Hence, we also give an elementary but systematic exposition of the main ideas

in the state of the art of this area of knowledge. This material is included partly because of his inherent interest as well as for research directions that could be considered in the future. Afterward, we introduce and thoroughly discuss the approach followed in MIRADAS to determine collision-free motions for all arms. Then, we demonstrate how our contributed algorithm behaves in several scenarios.

The last chapter gives an overall summary of the work presented, along with suggestions for future work that can be developed based on the advances done in this thesis.

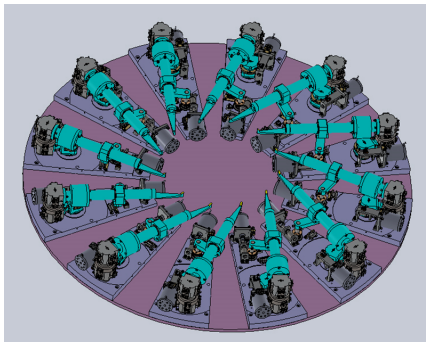
CHAPTER 2

MIRADAS: A NIR MOS FOR GTC

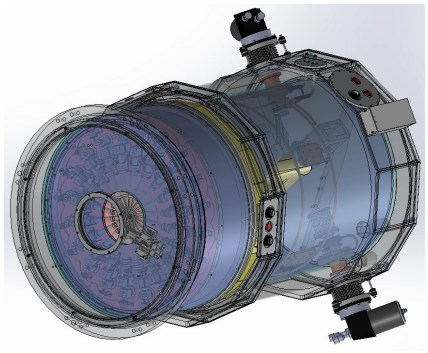
The Mid-resolution InfRAreD Astronomical Spectrograph (MIRADAS) is a NIR multi-object echelle spectrograph operating at spectral resolution R = 20000 over the 1-2.5 µm bandpass (Eikenberry, 2013; Eikenberry et al., 2012, 2016; Raines et al., 2018). It was selected in 2010 by GTC partnership as the next-generation NIR instrument for the world's largest optical/infrared telescope and is expected to be commissioned in 2020. This MOS facility is being developed by an international consortium including the University of Florida (UF), Universitat de Barcelona, Instituto de Astrofísica de Canarias, Institut d'Estudis Espacials de Catalunya, and Universidad Nacional Autónoma de México.

Many instruments being developed for the current generation of large telescopes and the future Extremely Large Telescopes (ELTs) contemplate the utilization of *deployable* IFUs (d-IFUs), especially for cryogenic applications. Indeed, MIRADAS is characterized by its MXS bench containing 12 robotic d-IFUs in the form of optical image-slicing probe arms. Although the use of this sort of pick-off opto-mechanical devices is not new, the ones included in MIRADAS present a fixed-optical path concept that makes them different to those employed in other instruments (i.e. KMOS). The MIRADAS cryogenic probe arm relies on the optical design developed by UF for the next-generation High-Resolution Near-Infrared Spectrograph (HRNIRS) for the Gemini Observatories (Eikenberry et al., 2006c,d). A similar concept was proposed by the initial feasibility studies of the IRMOS instrument for the Thirty Meter Telescope (Eikenberry et al., 2006a). In fig. 2.1 a complete view of MIRADAS as well as a drawing for the MXS bench is shown. Pictures of nine real arms in the laboratory and the instrument cryostat can be appreciated in fig. 2.2 and fig. 2.3 respectively.

MIRADAS is one of the most powerful astronomical instrument of its kind ever envisioned. When installed, this MOS facility will offer to the scientific community an observing efficiency



(a) Drawing of the MXS bench with the 12 probe arms.



(b) Drawing of the cryostat of MIRADAS with the MXS bench.

Figure 2.1: Drawings of several parts of MIRADAS.

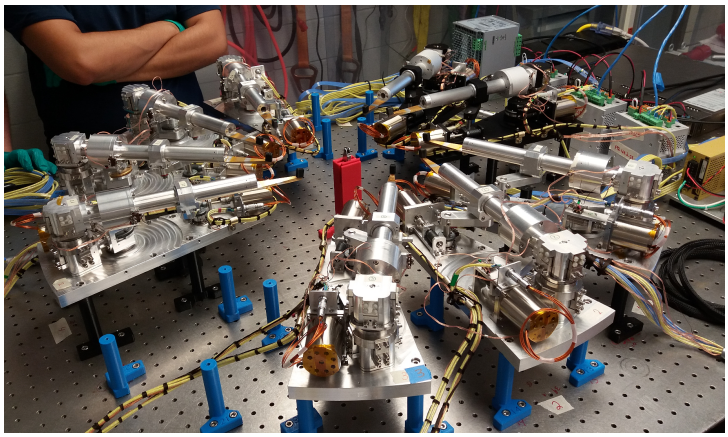
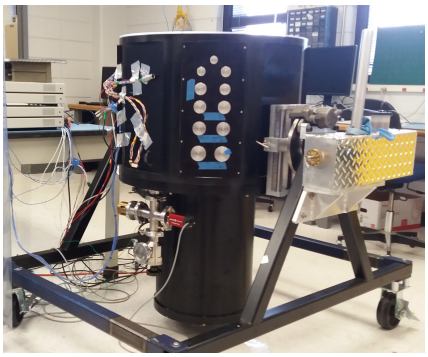


Figure 2.2: Nine real MXS probe arms in the laboratory ready for testing. They were arranged as if they were in the MXS bench.



(a) MIRADAS test cryostat at the UF lab.

(b) Installation of CIRCE (Edwards et al., 2006; Garner et al., 2016) at one of the four available GTC Folded Cassegrain foci. The cylindrical cryostat of this instrument is similar to the one in MIRADAS. *Image Credit: GTC*.

Figure 2.3: Cryostats for cryogenic instruments to be installed at the GTC.

more than an order of magnitude higher than current capabilities for 10 m class telescopes. Besides, the combination of the large collecting area of the host telescope and the capability of the instrument of providing multi-object mid-resolution near-infrared spectra, positions MIRADAS as a clear candidate to give answers to some of the leading scientific challenges of the coming decades. In particular, the MIRADAS science team, comprised of 39 scientists from 8 different institutions, has proposed four key science cases for the instrument. Contained in the document *Design Reference Cases* for MIRADAS, these scenarios provide the fundamental requirements drivers for the instrument. Experience has shown that most astronomical facilities scientifically outperform their original justifications. This is in part because of the imagination of astronomers, who tend to push each facility beyond its original objectives. It is incredibly complicated predicting the evolution of science decades in the future. However, more than ten additional science cases have been identified, which will make excellent use of the MIRADAS capabilities, ranging from stellar magnetic fields to galactic archeology of the Milky Way.

2.1 Science Cases

Ultimately, the success of an astronomical instrument is defined by the science it produces. Here, we detail the four science cases that are the driving motivation of the design and construction of MIRADAS.

2.1.1 Massive Stars in the Milky Way

Being one of the dominant luminosity sources in spiral galaxies, high-mass stars, and especially their evolutionary life cycles and death, have become one of the most popular fields of study in modern Astrophysics. They can help us to understand the mysteries of the history of galaxies across cosmic time since these stars are key contributors to abundance evolution in the Universe. Also, they are the primary source of turbulent energy and star formation feedback to the interstellar medium. Additionally, these stars are the source of super-luminous explosions such as gamma-ray bursts and hypernovae as well the progenitors of neutron stars and black holes. MIRADAS capabilities will allow the first major survey of the majority of these previously-hidden objects in the Milky Way and nearby galaxies.

2.1.2 Chemo-Dynamical Surveys of the Inner Galaxy

Knowledge of the abundance patterns in all the major components of the inner Galaxy will permit scientists to establish connections and finally draw conclusions about its formation and evolution. To accomplish this, measures have to be taken at a variety of locations, consequently, being indispensable simultaneously observations. Moreover, little is known about the link between dynamics, star formation, and chemical history. Hence, investigations employing a number of elemental tracers are needed to explore the limitations for the observations and provide the exact number densities of stars. MIRADAS is expected to make breakthrough contributions in this field because it will offer to users more than ten times the observational power previously available for such work.

2.1.3 Infrared Spectro-Polarimetry

MIRADAS will provide new opportunities for stellar studies. Astrophysical plasmas and magnetic fields are inextricably coupled to each other. Unfortunately, the latter cannot be studied employing traditional methods. They are known to produce many signatures, evident in the polarization of light, providing relevant quantitative information about them. The light coming from most natural sources is only weakly polarized, and the associated signals are difficult to detect. Therefore, studying these effects in stars other than the Sun requires spectro-polarimetry from 10 m class telescopes such as the GTC.

2.1.4 Building Blocks of Galaxy Evolution at Intermediate Redshift

One of the most important issues of observational cosmology is to understand how and when the galaxies built up their stellar mass and acquired their morphology. Blue Compact Dwarfs galaxies (BCDs) are those dwarfs with low luminosity, strong emission lines superposed on a blue continuum, and compact optical size. BCDs undergo an intense burst of star formation, which makes them extremely interesting for studying how this process works and the physical

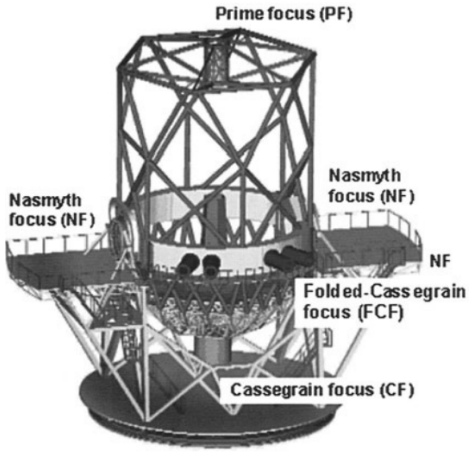


Figure 2.4: A drawing of GTC with the different foci available. Note the 4 Folded Cassegrain foci. MIRADAS will be installed in one of them. *Image Credit: Kohley (2000)*.

conditions under which it can be triggered, maintained, or quenched. MIRADAS will facilitate the kinematical analysis of numerous samples of BCDs at intermediate redshift, crucial for better understanding of the main processes involved in galaxy assembly and evolution.

2.2 Instrument Overview

The MXS system patrol field for MIRADAS is 5′ in diameter, or about 250 mm. The primary function of the MXS system is to select user-defined target FOVs and relay light from them to the rest of the spectrograph. This is accomplished employing 12 independently-controlled/actuated probes with pick-off mirror optics. Each probe arm patrols a sector of the MXS field (discussed in chapter 3), just above the input telescope focal plane, and feeds a 3.7×1.2 arcsec FOV to the spectrograph. The spectrograph input optics include a *slit slicer* reformatting each probe field into 3 end-to-end slices of a fixed 3.7×0.4 format. This approach combines the advantages of minimal slit losses in any seeing conditions better than 1.2 arcsec, while providing a narrow slit as well. The latter is crucial for the high resolution of MIRADAS in a Folded Cassegrain focal environment; see fig. 2.4. This reformatting also allows some (limited) two-dimensional spatial resolution. Furthermore, the spectrograph optics support a range of configurations, providing a high degree of versatility. Depending on the needs of the science program, the observer will be able to choose between maximal multiplex advantage and maximal wavelength coverage, with several in-between options. The basic parameters of MIRADAS are given in table 2.1.

The MIRADAS d-IFU is an optical relay consisting in a camera/collimator re-imager located

Table 2.1: MIRADAS general parameters

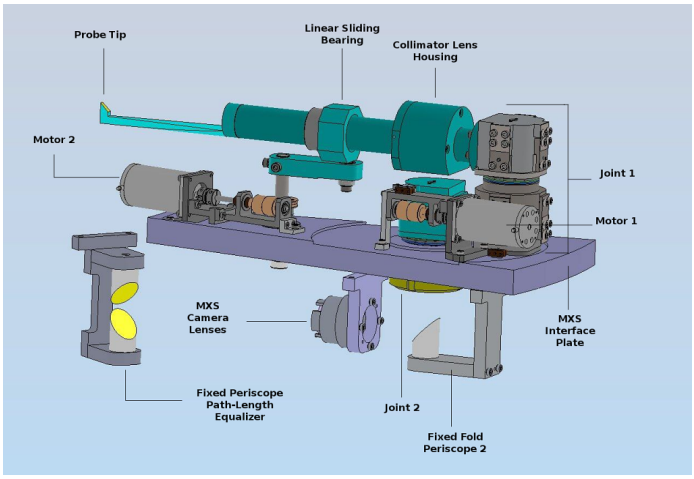
Parameter	Value	Comment
Probe field of view	3.7 arcsec×1.2 arcsec	12 probes
Slit slicer geometry	3 slices × 3.7 arcsec×0.4 arcsec	
Detector focal plane	$(4096 \times 2048) pixels$	(1 × 2) H2RG arrays

on the articulated probe arm. We show a view of the MXS probe mechanism and a layout for the optics in fig. 2.5. A pick-off mirror located near the telescope focal plane relays light down the probe arm, where it encounters a collimating doublet lens. The lens feeds light through a series of folds in the probe mechanism, which maintains a fixed optical path length while the probe arm is moved to variable target locations in the field of regard. The 4^{th} fold mirror is under the MXS optical bench and is fixed in its place and orientation. Shortly after the fixed fold comes the cold pupil stop, located at the pupil image created by the collimator doublet. Finally, a re-imaging doublet brings the beam to a focus.

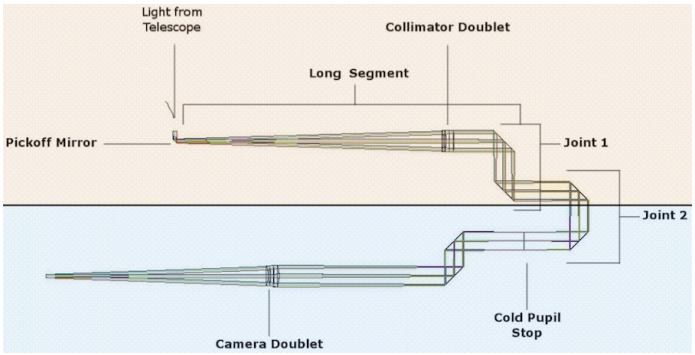
2.2.1 The MXS Control Software

As the tips of the MXS articulated mechanisms approach the center of the MXS bench, room for maneuvering becomes scarcer and scarcer. Moreover, to make efficient use of the telescope's observation time, the MXS system requires a control unit capable of simultaneously moving all probe arms while, at the same time, avoiding collisions between each other. Therefore, in such a cluttered environment, it is vital to implement a piece of software computing and safely coordinating the trajectories of the arms.

The several solutions proposed in this thesis form part of the MXS control software. The ultimate goal of this piece of software is to place the pick-off mirror of each probe arm in a particular sky location previously defined by a scientist. Although simple in concept, to accomplish this task, the system relies on several basic software packages appropriately connected. We can basically distinguish two different kinds of operations: *off-line* and *on-line*. The first refers to planning operations executed well in advance prior to observing time and carried out by a piece of high-level software running in a desktop machine. The second group contains those operations commanding the instrument, which are typically run at night during observation. These operations are performed by real-time low-level control software interacting with the real hardware. Among these operations, there is one that is responsible for the proper execution of the arms motions previously planned. Note that only off-line operations are within the scope of this thesis.



(a) A view of MXS probe arm with several optical elements



(b) MXS probe optical layout. The top (pink) shaded portion includes the moving parts of the probe arm, whereas the bottom (blue) shaded portion includes the fixed parts of the probe arm optical train which are below the MXS bench.

Figure 2.5: Drawings of the MXS probe arm.

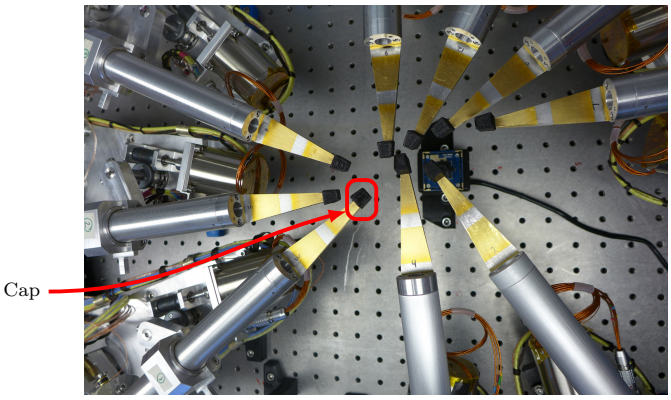


Figure 2.6: Nine MXS probe arms at their final locations as delivered by the target allocation algorithm. The pickoff mirrors were covered by a black rubber cap during testing.

2.2.1.1 Off-line operations

Provided that all science targets fit in the FOV of the instrument, the arms can be virtually positioned on the targets sequentially executing two different processing steps. The first is *target allocation*, which, roughly speaking, determines a set of <*arm, target*> pairs. In two different situations, the allocator can alternatively decide to position one or more arms at blank sky. First, if the MXS system has some arms not working and their real locations are known in advance. Secondly, if during computation, for any reason, a working arm cannot be assigned to a target, then the allocator will place the arm at a predefined *park* position or any other convenient location. In fig. 2.6, a real assignment plan for nine arms delivered by one of the target allocators presented in this thesis is shown.

Once a feasible assignment plan is obtained, then, a second step, known as *motion planning*, is run. It determines the motions of the arms so that every arm safely reaches its assigned target/location from its current placement. The motion planner delivers collision-free trajectories expressed in an actuator independent format. A preconditioning phase, present in the on-line part of the control software, will adequately convert them to actual motor controller commands.

As MIRADAS has been conceived as a GTC common-user instrument for many years to come, it does not restricts at all the sky objects fields it can work it. Hence, the user-defined targets might be spread over an area wider than the instrument's FOV. In such cases, an additional processing stage has to be carried out before the two previously mentioned. This process distributes the targets in different groups so that all elements in a batch fit in the FOV region. Additionally, it will also automatically compute telescope pointing information.

Although it is beyond the scope of this work, we mention that if the motion planner presented fails to compute trajectories, then a re-planning should be performed but, this time, with a

different set of algorithm parameters. If after several attempts, a successful plan is still not found, problematic targets might be temporarily discarded and a new assignment plan computed. Then, the motion planner would be called again with this new determined assignment plan. All this logic properly wiring these two processing steps should be implemented in a higher software layer, which, as commented before, is not included in this thesis. Note that all these different algorithm calls would happen well before the observations. Consequently, there should be no a priori impact on observation efficiency.

CHAPTER 3

THE MXS PROBE ARM

Before discussing the proposed motion-related solutions for MIRADAS, we are going to introduce some of the key concepts that underlie them. In particular, we will turn our attention to the characteristics of the opto-mechanical device employed in the MXS system to observe sky targets. This multiplexing system has 12 robotic detachable probe arms with pick-off mirrors working in a cryostat, each of which patrolling a limited region of the instrument FOV. Light is relayed from the astronomical targets down to the spectrograph through a series of optical elements located inside each probe arm. As we will see, the tubes and bars forming the mechanical structure are coupled in such a way (loop) that they provide a high level of stability when the arm works reversed. However, this convenient arrangement from the point of view of the optics results in a non-intuitive kinematic behavior, making the control of the probe arm more difficult.

Parts of this chapter have previously been published as: (i) "Kinematic modeling and path planning for MIRADAS arms", Proc. SPIE 9151: Advances in Optical and Mechanical Technologies for Telescopes and Instrumentation, 91515S, 2014 (Sabater et al., 2014) and (ii) "Roadmap search based motion planning for MIRADAS probe arms", Journal of Astronomical Telescopes, Instruments, and Systems 4(3), 034001, 2018. (Sabater et al., 2018a)

3.1 The Mechanism and its Linkage Model

A mechanism consists of a number of solid bodies connected in a chain so that the motions of one of the bodies is constrained by the others. These bodies, also known as links, are usually rigid. This kind of body does not suffer deformation under the action of external forces. The motion of the mechanism is enabled by the kinematic joints connecting two neighboring links.

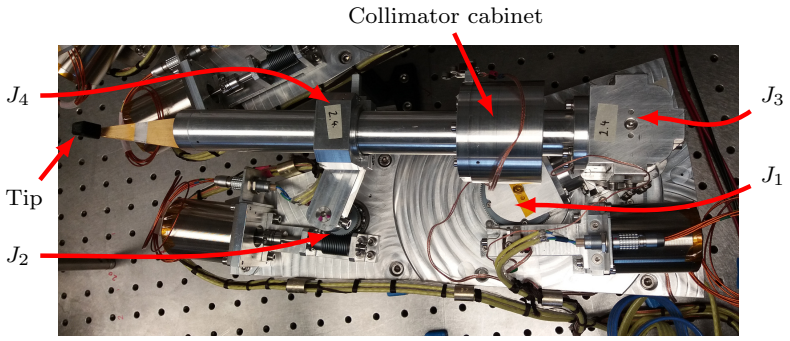
In literature, joints are frequently classified as lower pair joints if contact occurs over surfaces and as higher pair joints if contact occurs only at points or along lines. Lower pair joints are broadly used since provide good lubrication and wear is spread over the surface in contact. There are a total of six distinct types: revolute, prismatic, helical, cylindrical, spherical, and planar (Craig, 2005). Among those, the most popular lower pair joints are two possessing a single DoF. One of them is the *prismatic* joint, abbreviated as **P**, and the other is the *revolute* joint, abbreviated as **R**. In a connection where a **P** joint is involved, one link slides over the other resulting in relative translational motion between the two bodies, whereas with a revolute joint, one link has a turning or rotational motion relative to the other.

Mechanisms are often built using joints that exhibit just one DoF. However, in those case where a joint of n DoF is required, it can be modeled as a series of n distinct joints of one DoF each connected with *n-1* links of zero length. For instance, the more complex motion of a two DoF *cylindrical* joint can be obtained with the combination of a revolute and a sliding joint. This joint, abbreviated as **C**, provides a single-axis rotation as well as a single-axis translation. Typically, a cylindrical joint is implemented by means of a solid tube inserted in a larger hollow cylinder, where the former piece is allowed to freely rotate as well as to slide along the external one.

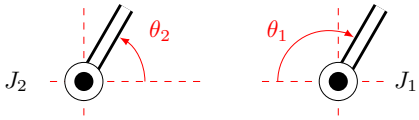
Mechanisms actuated by electrical motors and controlled by a computerized device are said to be robotic mechanisms. If they have a fixed base, then they are known explicitly as manipulators. This kind of mechanisms is commonly built around an arm ensuring some degree of mobility and an end-effector, attached to one end of that arm, performing the required task. Many of the specific-purpose manipulators seen in the manufacturing industry present a *serial* structure, also commonly referred to as an *open kinematic chain*. In serial robots, there is only one sequence of links connecting the two ends of the chain and the DoF of the whole structure is the sum of the DoF provided by each joint in the system. Lately, however, *closed kinematic chain* manipulators, which contain a sequence of links forming one or more loops, are gaining more and more attention. The DoF of these manipulators are inferior to the aggregated sum of the DoF of every joint due to the motion constraints imposed by the loops.

3.1.1 The Probe Arm of MIRADAS

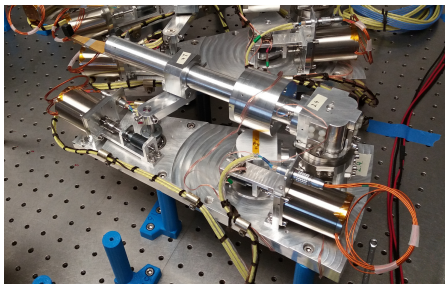
The probe arm, as shown in fig. 3.1a, is a two DoF closed kinematic chain. This mechanism, denoted by $\mathscr{A}$, is comprised of four joints (J_1 , J_2, J_3 and J_4) and three links, providing to the whole structure a planar motion in the (x, y) Cartesian space. These links are mainly hollow tubes containing optical components to relay a beam of light from the pick-off mirror -see fig. 3.1d- to the spectrograph. The location of any element of the arm, including the tip mirror, can be determined by the rotation angles, denoted by θ_1 and θ_2, of the revolute joints J_1 and J_2 respectively. Whereas the latter is unconstrained, being able to move in the interval $[0, 2\pi)$, the motion of the former is constrained to the range $[0, \pi]$. The prismatic motion of the cylindrical-like joint J_4, which freely slides over the arm longest tube, is constrained by the cylindrical cabinet containing the



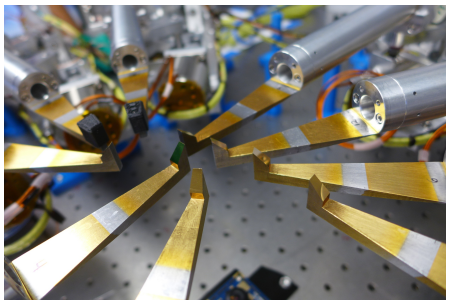
(a) An aerial view of a real MXS probe arm with electrical wiring and motors. The four joints of the mechanism are labeled. J_1, J_2 and J_3 are revolute joints, whereas J_4 slides and rotates over the tube where the collimator cabinet is present. A black protection cap is on the pick-off mirror to prevent scratches while the arm is not in operation.



(b) The rotation angle θ_2 of joint J_2 increments counterclockwise, while θ_1 increases clockwise.



(c) An MXS probe from a lateral view.



(d) Detail of several arms. Here, the pick-off mirrors of each arm can be appreciated.

Figure 3.1: The MIRADAS MXS probe arm.

collimator. Additionally, this joint rotates about an axis perpendicular to the plane where the arm moves. Although the arm has four joints, only two of them, J_1 and J_2, are actuated by stepper motors.

In fig. 3.2, we show the linkage model of the MXS probe arm, which will be extensively used in section 3.4 to derive the kinematic behavior of the mechanism. The position of all elements

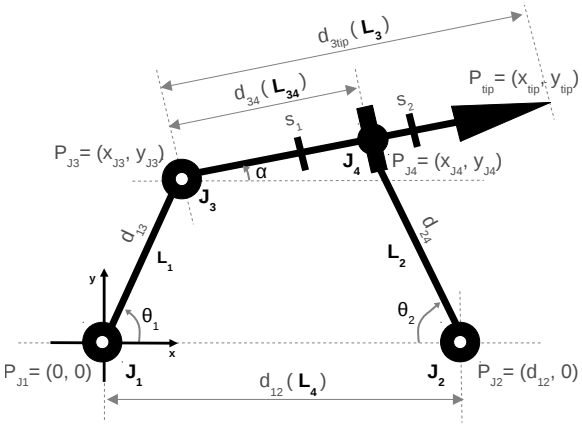


Figure 3.2: The probe arm model consists of several constrained linkages. It is a close-loop kinematic chain with four links (L_1, L_2, L_3 and L_4) and four joints (J_1, J_2, J_3 and J_4). The position of joint J_4 must always be between stop position s_1 (collimator cabinet) and s_2 (end of the long tube). The articular variables θ_1 and θ_2 denote the DoF of the mechanism. The origin of the coordinate frame of the arm is at joint J_1.

and lengths are described relative to a fixed coordinate frame attached at the center of joint J_1. As can be appreciated from the model, only those pairs of the independent variables (θ_1, θ_2) satisfying the following mathematical constraint are valid:

$$d_{s_1} < d_{34} < d_{s_2} \tag{3.1}$$

Equation (3.1) ensures that J_4 is always in the segment bounded by the stop points s_1 and s_2.

3.2 Configuration Space and Zone of Avoidance

Although the articulated mechanism $\mathscr{A}$ lives in a three-dimensional ambient space ($\mathscr{W}$), its control as well as its trajectories are determined in another world called the *configuration space*, *C-space* or simply $\mathscr{C}$ (Lozano-Pérez, 1983). Based on Lagrangian mechanics, since its introduction in the context of robotics, configuration space has been widely used to develop motion planning techniques. Its main advantage is that the volume occupied by the whole mechanism $\mathscr{A}$ in the $\mathscr{W}$ becomes a single point in $\mathscr{C}$. The dimension of *C-space* corresponds to the number of independent parameters or DoF of the system. Every single point $q_i \in \mathscr{C}$ is known as a configuration and consists of a particular combination of the parameters modeling the system. The volume in $\mathscr{W}$ occupied by the mechanism $\mathscr{A}$ when configured at $q \in \mathscr{C}$ is denoted by $\mathscr{A}(q)$ and comes determined by the kinematic behavior of the arm. Finally, even though the motions are planned in $\mathscr{C}$, since the arms actually move in $\mathscr{W}$, collisions need to be checked in this space. Therefore, every time

the motion planning algorithm wants to verify if two arms $\mathscr{A}_i$ and $\mathscr{A}_j$ located at configurations q_i and q_j collide against each other, translation from one space into the other is required. These transformations will be discussed later in section 3.4.

In physics, mathematics and control theory, C-space is considered as a differentiable manifold, leading to a few technical hurdles. Fortunately, in motion planning, no C-space calculus is required. Additionally, in this context, C-space can be defined as a topological manifold. In our case, an arm configuration is formed by a particular 2-tuple containing real numbers for the parameters θ_1 and θ_2. As seen in fig. 3.3, two different subsets of arm configurations can be distinguished: (i) configurations that are feasible and (ii) those which must be avoided. The latter group consists of the pairs leading to mechanically impossible configurations due to the violation of the constraint expressed in eq. (3.1). These configurations conform the *zone of avoidance* (ZoA), and during motion planning, those potential trajectories traversing this prohibited area have to be immediately discarded. Therefore, motion planning should only consider the feasible configuration space of the arm $\mathscr{C}_{feas} \subset \mathscr{C}$:

$$\mathscr{C}_{feas} = \{q \in \mathscr{C} \mid d_{s_1} < d_{34} < d_{s_2}\} \tag{3.2}$$

The variable d_{34} in terms of the joint variables can be expressed as the distance between points P_{J_4} and P_{J_3}; see fig. 3.2:

$$d_{34}(\theta_1,\theta_2) = d(P_{J_4},P_{J_3}) = \sqrt{(x_{J_4}-x_{J_3})^2+(y_{J_4}-y_{J_3})^2} \tag{3.3}$$

The locations of J_3 and J_4 only depend on the variables of the joints $J1$ and J_2:

$$x_{J_3} = d_{13}\cos\theta_1 \qquad y_{J_3} = d_{13}\sin\theta_1 \tag{3.4a}$$

$$x_{J_4} = d_{12} - d_{24}\cos\theta_2 \qquad y_{J_4} = d_{24}\sin\theta_2 \tag{3.4b}$$

Finally, substituting the expressions in (3.4) into (3.3), we find that:

$$d_{34}(\theta_1,\theta_2) = \sqrt{(d_{12}-d_{24}\cos\theta_2-d_{13}\cos\theta_1)^2+(d_{24}\sin\theta_2-d_{13}\sin\theta_1)^2} \tag{3.5}$$

3.2.1 Topology of the Arm *C-space*

Topology studies the properties of space that are immutable under arbitrary continuous transformations. This branch of mathematics is very helpful in understanding configuration spaces in general, and its concepts play a vital role in the development of motion algorithms. Manifolds are those topological spaces locally resembling a real n-dimensional Euclidean space (Lee, 2011).

Broadly speaking, the values an articular variable can hold are characterized by the topological space associated with the joint. In the vast majority of motion planning problems, the configuration space is a complex manifold resulting from the Cartesian product of each individual

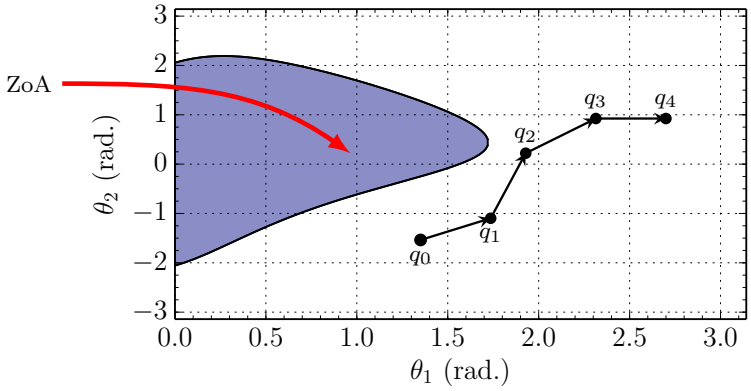


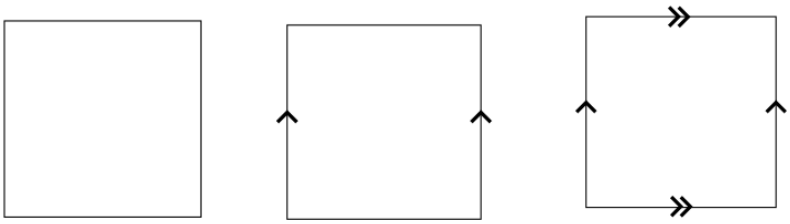
Figure 3.3: The arm $\mathscr{C}$ comprises the set of all (θ_1, θ_2) combinations. However, there is a subset of them, the ZoA, that is not feasible due to constraints in the mechanical design of the arm. The trajectories are determined in this control space as sequences of points q_i, each of which with an associated time profile.

topological space involved. In this context, and as an example, consider a robot moving in a 2D world. This robot is able to rotate counterclockwise by any $\theta \in [0, 2\pi)$ and then translate by any $x \in \mathbb{R}$[1] in the X direction and any $y \in \mathbb{R}$ in the Y direction. This allows for any possible position and orientation, and every x , y , θ combination leads to a unique robot placement. Although the set of all possible configurations $q = (x, y, \theta)$ clearly defines a subset of $\mathbb{R}^3$, when specifying the C-space, it must be taken into account that $\theta \pm 2\pi$ produces equivalent rotations. Then, the C-space $\mathscr{C}$ is a 3D manifold defined as follows: $\mathscr{C} = \mathbb{R}^2 \times \mathbb{S}^1$, where $\mathbb{S}^1$ accounts for θ and represents a circle in the topological sense. This means that when traveling in the X and Y directions, there is a frontier to the *C-space*. On the other hand, traveling in the Z direction causes a wraparound at 2π. If the robot were unable to rotate, then we would obtain the translation-only case and $\mathscr{C} = \mathbb{R}^2$. In fig. 3.4, we represent some 2D manifolds.

Taking into account the particularities of rotations, the unit circle $\mathbb{S}^1 = \{(x, y) \mid x^2 + y^2 = 1\}$ can be represented by embedding $\mathbb{S}^1$ in $\mathbb{R}^2$. Moreover, by employing identification, we can specify that some points of a space are identical, even though they originally were distinct. In the case of $\mathbb{S}^1 \in [0, 1]$ [2], it can be declared that 0 and 1 are equivalent, denoted as $0 \sim 1$. This identification has the effect of connecting the ends of the interval together, forming a closed loop. Similarly, the configuration space of a planar manipulator with two revolute joints can be represented by the torus $T^2 = \mathbb{S}^1 \times \mathbb{S}^1$; see fig. 3.5. This torus, as shown in fig. 3.4c, can be constructed by performing

[1] The space $\mathbb{R}$ represents the real numbers.

[2] We only allow the articular variable to run from 0 up to 1, conceptually representing the angular values 0 and 2π respectively.

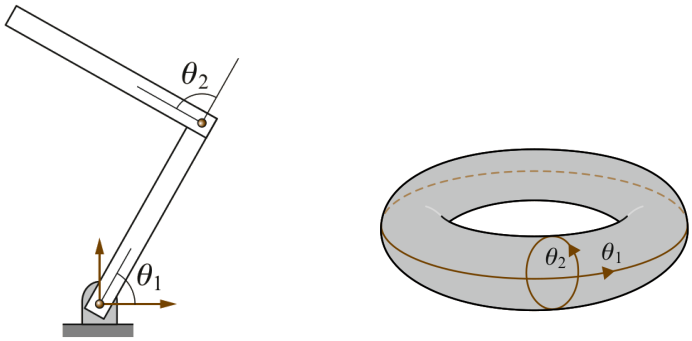


(a) Plane: $\mathbb{R}^2$.

(b) Cylinder: $\mathbb{R} \times \mathbb{S}^1$. The left and right sides are glued together.

(c) Torus: $T^2 = \mathbb{S}^1 \times \mathbb{S}^1$. The left and right sides are glued together as well as the top and down edges.

Figure 3.4: Some 2D manifolds obtained by identifying pairs of points along the boundary of a square region. *Image Credit:* LaValle (2006).



(a) A two-joint planar arm with no joint limits.

(b) A torus, the *C-Space* of the planar arm.

Figure 3.5: A two-joint planar arm and its configuration space. *Image Credit:* Siciliano and Khatib (2008).

identifications of the form $(0, y) \sim (1, y)$ and also $(x, 0) \sim (x, 1)$, identifying the top and bottom. The interested reader is referred to Choset et al. (2005) and LaValle (2006) for a comprehensive discussion about topological spaces in terms of motion planning.

As a general rule, we can derive the C-space of any other articulated mechanism taking into account that a prismatic joint can take any value of the one-dimensional manifold $\mathbb{R}$, whereas a revolute joint is denoted by the space $\mathbb{S}^1$. In our case, although the arm technically has two actuated revolute joints, since the range of motion of J_1 is limited to the finite interval $[0, \pi]$, it can be modeled more effectively by a finite $\mathbb{R}$ manifold. By combining the two manifolds, we obtain the topological space $\mathscr{C}$, which is a *cylinder* of limited height:

$$\mathscr{C} = \mathbb{R} \times \mathbb{S}^1 \tag{3.6}$$

where $\theta_1 \in [0, \pi]$, $\theta_2 \in [0, \pm 2n\pi)$ and $n \in \mathbb{N}$.

It is important to be aware of the particular topology of θ_2, mainly when calculating paths and distances between any pair of points in $\mathscr{C}$. Due to its loop nature, any point of the $\mathscr{C}_{feas}$ can be reached going upward or downward along the θ_2 axis. That is, by either rotating clockwise or rotating counterclockwise the joint J_2.

3.2.1.1 Metric Spaces

Sometimes, for convenience, a manifold is viewed as a metric space, where a distance function is defined between any pair of elements. The metrics of the topological spaces involved in the arm C-space are the following:

1. *Euclidean one-space*: it is used for rigid translations. The metric for two points $q, p \in \mathbb{R}$ is defined as:

$$dist_{\mathbb{R}}(q, p) = |q - p| \tag{3.7}$$

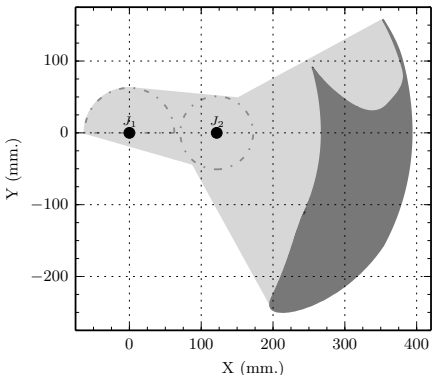
2. *Circle* $\mathbb{S}^1$: it can be represented by $\mathbb{S}^1 = [0, 1]/0 \sim 1$, a unit interval with identified end points. It arises from rigid rotations. The metric for two points $q, p \in \mathbb{S}^1$ is defined as:

$$dist_{\mathbb{S}^1}(q, p) = min(|q - p|, 1 - |q - p|) \tag{3.8}$$

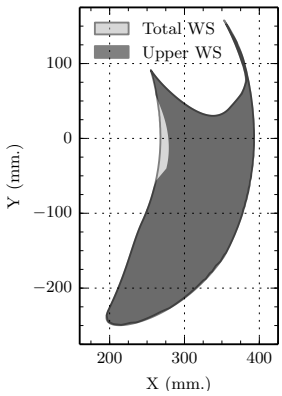
3.3 Workspace and Envelope

The *workspace* ($\mathscr{W}_s$) of the mechanism represents the portion of $\mathscr{W}$ that can be reached by the arm pick-off mirror. Thus, it determines which points of the instrument FOV can be patrolled by the arm. Its shape depends on the manipulator structure as well as on the presence of mechanical joint limits. In our case, the $\mathscr{W}_s$, shown in fig. 3.6a, can be found by substituting all points of $\mathscr{C}_{feas}$ into the forward kinematic equations; see section 3.4.1 for more details.

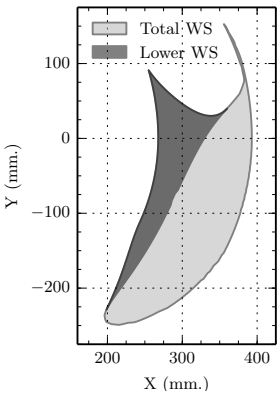
Although the $\mathscr{W}_s$ is frequently seen as an area, it can also be represented as a function of d_{34}, resulting in a cylindrical $\mathscr{W}_s$. This 3D figure can be easily computed by building a grid in $\mathscr{C}_{feas}$ and calculating for each point its corresponding d_{34} value with the help of eq. (3.5). In fig. 3.7a and in fig. 3.7b, two different views of this cylinder-like volume are shown. We can distinguish two distinct regions there, the upper ($\mathscr{W}_{s_{up}}$) and the lower one ($\mathscr{W}_{s_{down}}$), both seen in fig. 3.6b and in fig. 3.6c respectively. These zones are the result of individually projecting the two sides, the upper one and the lower one, of the 3D $\mathscr{W}_s$ on the (x, y) plane. As it can appreciated, $\mathscr{W}_{s_{up}}$ and $\mathscr{W}_{s_{down}}$ share a portion of the total $\mathscr{W}_s$. These common points can be reached by two different arm configurations, one belonging to the upper workspace and the other to the lower. That is because of the 360° range of motion of the joint J_2. Besides, we can also see that the most significant contribution to the two-dimensional $\mathscr{W}_s$ comes from $\mathscr{W}_{s_{up}}$.



(a) The arm *workspace* (in dark grey) and *envelope* (in light grey), both calculated according to the frame in J_1. The locations of J_1 and J_2 and their range of motion, 180° and 360° respectively, are also represented.



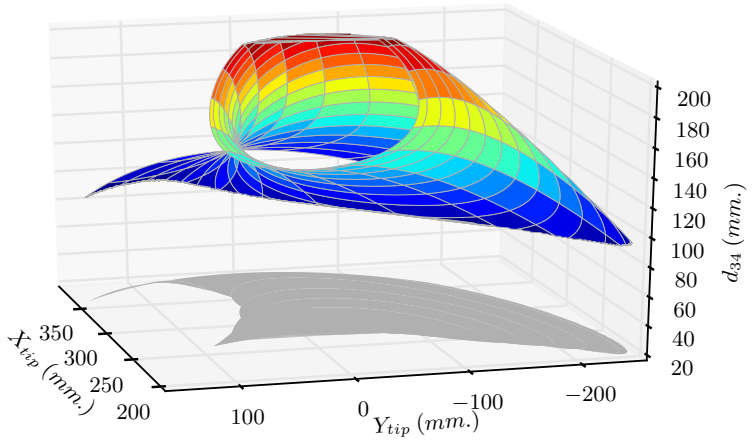
(b) The *upper workspace* appears superposed over the arm workspace. The upper workspace is the result of translating the configurations belonging to the upper side of the cylinder shown in fig. 3.7b into the Cartesian space.



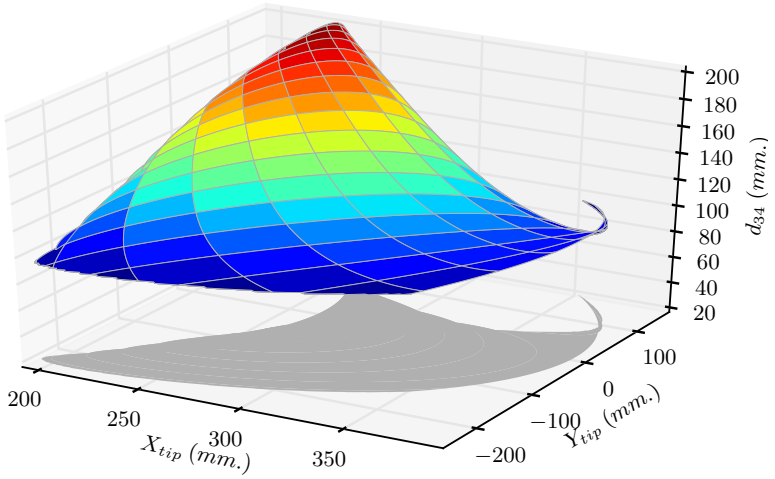
(c) The *lower workspace* appears superposed over the arm workspace. The lower workspace is the result of translating the configurations belonging to the lower side of the cylinder shown in fig. 3.7b into the Cartesian space.

Figure 3.6: The probe arm workspace and envelope.

The *envelope* ($\mathscr{E}_{arm}$) is the area in the Cartesian plane occupied by all the bodies of the arm when it executes all possible configurations (Siciliano and Khatib, 2008). The envelope encloses the arm workspace and can been seen in fig. 3.6a.



(a) The 3D workspace. The cylinder-shaped volume shows that some points of the workspace can be reached by two different values of d_{34}. As this variable represents the position of joint J_4, it means that each of those points can be reached by two different (θ_1, θ_2) pairs.



(b) A different view of the 3D workspace. The *upper workspace* is the projection of the upper part of the cylinder-shaped volume into the (x, y) plane, while the projection of the lower part is known as the *lower workspace*.

Figure 3.7: The probe arm 3D workspace.

3.4 Kinematic Behavior

In science, kinematics studies motion without considering the forces that cause it. It frequently involves analyzing the position of the bodies as well as its first and second order time derivatives. Therefore, in the context of mechanisms, kinematics refers to the geometrical aspects of the elements, which mainly includes its position in the world they live. Additionally, in those cases where they are needed, velocities and accelerations of the bodies are also studied.

As we have seen previously, mechanisms, in general, are controlled in a world different from the one they move. The kinematics expressions provide a way to connect both worlds. The transformation of one point from C-space into one in $\mathcal{W}_s$ and vice versa is achieved by what is known as *forward kinematics* and *inverse kinematics* respectively. A popular way to address these problems is by applying the Denavit and Hartenberg (DH) convention for systematically selecting frames of references and using a combination of relative geometric transformations (Hartenberg and Danavit, 1964). This approach unambiguously describes the positions of the robot links and joints as well as the position and orientation of the end-effector according to the robot base frame. In standard serial robots, the use of DH is straightforward; however, in mechanisms containing links connected so that loops are formed (closed kinematic chains) things become more complicated.

In the following sections, we present an analytic solution for forward kinematics, whereas a geometric solution is found for inverse kinematics.

3.4.1 Forward Kinematics

In this section, we formulate an analytical solution for the forward problem. The equations derived here express the position of the manipulator's end-effector, the arm pick-off mirror, as a function of the joint variables with respect to the coordinate frame attached to the arm base. According to the linkage model presented in fig. 3.2 the position (x_{tip}, y_{tip}) of the pick-off mirror can be obtained as follows:

$$x_{tip}(\theta_1,\theta_2) = d_{13}\cos\theta_1 + d_{3tip}\cos\alpha(\theta_1,\theta_2) \tag{3.9a}$$

$$y_{tip}(\theta_1,\theta_2) = d_{13}\sin\theta_1 + d_{3tip}\sin\alpha(\theta_1,\theta_2) \tag{3.9b}$$

where $(\theta_1,\theta_2) \in \mathscr{C}_{feas}$ and the angle α is defined in terms of θ_1 and θ_2.

The value of α can be calculated from the loop-closure expression:

$$\vec{L}_1 + \vec{L}_{34} + \vec{L}_2 + \vec{L}_4 = 0 \tag{3.10}$$

Expanding the vectorial expression in eq. (3.10) into its x and y components:

$$d_{13}\cos\theta_1 + d_{34}\cos\alpha + d_{24}\cos\theta_2 - d_{12} = 0 \tag{3.11a}$$

$$d_{13}\sin\theta_1 + d_{34}\sin\alpha - d_{24}\sin\theta_2 = 0 \tag{3.11b}$$

and combining eq. (3.11a) and eq. (3.11b), we find that:

$$\alpha(\theta_1,\theta_2) = \arctan\left(\frac{-d_{13}\sin\theta_1 + d_{24}\sin\theta_2}{d_{12} - d_{13}\cos\theta_1 - d_{24}\cos\theta_2}\right) \tag{3.12}$$

Finally, using trigonometric equivalent expressions, the position of the end effector can be expressed as:

$$x_{tip}(\theta_1,\theta_2) = d_{13}\cos\theta_1 + d_{3tip}\frac{1}{\sqrt{1+u(\theta_1,\theta_2)^2}} \tag{3.13a}$$

$$y_{tip}(\theta_1,\theta_2) == d_{13}\sin\theta_1 + d_{3tip}\frac{u(\theta_1,\theta_2)}{\sqrt{1+u(\theta_1,\theta_2)^2}} \tag{3.13b}$$

where u is:

$$u(\theta_1,\theta_2) = \frac{-d_{13}\sin\theta_1 + d_{24}\sin\theta_2}{d_{12} - d_{13}\cos\theta_1 - d_{24}\cos\theta_2} \tag{3.14}$$

In those cases where the location of joint J_3 needs to be determined, the value of α can be efficiently computed from the two-argument function *atan2*, which is available in the standard libraries or packages of most popular programming languages. This function is preferred over the one-argument arctan because it distinguishes between diametrically opposite directions:

$$\alpha = atan2(\sin\alpha, \cos\alpha) \tag{3.15}$$

where $\cos\alpha$ and $\sin\alpha$ can be found applying the loop-closure expressions in eq. (3.11):

$$\cos\alpha = \frac{d_{12} - d_{24}\cos\theta_2 - d_{13}\cos\theta_1}{d_{34}} \tag{3.16a}$$

$$\sin\alpha = \frac{d_{24}\sin\theta_2 - d_{13}\sin\theta_1}{d_{34}} \tag{3.16b}$$

and the value of d_{34} can be obtained from eq. (3.5) on page 33.

3.4.2 Inverse Kinematics

Inverse kinematics determines the values of the articular variables θ_1 and θ_2 positioning the end-effector in a given point of the arm workspace. Since the forward equations are non-linear functions of the independent parameters, obtaining an analytical solution here is a difficult task. Thus, we present a geometric solution based on intersections between circles and lines.

According to the model in fig. 3.2, any arm configuration has to satisfy two conditions: (i) the distance between joints J_1 and J_3 must be d_{13} and (ii) the distance between J_3 and tip must be d_{3tip}. Hence, all mechanically feasible positions of J_3 are obtained by intersecting two circles: C_1 and C_{tip} (see fig. 3.8). The former, having a radius of length d_{13}, is centered at J_1, whereas the latter is centered at the desired end-effector position ($\vec{P}_{tip}$) and has a radius of length d_{3tip}.

To geometrically determine the intersecting points between both circles, first of all, the distance between both centers is calculated as follows:

$$d_{1tip} = |P_{J_{tip}} - P_{J_1}| \tag{3.17}$$

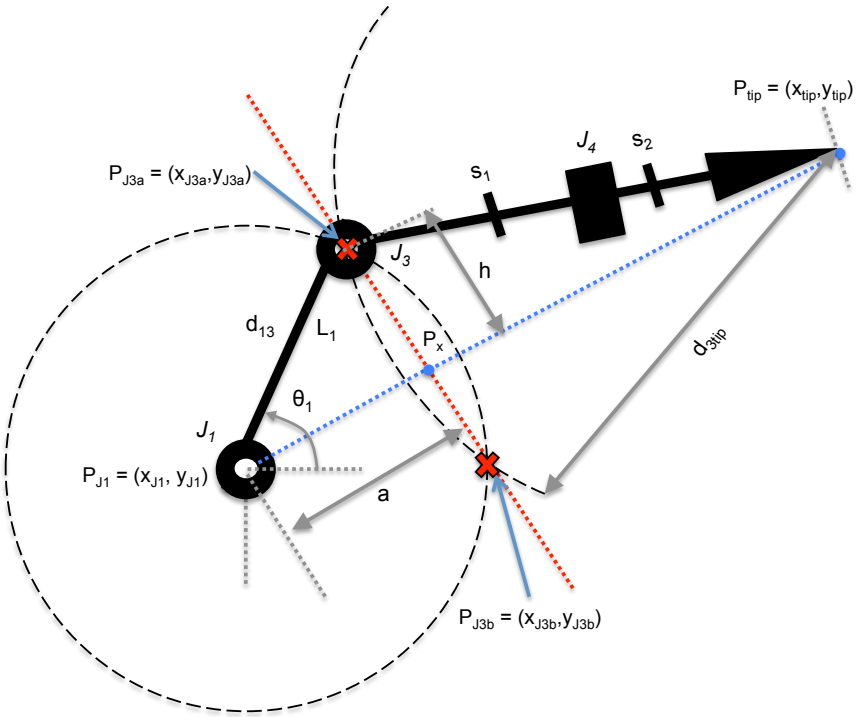


Figure 3.8: For a given pick-off mirror final placement, the position of J_3 is obtained by intersecting the two dashed circles. Then, once known the position of J_3 and because J_1 is fixed at a known point of the MXS bench, we can easily find the angle θ_1.

Then, as the values of d_{1tip} and the radius of C_1 and C_{tip} are known beforehand, we can use the Pythagorean theorem twice to obtain the distances a and h represented in fig. 3.8:

$$d_{13}^2 = a^2 + h^2 \tag{3.18}$$

$$d_{3tip}^2 = (d_{1tip} - a)^2 + h^2 \tag{3.19}$$

We continue by computing $\vec{P}_x$, the point of intersection of the red dashed line[3] and the blue one[4] in fig. 3.8, as follows:

$$\vec{P}_x = a\,\frac{(\vec{P}_{tip} - \vec{P}_{J_1})}{d_{1tip}} + \vec{P}_{J_1} \tag{3.20}$$

where $\frac{(\vec{P}_{tip} - \vec{P}_{J_1})}{d_{1tip}}$ is a unit vector having the same direction as the blue straight line.

[3] The straight line passing through $\vec{P}_{J_{3a}}$ and $\vec{P}_{J_{3b}}$.
[4] The straight line connecting $\vec{P}_{J_1}$ and $\vec{P}_{J_{tip}}$

And once known the value of $\vec{P}_x$, the two possible intersecting points between C_1 and C_{tip}, $\vec{P}_{J_3a}$ and $\vec{P}_{J_3b}$ respectively, are determined applying the expression:

$$\vec{P}_{J_3a,J_3b} = \vec{P}_x \pm h\frac{(\vec{P}_{tip} - \vec{P}_{J_1})}{d_{1tip}} \times \hat{k} \tag{3.21}$$

where $\hat{k}$ is the unit vector pointing in the direction of the Z-axis in the Cartesian base $(\hat{i}, \hat{j}, \hat{k})$.

These two points indicate the two mathematical possible placements for joint J_3. Now, making use of the location of joint J_1, which is known, we can obtain the two values for the variable θ_1. They come determined by the angles between the lines connecting the intersecting points and $\vec{P}_{J_1}$. But due to the limited range of motion of J_1, both solutions found may not fulfill the mechanical constraints of the arm. Hence, if any value of θ_1 is not in the $[0, \pi]$ interval, then, it should be discarded.

Reached this point, we proceed with the determination of the position of J_4. In the first step of this process, we draw a straight line passing through the previously calculated position of J_3 and the desired end-effector position. This line will symbolize the link L_3 of the arm model. Then, we additionally draw a circle (C_2) with center at J_2 $(\vec{P}_{J_2})$ and radius equal to d_{24}. The two possible intersecting points between L_3 and C_2 represent the two possible positions of J_4. Mechanically, J_4 must be located in the segment bounded by the stop points s_1 and s_2. Thus, as in the case of J_3, only positions satisfying the constraint should be accepted.

In order to find the placement of J_4 $(\vec{P}_{J_4})$, we follow a geometric approach similar to the one previously used in the resolution of J_3; see fig. 3.9. Initially, we found the distance b by projecting the vector $(\vec{P}_{J_2} - \vec{P}_{J_{tip}})$ over a unit vector $(\hat{P}_{L_3})$ in the direction of L_3. The vector $\hat{P}_{L_3}$ can be expressed as follows:

$$\hat{P}_{L_3} = \frac{\vec{P}_{J_3} - \vec{P}_{J_{tip}}}{|\vec{P}_{J_3} - \vec{P}_{J_{tip}}|} \tag{3.22}$$

In Euclidean geometry, the projection of one vector over other can be obtained by applying the dot product. Therefore, b is:

$$b = (\vec{P}_{J_2} - \vec{P}_{J_{tip}}) \cdot \hat{P}_{L_3} \tag{3.23}$$

Then, $\vec{P}_p$, as given in fig. 3.9, can be found using the following vector addition:

$$\vec{P}_p = b\hat{P}_{L_3} + \vec{P}_{tip} \tag{3.24}$$

And the two crossing points between L_3 and C_2, denoted by $\vec{P}_{J_4a}$ and $\vec{P}_{J_4b}$, can be calculated by means of the following expression:

$$\vec{P}_{J_4a,J_4b} = \vec{P}_p \pm w\frac{(\vec{P}_p - \vec{P}_{J_2})}{v} \times \hat{k} \tag{3.25}$$

where v and w -see fig. 3.9- are:

$$v = |\vec{P}_p - \vec{P}_{J_2}| \tag{3.26}$$

$$w^2 = d_{24}^2 - v^2 \tag{3.27}$$

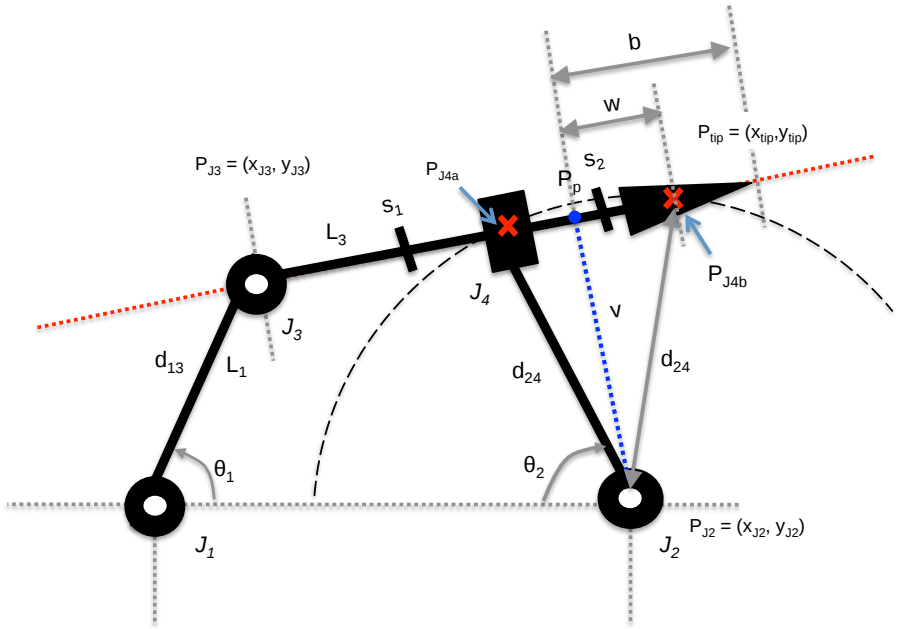


Figure 3.9: Once determined the position of J_3, we draw a line from there to the tip. The intersections between this line and a circle centered at J_2 give the two theoretical positions of J_4.

Again, we obtain two different values for the position of J_4. Since the location of J_2 is known beforehand[5], we can straightforwardly find the values for θ_2 just as we did with θ_1.

3.4.3 Differential Kinematics

Differential or instantaneous kinematics study local changes or local changes per time. Basically, it solves the problem of finding a relationship between the manipulator articular velocities $\dot{q}$ and the corresponding end-effector linear velocity $\dot{p}$ and angular velocity ω. In this mapping task between both spaces, the *geometrical Jacobian* matrix plays an important role.

A common approach is employing the *analytical Jacobian*, which maps velocities in joint space to time-derivatives of the end-effector configuration. As its name indicates, this Jacobian is analytically obtained through differentiating the forward kinematics equations with respect to the joint variables. The geometrical and analytical Jacobians generally differ since the time-derivative of the end-effector orientation does not usually correspond to the angular velocity. However, for those manipulators whose particular geometry enables the determination of the

[5]Like J_1, the position of J_2 comes determined by its physical placement in the MXS bench.

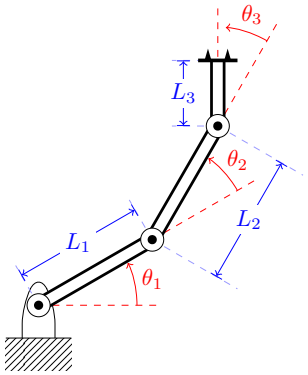


Figure 3.10: A three-link planar arm containing 3 revolute joints, characterized by the angles θ_1, θ_2 and θ_3 respectively.

end-effector pose with reference to a minimal representation of the ambient space, both Jacobians are essentially the same (Siciliano et al., 2009). In fact, that is the case of a manipulator whose DoFs cause rotations of the end-effector all about the same fixed axis in space. Particularly, for the three-link manipulator in fig. 3.10, which shows a planar motion only permitting rotations about the Z-axis, there is no difference between both matrices.

Jacobians are especially valuable when it comes to the characterization of any manipulator. For instance, they are indispensable for finding singularities, analyzing redundancy as well as for the control of the mechanism (Craig, 2005). They are useful if, due to the requirements of the task to accomplish, the end-effector should move at a given desired speed.

3.4.3.1 Geometrical Jacobian

Consider an n-DoF manipulator where the joint velocity $\dot{q}$ is a $n \times 1$ vector

$$\dot{q} = (\dot{q_1},\ \dot{q_2},\ \dot{q_3}\ \dots\ \dot{q_n})^T \tag{3.28}$$

Then, in the general case, the 6×1 vector $\dot{X}$ represents the end-effector velocities

$$\dot{X} = \begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} \dot{p} \\ \omega \end{pmatrix} = (\dot{x},\ \dot{y},\ \dot{z},\ \omega_x,\ \omega_y,\ \omega_z)^T \tag{3.29}$$

In compact form, eq. (3.29) can be written as

$$\dot{X} = J(q)\ \dot{q} \tag{3.30}$$

Equation (3.30) is the differential kinematics expression and the $6 \times n$ matrix J is the manipulator geometrical Jacobian, which can be decomposed in two different components

$$J(q) = \begin{pmatrix} J_d \\ J_o \end{pmatrix} \tag{3.31}$$

where

$$\dot{p} = J_d(q)\,\dot{q} \tag{3.32}$$

$$\omega = J_o(q)\,\dot{q} \tag{3.33}$$

The $3 \times n$ matrices J_d and J_o in eq. (3.32) and eq. (3.33) are respectively called the *displacement Jacobian* and the *orientation Jacobian*. The former expresses the contribution of joint velocities to the end-effector linear velocity, whereas the latter relates the contribution of joint velocities to the angular velocity. In practice, for a particular manipulator, these Jacobian matrices are directly computed at the differential level following a geometric approach (Siciliano et al., 2009).

3.4.3.2 Analytical Jacobian

Let us consider the function $\mathscr{F}$ denoting the forward kinematics expression of a manipulator, mapping the C-space defined by the variable q to the Cartesian space defined by X

$$X = \mathscr{F}(q) \tag{3.34}$$

Expanding eq. (3.34), it can be rewritten as

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} f_1(q) \\ f_2(q) \\ \vdots \\ f_n(q) \end{pmatrix} \tag{3.35}$$

Now, consider a manipulator at a particular configuration q. Then, each of its joints undertakes an infinitesimally small displacement represented by the vector δq, which results in a displacement of its end-effector position and orientation denoted by the vector δx. If we use the chain rule, the following set of expressions is obtained

$$\begin{aligned} \delta x_1 &= \frac{\partial f_1}{\partial q_1}\delta q_1 + \frac{\partial f_1}{\partial q_2}\delta q_2 + \ldots + \frac{\partial f_1}{\partial q_n}\delta q_n \\ \delta x_2 &= \frac{\partial f_2}{\partial q_1}\delta q_1 + \frac{\partial f_2}{\partial q_2}\delta q_2 + \ldots + \frac{\partial f_2}{\partial q_n}\delta q_n \\ &\vdots \\ \delta x_n &= \frac{\partial f_n}{\partial q_1}\delta q_1 + \frac{\partial f_n}{\partial q_2}\delta q_2 + \ldots + \frac{\partial f_n}{\partial q_n}\delta q_n \end{aligned} \tag{3.36}$$

which can be written more simply in vector notation as follows:

$$\delta X = \begin{pmatrix} \frac{\partial f_1}{\partial q_1} & \frac{\partial f_1}{\partial q_2} & \ldots & \frac{\partial f_1}{\partial q_n} \\ \frac{\partial f_2}{\partial q_1} & \frac{\partial f_2}{\partial q_2} & \ldots & \frac{\partial f_2}{\partial q_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial q_1} & \frac{\partial f_n}{\partial q_2} & \ldots & \frac{\partial f_n}{\partial q_n} \end{pmatrix} \delta q \tag{3.37}$$

The above matrix of partial derivatives is called the analytical Jacobian, which is a function of the position of the manipulator expressed in terms of q. Using a compact notation, we can write the following expression

$$\delta X = J_a(q)\,\delta q \tag{3.38}$$

By differentiating both sides of eq. (3.38) with respect to time, then the analytical Jacobian relates the mechanism velocities in the joint domain to the-end effector velocities in the Cartesian space

$$\dot{X} = J_a(q)\,\dot{q} \tag{3.39}$$

3.4.3.3 MXS Probe Arm Jacobian

The Jacobian to be determined is the matrix J_θ relating the pick-off mirror linear velocity to the velocities of the variables of the joint space

$$\begin{pmatrix} \dot{x_{tip}} \\ \dot{y_{tip}} \end{pmatrix} = J_\theta \begin{pmatrix} \dot{\theta_1} \\ \dot{\theta_2} \end{pmatrix} \tag{3.40}$$

We start by differentiating the forward kinematics expressions found in eq. (3.13a) and eq. (3.13b) with respect to θ_1 and θ_2

$$\dot{x}_{tip} = d_{13}\, s_{\theta_1}\, \dot{\theta}_1 \; - \; \frac{d_{3tip}\, u}{(1+u^2)^{3/2}}\, \dot{u} \tag{3.41a}$$

$$\dot{y}_{tip} = d_{13}\, c_{\theta_1}\, \dot{\theta}_1 \; + \; \frac{d_{3tip}}{(1+u^2)^{3/2}}\, \dot{u} \tag{3.41b}$$

where $c_{\theta_i} \coloneqq \cos\theta_1$, $s_{\theta_i} \coloneqq \sin\theta_i$ and $c_{\theta_i\theta_j} \coloneqq \cos(\theta_i + \theta_j)$. These abbreviations will also be extensively employed in the following equations.

Since $u = f(\theta_1, \theta_2)$, we can obtain $\dot{u}$ as follows

$$\dot{u} = \frac{\partial f}{\partial \theta_1}\dot{\theta}_1 \; + \; \frac{\partial f}{\partial \theta_2}\dot{\theta}_2 \tag{3.42}$$

and using the above expression, eq. (3.41a) and eq. (3.41b) can be written as

$$\dot{x}_{tip} = d_{13}\, s_{\theta_1}\, \dot{\theta}_1 \; - \; \frac{d_{3tip}\, u}{(1+u^2)^{3/2}} \left(\frac{\partial f}{\partial \theta_1}\dot{\theta}_1 + \frac{\partial f}{\partial \theta_2}\dot{\theta}_2 \right) \tag{3.43a}$$

$$\dot{y}_{tip} = d_{13}\, c_{\theta_1}\, \dot{\theta}_1 \; + \; \frac{d_{3tip}}{(1+u^2)^{3/2}} \left(\frac{\partial f}{\partial \theta_1}\dot{\theta}_1 + \frac{\partial f}{\partial \theta_2}\dot{\theta}_2 \right) \tag{3.43b}$$

Finally, we obtain J_θ

$$J_\theta = \begin{pmatrix} -d_{13}s_{\theta_1} - \frac{d_{3tip}u}{(1+u^2)^{3/2}}\frac{\partial f}{\partial \theta_1} & -\frac{d_{3tip}u}{(1+u^2)^{3/2}}\frac{\partial f}{\partial \theta_2} \\ d_{13}c_{\theta_1} - \frac{d_{3tip}}{(1+u^2)^{3/2}}\frac{\partial f}{\partial \theta_1} & \frac{d_{3tip}}{(1+u^2)^{3/2}}\frac{\partial f}{\partial \theta_2} \end{pmatrix} \tag{3.44}$$

where

$$\frac{\partial f}{\partial \theta_1} = \frac{d_{13}^2 + d_{13}d_{24}c_{\theta_1\theta_2} - d_{13}d_{12}c_{\theta_1}}{(d_{12} - d_{13}c_{\theta_1} - d_{24}c_{\theta_2})^2} \tag{3.45}$$

$$\frac{\partial f}{\partial \theta_2} = \frac{-d_{24}^2 - d_{13}d_{24}c_{\theta_1\theta_2} + d_{24}d_{12}c_{\theta_2}}{(d_{12} - d_{13}c_{\theta_1} - d_{24}c_{\theta_2})^2} \tag{3.46}$$

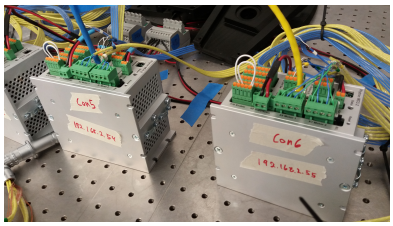
3.5 Controlling the Arm

The motion of the arm is controlled by the rotation of joints J_1 and J_2, each actuated by a different Phytron cryogenic stepper motor. Figure 3.11b shows part of the one driving joint J_1. The commercial off-the-shelf (COTS) motors manufactured by Phytron are a common choice among designers of cryogenic instruments due to their reliability and durability in such a harsh environment. Both steppers are wired to the same industrial-grade Phytron *MCC2* ethernet dual-axis motor controller, which can be appreciated in fig. 3.11a. Additionally, a number of switches are conveniently placed in several parts of the arm. Joint J_1 has two limit switches and a datum switch (see fig. 3.11b and fig. 3.11c), whereas J_2, whose motion is not mechanically constrained, has only a datum switch (see fig. 3.11d). There is also a limit switch attached to the collimator cabinet to detect the end of joint J_4 travel. With the help of these switches, a calibration procedure can determine the real range of motion of the arm, including its associated ZoA.

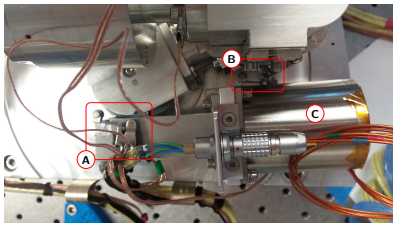
3.5.1 Generation of Motions

Although there is a whole chapter devoted to motion planning, there are few fundamental issues, while not warranting a section of their own, are worth mention here. The following aspects should be kept in mind when reading the rest of this work.

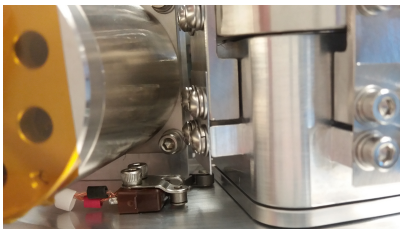
An arm path is defined as an ordered sequence of configuration space points connecting the starting and goal locations without regarding timing. However, a trajectory indicates the way an arm is moved in a controlled manner. In other words, in a trajectory, we consider when each part of the path is obtained. That is, a trajectory specifies a parametrization of a geometrical path in terms of time. The output of typical robot motion planners is a sequence of points linking one location to another through a series of intermediate points (way-points) where every two points are considered to be connected with a continuous straight line segment. Such is the case in MIRADAS, as can be appreciated in fig. 3.3. Unfortunately, that output is not realistic because it is not differentiable at the way-points. As the robots (the probe arm in our case) have non zero mass, and the forces that can be applied are finite, the mechanisms cannot instantaneously change its direction of motion. Therefore, a common approach in robot's control is approximate (or interpolate) the solution provided by the motion planners by a class of polynomial functions,



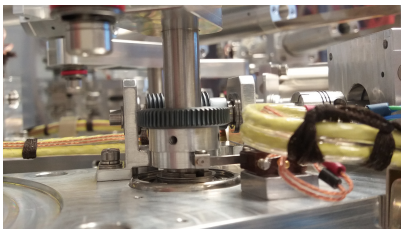
(a) Two Phytron MCC2 ethernet controllers wired, each controlling the two actuated joints of a MXS arm.



(b) The joint J_1 with its two limit switches: (A) front and (B) back. The cryogenic motor (C) actuating the joint.



(c) Detail of J_1 back switch and part of the cryogenic step motor.



(d) Gear wheel of joint J_2 and the datum switch of the joint.

Figure 3.11: Motor controller and different switches of the MIRADAS MXS probe arm.

making this way the whole path differentiable (Craig, 2005). Smooth trajectories are preferred since they reduce wear on the mechanism, unwanted vibrations as well as errors in path tracking.

Much work has been done in the field of smooth motion generations (Bobrow et al., 1985; Chand and Doty, 1985; Constantinescu and Croft, 2000; Jeon and Ha, 2000). Splines are often used, while, due to their tendency to oscillate, higher order polynomials are not considered to avoid retrograde motions. An alternative approach is to apply different functions in different segments of the path. This method is generally adopted in practice as it increases the flexibility of the overall function, and the trajectory may be adapted to different constraints.

A fast and suitable approach for industrial applications is the generation of trapezoidal trajectories, which use linear segments with parabolic blends (LSPB). In this technique, if only two points describe the geometric path, the resulting trajectory is composed of three phases: a straight line connected to the starting and goal positions by two different parabolic segments (Craig, 2005). During the blend portions of the trajectory, in the starting and arrival phases, constant acceleration is applied to change velocity smoothly, whereas in the linear phase a constant cruise velocity is used. The three phases are fitted together in such a way that the resulting path is continuous in position and velocity. Figure 3.12 shows a simple path built following the LSPB technique. As can be appreciated, it has been assumed that both blends use

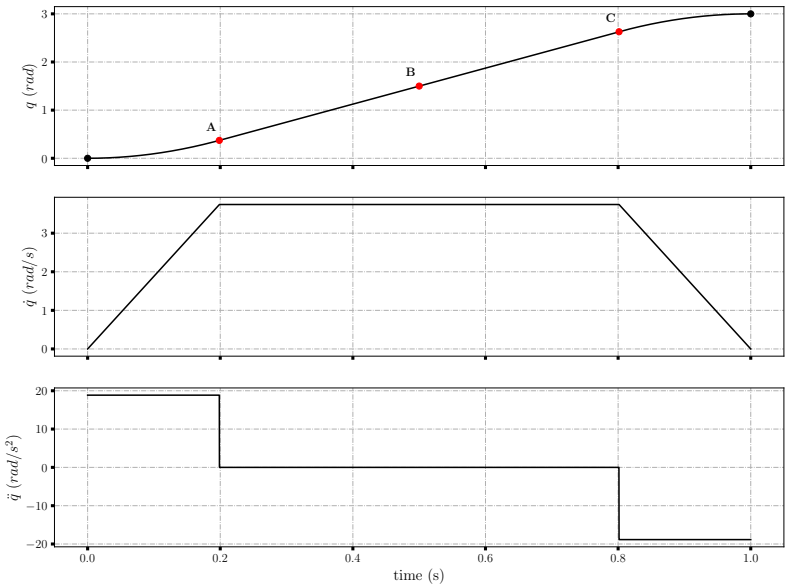


Figure 3.12: Time history of the position, velocity, and acceleration with a symmetrical trapezoidal trajectory involving only the initial and goal point. Label (A) indicates the final point of the first phase, the initial blend, whereas label (C) shows the point where the final blend start. Finally, label (B) shows the point of symmetry of the resulting trajectory.

the same acceleration magnitude. But, the first and the last parts might be handled slightly different considering different blend features. In some contexts, the initial and final blends are known as *ramp-up* and *ramp-down* since, respectively, velocity increases and decreases linearly.

3.5.1.1 Parabolic Blends through a Sequence of Points

As mentioned before, a path connecting an initial and a goal point often involves passing through several intermediate points. In industrial environments, these paths are frequently approximated using interpolation, especially multisegment linear polynomials with parabolic blends. However, this method slightly differs from the one applied when a path consists of only two points (see fig. 3.12). Linear functions keep connecting the intermediate points, but, now the blend regions used around each intermediate point, as seen in fig. 3.13, do not pass through the way-points. If we want our mechanism to follow the path precisely, it would have to come to a complete stop at each of these intermediate points, leading then, to slow executions as well as excess wear in the joints. One standard method of avoiding a complete stop while passing through the intermediate

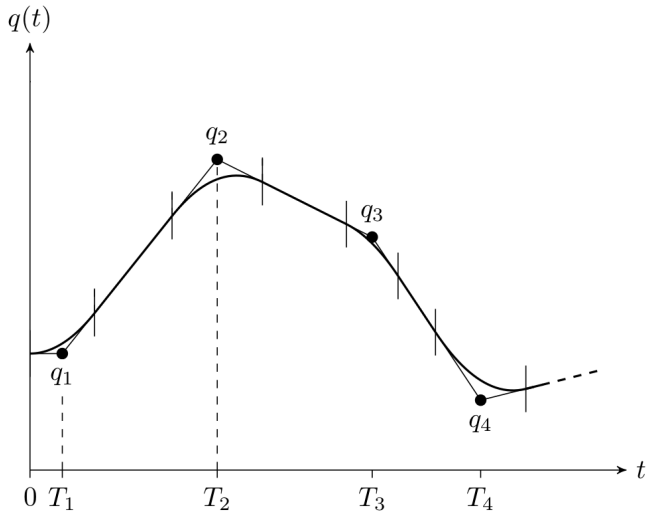


Figure 3.13: Trajectory obtained by interpolating a sequence of points using linear polynomials with parabolic blends.

points is to rely on two pseudo waypoints, one on each side of the original intermediate point (Craig, 2005). An alternative is to start generating the second segment ahead of time with respect to the end of the first segment; however, passing through the waypoint at the desired time is not guaranteed (Siciliano et al., 2009).

3.5.1.2 MCC2 and Intermediate Points

The Phytron $MCC2$ controller was conceived for simple industrial applications where no elaborated trajectories were required. As a result, it does not support interpolation where one or more waypoints in a path are specified (Phytron, 2010, 2012). Specifically, every trajectory comes defined only by the cruise velocity, the starting and the final position, automatically applying the controller parabolic blends around those points to smooth the motion. Thus, the only way to execute trajectories as delivered by motion planning algorithms is to stop every time an intermediate point is reached. Considering this *start-stop* behavior and, in order to extend the life of the motors as well as to decrease joints wear, the MIRADAS motion planner should generate paths with as few as possible intermediate points. In other words, a single linear segment connecting the starting and final arm configurations should be the preferred choice. Only in those cases where a linear path is not safe to follow, more complex paths should be gradually explored. In addition, this controller is not able to buffer or store incoming commands for future execution.

If the device receives a new motion command while it is executing one, the current is suddenly aborted, and the one that has just arrived is executed. Therefore, it is clear that, by all means, aborting commands should be avoided since the arms might be left in inconvenient locations.

3.6 Converting from FOV Frame into Arm Frame

MIRADAS has a total of 12 probe arms distributed around a circular bench. Each of these mechanisms observes a different user-defined sky object. These objects are transformed, as we will see in section 4.1, from celestial coordinates to positions in the instrument FOV. In this section, it is shown how coordinates specified in the FOV frame, whose origin is at the center of the MXS plate, can be expressed in terms of the probe arm coordinate system. This transformation is vital if we want to know if a given science target can be reached by a particular arm in the MXS system.

Using the notation of Siciliano and Khatib (2008), let O_i be the origin of a coordinate reference frame i and $(\hat{x}_i, \hat{y}_i, \hat{z}_i)$ the orthogonal basis of this system. The symbol ${}^{i}\mathbf{r}_j$ denote the vector $\mathbf{r}_j$ expressed relative to the frame i. Then, the position of O_{fov} relative to the frame arm_i can be described by the following $3x1$ vector:

$$ {}^{arm_i}\mathbf{O}_{fov} = \begin{pmatrix} {}^{arm_i}\mathbf{O}^{x}_{fov} \\ {}^{arm_i}\mathbf{O}^{y}_{fov} \\ {}^{arm_i}\mathbf{O}^{z}_{fov} \end{pmatrix} \tag{3.47} $$

where the components are the projections of the vector ${}^{arm_i}\mathbf{O}_{fov}$ onto the corresponding axes.

A vector ${}^{fov}\mathbf{r}$ can be transformed into ${}^{arm_i}\mathbf{r}$ as follows:

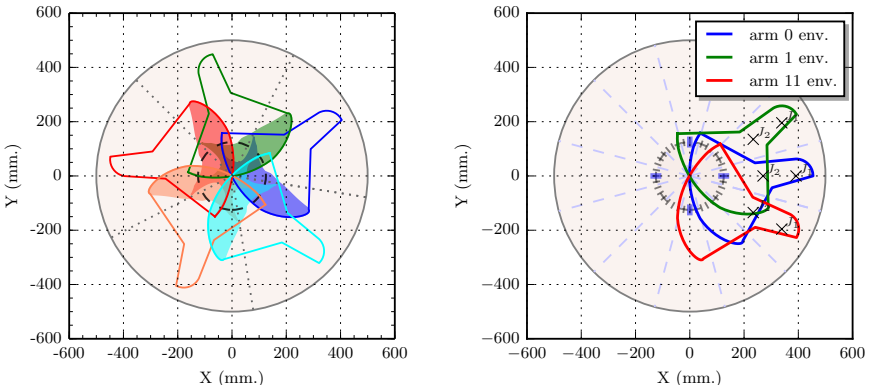
$$ {}^{arm_i}\mathbf{r} = \begin{pmatrix} {}^{arm_i}\mathbf{r}^{x} \\ {}^{arm_i}\mathbf{r}^{y} \\ {}^{arm_i}\mathbf{r}^{z} \end{pmatrix} = {}^{arm_i}\mathbf{R}_{fov}\,{}^{fov}\mathbf{r} + {}^{arm_i}\mathbf{O}_{fov} \tag{3.48} $$

where ${}^{arm_i}\mathbf{R}_{fov}$ indicates the orientation of the frame *fov* relative to arm_i, which can be written as:

$$ {}^{arm_i}\mathbf{R}_{fov} = \mathbf{R}_z(\alpha) = \begin{pmatrix} cos(\alpha) & -sin(\alpha) & 0 \\ sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.49} $$

3.7 Patrolling Strategies

A patrolling strategy basically defines the points of the MIRADAS FOV that each probe arm is permitted to observe. Here we discuss two different approaches: (1) *Workspace* patrolling and (2) *Slice-of-Pie* (SoP) patrolling.



(a) The full-scale MIRADAS MXS plate is divided into 5 equal portions, delimited by the grey dotted lines. Each of these portions contains one probe arm, represented here by its *envelope* and *workspace*. As can be appreciated, these layout presenting only 5 arms can patrol the whole instrument FOV (black dashed circle).

(b) The MXS plate is divided into 12 equal-size regions. There are three *envelopes* represented: the one for arm_0 is shown in blue and the envelopes of its neighbors, arm_1 in green and arm_{11} in red. The size of the area resulting from the intersection of the three envelopes gives an idea of the number of collisions that can arise with only these arms.

Figure 3.14: Aerial views of the MIRADAS MXS bench with two different probe arm distributions.

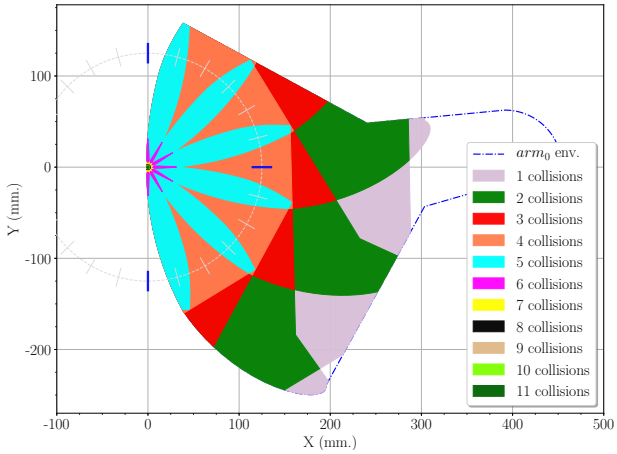


Figure 3.15: The potential collisions of arm_0. Each colored area inside the arm envelope represents the maximum number of intersections arm_0 can experience when traversing such zone.

3.7.1 Workspace Patrolling

In this patrolling policy, each arm of the system can access all points of the MIRADAS FOV contained in its workspace. As it is shown in fig. 3.14a, the whole field-of-regard can be patrolled with only five arms evenly distributed around the MXS bench. Hence, with more arms available, as it is the case in fig. 3.14b where the real layout of MIRADAS can be appreciated, the overlapping between workspaces increases, providing this way a higher degree of redundancy and flexibility to the MXS system. These characteristics are very helpful, especially when one or more arms are broken. But, unfortunately, this redundancy negatively impacts on the complexity of the motion planner. It also has implications on the computation time since the chance of two arms colliding increments, consequently requiring more extra checks.

From the point of the control system, it could be useful to know those regions of an arm envelope having higher collision probability. The motion planning algorithm could take advantage of this information, avoiding the generation of paths passing through some conflicting zones. These different collision areas are shown in fig. 3.15. As can be appreciated, the potential intersections increase as the arm gets closer to the FOV center, a point reachable by all arms in the system.

3.7.2 SoP Patrolling

In this strategy the field-of-regard is divided into as many identical areas as there are arms in the MXS system. Each region, called slice-of-pie, is a subset of the arm workspace. A given SoP is patrolled by only one arm and one arm patrols only one SoP. The primary aim of this approach is to reduce the broad search tree that arises from workspace patrolling and, at the same time, make the control simpler. This approach can be effectively employed and will guarantee that there are no potential collisions between any pair of arms as long as the following conditions are met:

(a) **Coverage**: the area resulting from the combination of all SoPs covers the whole FOV.

(b) **No over-lapping**: there is no over-lapping between the different SoPs in the system.

(c) **Envelope**: there is no intersection between the envelopes associated to each SoP.

(d) **Motion**: a mechanically feasible trajectory can be determined between any two points of a concrete arm SoP without traversing other arm SoP.

The above constraints can be easily accomplished with a mechanical structure based on a revolute-prismatic (RP) open kinematic chain. A manipulator of this kind, which is composed of two joints, describes a polar (r,φ) motion in the Cartesian plane. That is the case of the mechanical design of the arm used in KMOS (Bennett et al., 2008), an instrument for ESO's VLT technically similar to MIRADAS.

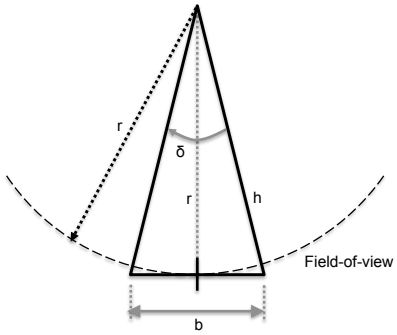


Figure 3.16: In SoP patrolling, the slice-of-pie defines the portion of the MIRADAS FOV patrolled by each probe arm of the MXS system. Here we can see how the size of a triangular SoP can be calculated if we know δ (which depends on the number of arms in the system) and r, the radius of the instrument FOV.

3.7.2.1 SoP Shape

From the point of view of MIRADAS FOV, the more logical and intuitive choice is a quasi-triangular area in the arm Cartesian space. The size of the triangle enclosing this SoP, seen in fig. 3.16, can be determined considering the arc of the FOV each arm has to cover (δ) and the radius (r) of the FOV:

$$\delta = \frac{2\pi}{n} \tag{3.50a}$$

$$h = \frac{r}{\cos(\delta/2)} \tag{3.50b}$$

$$b = 2h\sin(\delta/2) \tag{3.50c}$$

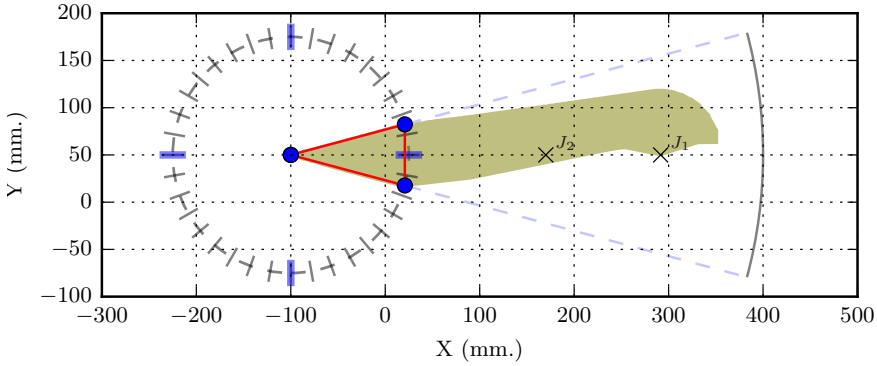
where n is the number of arms in the MIRADAS MXS system.

3.7.2.2 SoP Envelope

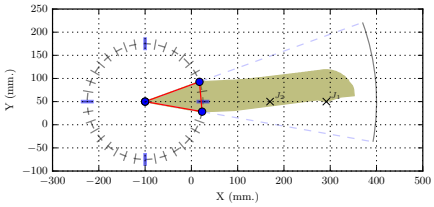
As discussed above, the SoP patrolling strategy can be effectively implemented if there is no superposition between the envelopes associated with each SoP of the MXS system. We define the *envelope of one SoP* as the area swept out by one arm when it patrols all points contained in its assigned SoP.

3.7.2.3 SoP Location and Assignment

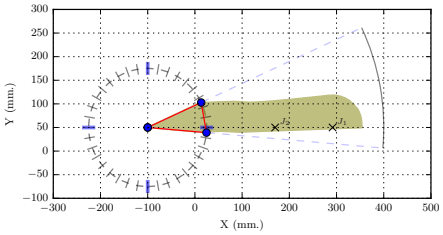
Once known the shape and size of the SoP, we can divide the MXS plate into n equal SoPs. Each of these SoPs has to be allocated to a different arm, which will be the one in charge of



(a) The black dashed circle centered at $(-100, 50)$ represents the MIRADAS FoV. We can also see one (grey dashed triangle) out of 12 triangular pieces that form the MXS plate when this contains a total of 12 probe arms. The arm placed in that piece patrols the triangular SoP in red. The area in brown is the envelope associated to this arm when it patrols all points contained in the SoP shown.



(b) The same MXS system configuration than in Figure 3.17a, but here the SoP is rotated 5° counterclockwise. The placement of the probe arm is the same as in Figure 3.17a. The $\mathscr{E}_{sop}$ in brown.



(c) The same MXS system configuration than in Figure 3.17a, but here the SoP is rotated 10° counterclockwise. The placement of the probe arm is the same as in Figure 3.17a. The $\mathscr{E}_{sop}$ in brown.

Figure 3.17: Three different FOV locations of the SoP assigned to a given arm.

patrolling any point inside its assigned SoP. Although multiple options arise, fig. 3.17 shows three good placements in terms of collision avoidance. In Figure 3.17a, it can be appreciated the most intuitive placement, where each arm is assigned the triangular SoP that is perfectly aligned with the line defined by the centers of the arm joints J_1 and J_2. That is, the SoP is symmetric with respect to the straight line passing through the center of J_1 and J_2. In Figure 3.17b and Figure 3.17c, we show the same triangular SoP but now rotated 5° and 10° counterclockwise respectively. In all these three solutions, the envelope created by the arm is totally contained in the area resulting from projecting the SoP boundaries beyond the MIRADAS FOV. Thus, there is no intersection between the envelopes of adjacent arms apart from the points belonging to the common sides of two adjacent SoPs. By selecting any of these SoP placements, the envelope constraint (item (c) of the list in section 3.7.2) is satisfied.

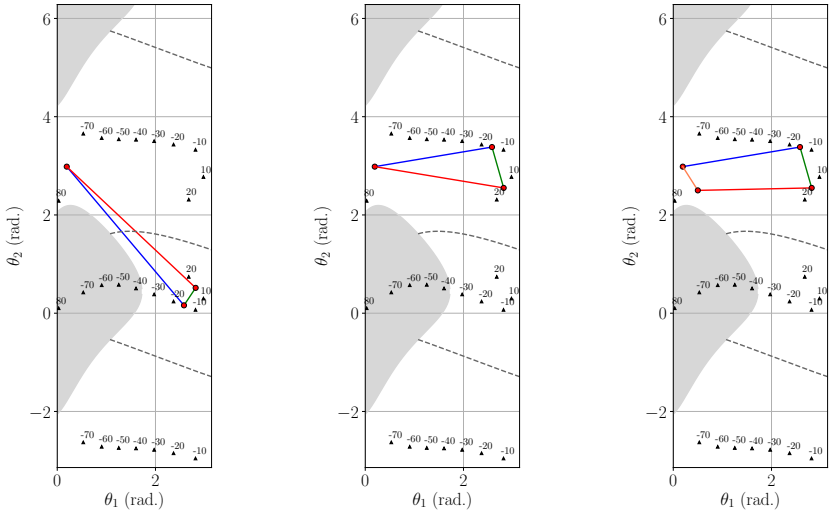
3.7.2.4 Alternative SoP Shapes

Unfortunately, any SoP with a triangular-like shape in the arm workspace translates into a configuration space area with a curved boundary, see equations in section 3.4.2. Thus, any two points belonging to the joint space cannot be connected with a single straight line, the preferred path due to the motor controller particularities commented in section 3.5.1.2. We, therefore, explore here alternative SoP shapes that might be more advantageous from a control perspective.

Considering the constraints imposed by the commercial controller, a good choice for a SoP would be one that when converted into the articular space it becomes a polygon. Consequently, a reasonable approach would be to focus first on convenient SoP in the articular space and then analyzing the corresponding shape once transformed into the Cartesian plane. A polygon in the configuration space leads to a curved shape in the Cartesian plane; refer to equations in section 3.4.1. This shape could be effectively employed as SoP if it satisfies, at least, the full coverage condition mentioned at the beginning of section 3.7.2. That is, if the surface resulting from the addition of all SoPs in the Cartesian space covers the entire FOV.

When constructing polygons in the configuration space, their vertexes should be chosen carefully. If not, the associated shape in the Cartesian space could be inappropriate. To illustrate this, we start by selecting three representative points in the Cartesian space: the FOV center and two points in the FOV boundary separated by 30°. If we translate these points into articular coordinates and connect them by straight lines, we obtain the triangle in fig. 3.18a. In this figure, the original configuration space and the ZoA is shown as well as one replica of them along the vertical axis. Just as a reminder, mention that due to the topology of joint J_2 (360° range of motion), both ends of the interval $[0, 2\pi]$ are glued together. That is, the value 2π wrap around to 0. As a result, some points of the articular space are identical, even though they initially seem distinct. In addition, in the figure, we have also used black dashed lines to delimit the regions of the configuration space associated with the generation of the upper and lower workspace of the arm. Points contained in between two consecutive dashed lines belong to the same 3D workspace region. Small triangular markers have been also utilized to represent the configurations positioning the arm tip in the corresponding degree marker on the FOV boundary shown in fig. 3.19b. As expected, a majority of points in the FOV boundary can be reached by more than one point in the configuration space.

At first sight, the triangle at fig. 3.18a looks as if it would be a good choice from a control point of view. But, once it is converted back to the Cartesian plane, an undesired effect arises, as appreciated in fig. 3.19a. The boundary of the resulting Cartesian shape intersects itself. The intersection between the red and blue curves is due to the connection of points of the configuration space belonging to different regions of the arm workspace, one from the upper side and the other from the lower. To avoid this sort of conflicts, all the vertexes of the polygon selected in the configuration space, once properly translated, should belong to the same region of the 3D workspace. Taken into account this, a valid triangular SoP in the configuration space



(a) The articular SoP here is a perfect triangle, where the vertexes come represented by the center of the FOV in the upper workspace and the 15° and -15° points in the lower one. The associated Cartesian SoP is shown in fig. 3.19a.

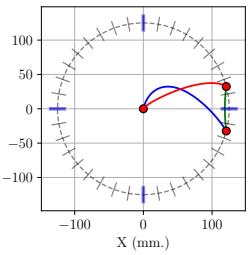
(b) A different triangular SoP connecting the same three Cartesian points than in fig. 3.18a, but now all of them are located in the arm upper workspace. This polygon leads to the Cartesian SoP shown in fig. 3.19b.

(c) A quadrilateral. Again, all points of the four-edge polygon belong to the arm 3D upper workspace, therefore, no intersections will occur in the Cartesian plane. This shape leads to the Cartesian SoP shown in fig. 3.19c.

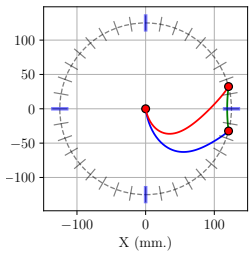
Figure 3.18: Alternative SoPs in the articular space.

is represented in fig. 3.18b. Here, all points of the SoP boundary are contained in the same 3D workspace region, resulting then in a nonconflicting curved shape in the Cartesian plane; see fig. 3.19b. Although this solution could be convenient to reduce the arm control complexity, unfortunately, as seen in Figure 3.19d, many spots in the FOV could not be observed because they are not patrolled by any of the 12 arms in the system.

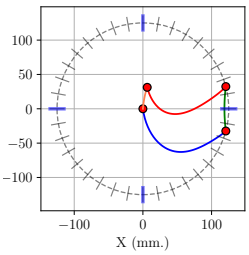
In fig. 3.19c, a different sample SoP shape is shown. It is the result of converting the four-sided polygon represented in fig. 3.18c to the Cartesian plane. Although this choice presents a higher degree of FOV coverage, as appreciated in fig. 3.19e, some areas still remain uncovered. On the other hand, this is an excellent example of the limitations of this patrolling approach when considering particular motor controllers characteristics. More elaborated polygons in the configuration space can be attempted to achieve an even higher level of coverage. But, they definitely will lead to such convoluted Cartesian representations that determining the SoP where a given target lies in would turn into a very complicated task.



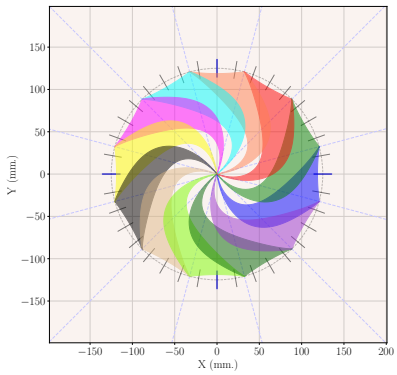
(a) A SoP that crosses itself. Each mark in the SoP boundary represents and increment of 10°. The SoP connects three points: the FOV center, a point in the FOV boundary at 15° and other at -15°.



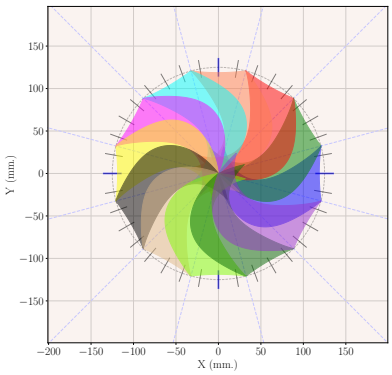
(b) A valid alternative SoP connecting the same points that the ones in fig. 3.19a. This is the Cartesian plane shape arising from the triangular SoP in the configuration space seen in fig. 3.18b.



(c) Other valid alternative SoP connecting four different points. This is the Cartesian plane shape arising from the four-sided polygon SoP in the configuration space that can be seen in fig. 3.18c.



(d) Twelve replicas of the valid SoP seen in fig. 3.19b properly distributed in the instrument FOV fig. 3.19a.



(e) Twelve replicas of the valid SoP seen in fig. 3.19c. Note that the unpatrolled spots in the FOV are smaller than those in fig. 3.19d

Figure 3.19: Alternative SoPs in the Cartesian space.

3.8 Conclusion

Prior to the design of any MIRADAS motion technique, a detailed probe arm study is required. In this chapter, we covered the more relevant characteristics of the probe arm, providing an in-depth analysis of its behavior. In particular, we presented a kinematic model and the configuration space, paying special attention to that region of it that is prohibited since would lead to mechanically infeasible arm configurations. Additionally, we introduced the envelope as well as the workspace

of the MIRADAS probe arm. We saw that due to the 360° range of motion of one active joint, the same point in the workspace can be frequently reached by two different configurations. One of them belonging to what we have called the *upper* workspace and the other to the *lower* one. Also, we mentioned how motion planners could exploit this particularity.

This chapter presented two different arm patrolling strategies as well. The workspace patrolling is the one providing, in principle, target allocation algorithms with a higher degree of versatility. Implementing the SoP patrolling strategy with other mechanisms, such as polar motion manipulators, is an intuitive process that can give good results. However, in MIRADAS, it is a challenging task. As seen, the options for a suitable SoP are drastically reduced because of the particular arm geometry and the restrictions imposed by the motor controllers. Although this strategy might be usable, if applied, it would not ensure that all points of the instrument FOV could be observed. Finally, we introduced the limitations that the COTS controllers employed present, especially their inability to interpolate trajectories. These limitations constrain the design of motion planners, which will have to try to reduce the number of intermediate points in a trajectory to avoid arm wear. Experimenting with alternative motor controllers, especially more advanced ones offering interpolation functionalities, would be a natural choice. But, unfortunately, to the best of our knowledge, the controllers found by the authors capable of interpolating are not 100% compatible with the cryogenic motors used in the design.

CHAPTER 4

FIELD SEGMENTATION

Nowadays, MOS has become a very important observation technique in astronomy. It is especially suited for those cases where there are a large number of astronomical targets sparsely distributed in the FOV such as star or galaxy clusters and the properties of several individual objects want to be studied. In general, planning observations for MOS is more involved than it is for instruments not providing this functionality. MOS instruments need to be adequately configured to the particular target field of interest before starting observations. This process tends to be different for each MOS instrument since it is not easy to find instruments showing a high degree of similarity. For instance, in MIRADAS the motions of the 12 probe arms to reach their corresponding destinations have to be carefully computed. However, before addressing that task, the targets must be analyzed and grouped adequately for simultaneous observation. This selection of targets in MOS instruments is often known as *field configuration*, being its main aim to maximize the allocated target yield to ensure the efficiency of the multiplexing systems.

If all the user-defined targets are close enough to fit all of them inside the area of the MIRADAS FOV, field configuration basically consists in an allocation problem of determining a sequence of several assignments plans, each of them containing up to 12 <*target, arm*> pairs. Each of these pairs defines which of the available arms in the system is responsible for observing a specific target. The number of pairings returned mainly depends on the location of the targets as well as the kinematics characteristics of the arms, but they can never be greater than the number of arms available in the MXS system. Although many criteria exist for guiding this matching process, the most widely employed in many MOS instruments is grouping celestial targets as a function of their importance.

Contrarily to other similar instruments, MIRADAS has been conceived as a common-user instrument. This means that, during its years in operation, MIRADAS should be versatile

enough to observe a wide variety of fields, most of them not included in the science cases that motived the construction of the instrument. Consequently, the targets might be anywhere and especially scattered over an area more extensive than the instrument's FOV. If that is the case, then, the telescope will need to be moved to different sky locations before the acquisition of the corresponding targets. These telescope pointing locations as well as the targets to be observed in each of them are determined by an additional processing step that we call *segmentation* of scattered fields. It computes some sort of arrangement that properly distributes the targets in different groups or *clusters* so that all elements in a cluster fit in the FOV region. Hence, it makes sense that this clustering task groups targets following a geometrical approach: the distance between any pair of elements in a set has to be less than or equal to the FOV diameter. This partition of a given field is performed before addressing the arm allocation problem introduced above.

This chapter provides a contribution related to automatically partitioning fields. In particular, in section 4.2, we lay the basis for understanding the segmentation approach utilized in MIRADAS, which is discussed in section 4.3. But, first of all, we start in section 4.1 by introducing how celestial points are defined and the relationships between the involved reference systems. These transformations are indispensable for converting bodies defined in an astronomical reference system into positions in the instrument FOV, which is the kind of coordinates used in the MIRADAS field configuration algorithm. To the best of our knowledge, the field configuration tools for the already built MOS instruments assume that the user is responsible for manually segmenting scattered fields (Carlin et al., 2012; Morales et al., 2012; Saunders et al., 2014; Wegner and Muschielok, 2008; Yuan et al., 2013). However, EMIR, a multi-slit MOS sharing the common-user characteristics of MIRADAS, requires this functionality as well. As far as we know, it is not implemented yet (Garzón et al., 2006, 2014).

Parts of this chapter have previously been published as: "Using clustering for dispersed objects fields segmentation in MIRADAS instrument", Proc. SPIE 10707: Software and Cyberinfrastructure for Astronomy V, 107070N, 2018 (Sabater et al., 2018b).

4.1 Positions on the Sky

Astronomers define the location of bodies in space by using their position in a celestial coordinate system. Even though these objects can be at a wide range of distances from us, when astronomers think about them, they do as if all were projected on the inside surface of an imaginary sphere known as the celestial sphere. Since, for convenience, they implicitly assume that all bodies are at the same distance, astronomers work only with two coordinates: *longitude* and *latitude*.

There exist a variety of reference systems. The three most common differ in how they define a central location and the lines of zero latitude and zero longitude as seen from that central location. These are the *equatorial*, *ecliptic* and *galactic coordinate* systems, which utilize the

Earth, the ecliptic plane[1], and the center of our galaxy, respectively, as their central locations.

In the following section, we will introduce the Equatorial system, employed by scientists in MIRADAS. More details on Spherical Astronomy, the field studying the location of objects on the celestial sphere, can be found in Green and Green (1985) and Zombeck (2006).

4.1.1 The Equatorial Coordinate System

This is the standard reference system for professional and amateur astronomers using telescopes equipped with equatorial mounts following the movement of the sky during the night. This system is very similar to the latitude and longitude coordinate system used in Earth. The Earth's equator and poles are projected outward into space to form the celestial sphere with the Earth at the origin of the system. The celestial equator is the *great circle*[2] formed by the projection of Earth's equator onto the sky, whereas the North and South celestial poles are the points on the sky directly above Earth's North and South poles.

As shown in fig. 4.1, positions of celestial objects are described by their *right ascension* (RA) and *declination* (DEC), respectively defining the angle eastward from the vernal equinox[3] and the angle above or below the celestial equator. Analogously, the RA component refers to the object's longitude (east-west direction), whereas DEC to the object's latitude (north-south direction). Unlike Earth's reference system, the RA position, represented by the symbol α, is measured in hours, minutes and seconds, whereas the DEC component, symbolized by δ, in degrees, arcmin and arcsec.

4.1.2 From Celestial Coordinates to Coordinates in the Focal Plane

As stated before, bodies on space are defined using celestial coordinates. In MIRADAS, they come specified in the Equatorial reference system described above. However, MIRADAS field configuration is performed considering the positions of the objects in the instrument focal plane. Therefore, we need to understand how we can properly project a celestial field onto the surface of the MIRADAS MXS plate as well as the transformation between the reference systems governing both spaces. That is, we need the expressions converting an equatorial coordinate (α,δ) into a standard coordinate (ξ,η).

4.1.2.1 Standard Coordinates and Plate Scale

A standard coordinate (ξ,η), also known as *tangential* or *normal* coordinate, is expressed in terms of a rectangular Cartesian system that lies in a plane tangent to the celestial sphere. By means of *gnomonic* projection, a tangent plane can be constructed from a curved area. This is achieved by projecting the axis of the optical system upwards to intersect the celestial sphere, with the

[1] The path the Earth takes around the Sun.
[2] It passes through the center of the sphere, splitting the sphere into two identical hemispheres.
[3] The point where the celestial equator intersects the ecliptic.

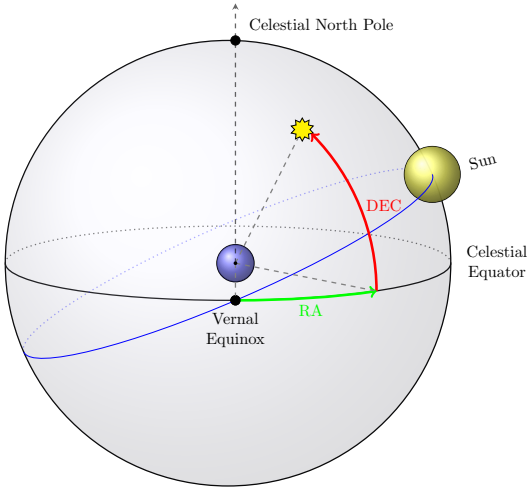


Figure 4.1: The celestial sphere with its Equator and North Pole. Positions of objects are specified by their RA and DEC coordinates. The Vernal Equinox determines the zero for the RA component.

intersection point T, defined by (α_0, δ_0), lying at the origin of the standard reference system; see fig. 4.2. The positive ξ- and η-axes are defined with respect to increasing right ascension and increasing declination respectively. In other words, the ξ-axis is tangent to the right ascension circle passing through T, with positive values lying toward the east, while the positive η-axis is perpendicular to ξ-axis and points toward the north pole.

The standard coordinates are usually given in arcmin, arcsec, radians or degrees. The (ξ, η) coordinates of a celestial object located at (α, δ) with respect to the tangential point T at (α_0, δ_0) are given by the following expressions

$$\xi = \frac{\cos\delta \sin(\alpha - \alpha_0)}{\sin\delta_0 \sin\delta + \cos\delta_0 \cos\delta \cos(\alpha - \alpha_0)} \tag{4.1}$$

$$\eta = \frac{\cos\delta_0 \sin\delta - \sin\delta_0 \cos\delta \cos(\alpha - \alpha_0)}{\sin\delta_0 \sin\delta + \cos\delta_0 \cos\delta \cos(\alpha - \alpha_0)} \tag{4.2}$$

and the inverse transformation is obtained by

$$\tan(\alpha - \alpha_0) = \frac{\xi}{\cos\delta_0 - \eta \sin\delta_0} \tag{4.3}$$

$$\sin\delta = \frac{\sin\delta_0 + \eta \cos\delta_0}{\sqrt{1 + \xi^2 + \eta^2}} \tag{4.4}$$

Further discussion of the above expressions can be found in Zombeck (2006) and the corresponding derivation in Green and Green (1985).

Two things should be pointed out in relation to standard coordinates. First, as stated before, the standard coordinates are calculated using a gnomonic projection, which is neither conformal nor equidistant (Coxeter et al., 1969). Thus, generally, angles and distances in the resulting tangent plane do not equal angles and distances on the celestial sphere. However, angles measured at T are the same on the sphere and the plane. On the other hand, strictly speaking, telescope pointing coordinates are required to translate from the angular space into the linear one. This information specifies the point T, which, at the same, determines the origin of the tangential plane. But, in MIRADAS, the tangential point is not explicitly provided by the astronomer along with the celestial coordinates of the objects to be observed. We assume that T is at the center of the cloud of those celestial points, which, in fact, it is a good enough approximation in our case.

The Plate Scale It relates an angular position of an object to its corresponding linear position at the focal plane. This scale factor is particular to each telescope since it depends on the characteristics of all optical components. It is often expressed in arcsec/mm and can be determined as follows

$$p = \frac{206265}{F} \tag{4.5}$$

where F is the telescope's Effective Focal Length (EFL) in millimeter, and the constant 206265 converts from radian to arcsec.

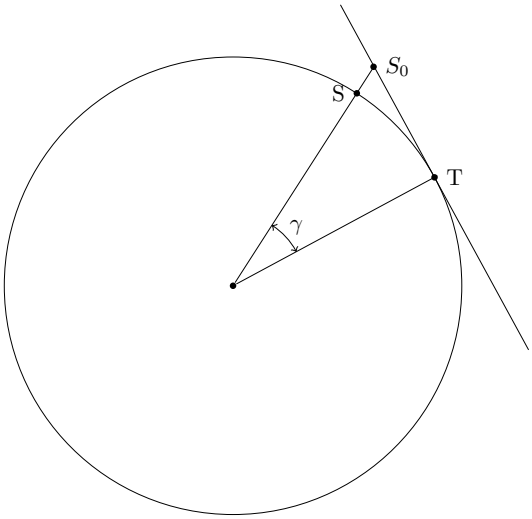


Figure 4.2: Construction of a tangent line at the tangent point T and the projection of a point S located on the celestial circle into the line.

4.2 Introduction to Analysis Techniques for Segmentation

The partition of a data set of interest into distinct subsets so that members within a group are related to one another is known as *cluster analysis*, a well-established technique in data mining. In a broad sense, data mining refers to the process of discovering patterns in large data sets. Nowadays, astrophysics and cosmology are rapidly becoming more and more rich with data, where catalogs are increasing from gigabytes into terabytes and petabytes. However, solving specific data mining problems such as clustering and classification has been standard practice in classical astronomy. For ages, experimental astronomers have been collecting data on objects in the sky to analyze their physical properties, and, finally, to understand the underlying physics leading to such properties. For instance, supervised and unsupervised machine learning have been lately adopted to analyze many of the available sky surveys. Data mining has provided useful information to identify galaxy morphologies, to classify celestial objects or to seek for novel discoveries (Djorgovski et al., 2000; Edwards and Gaber, 2013; Kremer et al., 2017). For a thorough survey of data mining and its applications in astronomy, the interested reader is referred to the works of Ball and Brunner (2010), Edwards and Gaber (2014), and Borne (2009).

In the context of data mining, the goal in cluster analysis is that points being in the same cluster have high similarity from one another, while points in different clusters are very dissimilar from one another. There exist a sheer number of clustering methods for data mining, with several books such as Everitt et al. (2011) and Aggarwal and Reddy (2013) exclusively devoted to this general area of knowledge. However, in astronomy, clustering often has the more concrete meaning of grouping together elements considering their spatial locations. Specifically, astronomers find groupings of stars close together in the sky, which they call star clusters. Moreover, on a larger scale, astronomers look for clusters of galaxies, which are groupings of galaxies. No clustering technique is universally applicable, and different techniques are in favor of distinct clustering purposes. A brief review of clustering concepts and methods more relevant from the point of view of segmentation in MIRADAS is given in the following sections.

4.2.1 Proximity measures

In clustering, we need a way to properly measure the elements and the relative proximity to each other in order to decide which of them belong to a group. Proximity measures are expressed in two forms, *dissimilarity* and *similarity*, although on many occasions the concept of distance is used. These metrics basically tell how different or resembling are two entities. They are widely utilized to compare two individual points of a collection; however, they can be extended to compare one point and a set of points, and to compare two distinct sets of points.

4.2.1.1 Measures for a pair of points/vectors

Here, we introduce three popular measures, beginning with the most common family of dissimilarity measures, the Minkowski distance. In fact, there exists a myriad of metrics, including the Hamming, Mahalanobis, and Hellinger variational distances. In Deza and Deza (2009), Anderberg (1973), and Everitt et al. (2011), a discussion about the application of these other metrics is provided.

l_r Norm Dissimilarity Measure: This generalized dissimilarity measure is also known as weighted Minkowski distance (d_M) and is defined as follows

$$l_r = d_M(X,Y) = \left(\sum_{i=1}^{n} {w_i}^r |X_i - Y_i|^r \right)^{1/r} \tag{4.6}$$

where X and Y are two points of dimension n.

The weights w_i assign the importance of each attribute. If $w_i = 1$ for $1 \leq i \leq n$, this metric is called unweighted l_r. The city block distance or l_1 norm describes distances on a rectilinear configuration. It is also commonly referred as Manhattan distance as it measures distances traveled in street configuration. The unweighted l_2 norm is Euclidean distance. This distance metric measures dissimilarity; therefore, the higher the value, the more dissimilar are two points.

Cosine Similarity Measure: This normalized metric is a measurement of orientation and not magnitude. Hence, it tells how related are two points by looking at the angle α between them

$$\cos\alpha = s_{cos}(X,Y) = \frac{X \cdot Y}{\|X\|\|Y\|} = \frac{X^T Y}{\|X\|\|Y\|} \tag{4.7}$$

where X and Y are two different vectors, and $X \cdot Y$ represents the dot product between two vectors. This is a similarity measure. So, the higher the value, the more similar two objects are. This measure is broadly employed in text retrieval to match vector queries to the dataset (Murtagh and Contreras, 2012).

Dot Product Similarity Measure: This metric measures similarity with respect to angle and amplitude

$$s_{dot}(X,Y) = X \cdot Y = X^T Y \tag{4.8}$$

where X and Y are two different vectors. If both vectors are unit length, then the dot product is equivalent to the cosine similarity.

4.2.1.2 Measures for sets of points/vectors

Proximity measures between two vectors are simple to extend to those cases involving a vector and a set of vectors or two sets of vectors by using functions such as *min* and *max*. Let d be some

dissimilarity defined on two points, denoted by x and y, and Γ a set of points:

$$d_{min}(x,\Gamma) = \min_{y\in\Gamma} d(x,y) \tag{4.9}$$

$$d_{max}(x,\Gamma) = \max_{y\in\Gamma} d(x,y) \tag{4.10}$$

These two expressions can be generalized for two different sets of points, Γ and Δ, determining what in clustering is known as *linkage criterion*

$$d_{min}(\Gamma,\Delta) = \min_{x\in\Gamma, y\in\Delta} d(x,y) \tag{4.11}$$

$$d_{max}(\Gamma,\Delta) = \max_{x\in\Gamma, y\in\Delta} d(x,y) \tag{4.12}$$

Equation (4.11) and eq. (4.12) are popularly termed as *single-linkage* and *complete-linkage*, respectively. The former defines the proximity between two sets of points as the minimum proximity between their members. And the latter specifies the proximity between two sets as the maximum distance between their members. Single-linkage leads to *nearest-neighbor* clustering and complete-linkage to *farthest-neighbor* clustering.

However, in some contexts, especially in astronomical data analysis, *Ward's minimum variance* criterion (Murtagh and Legendre, 2011; Ward Jr, 1963) is more indicated. Such is the case, for instance, in the ESA Gaia mission. There, this criterion is used in conjunction with hierarchical clustering during the cross-matching process, which provides a link between every astronomical observation and the corresponding source in the catalog (Lindegren, 2005).

In the Ward's method, the intrinsic dissimilarity between two sets of points Γ and Δ is measured by the sum of squared residuals (SSR) with respect to the point set center

$$d_{Ward}(\Gamma,\Delta) = R(\Gamma\cup\Delta) - R(\Gamma) - R(\Delta) \tag{4.13}$$

where $R(C)$ is SSR in the set C.

The sum of squared residuals of an arbitrary set of points Ω is:

$$R(\Omega) = \sum_{x\in\Omega} = \|x - m(\Omega)\|^2 \tag{4.14}$$

where $m(\Omega)$ is the center of the set Ω, whose coordinates are chosen to minimize the SSR. In the linear case corresponds to the center of gravity

$$m(\Omega) = \frac{1}{n}\sum_{x\in\Omega} x \tag{4.15}$$

where n is the number of points in the set Ω.

A good overview of more commonly employed linkage criteria between two sets of points such as *centroid-linkage* and *average-linkage* can be found in Murtagh and Contreras (2012).

4.2.2 Probabilistic Clustering

In probabilistic clustering, a model is constructed, and the data is grouped according to its statistical nature. The basic assumption behind is that in the input dataset there exists a specified number of populations, the clusters, each of which characterized by its own probability density function. Therefore, the resulting model, known as Mixture Model, is composed of a weighted sum of several previously picked generating distributions (McLachlan and Peel, 2004). The form of these individual functions is known, being Gaussian the most frequently employed, but their parameters and prior probabilities are unknown. Additionally, it is also unknown which mixture generated each of the individual points in the dataset. These unknown variables of the mixture are often iteratively computed with the help of the Expectation Maximization (EM) algorithm, which finds *maximum likelihood* estimates (Dempster et al., 1977; McLachlan and Krishnan, 2007). Many astronomical datasets consist of contributions from different populations of objects. Therefore, they can be in some way modeled utilizing mixtures (Connolly et al., 2000).

In the classical approach, the number of clusters to find needs to be prefixed in advance. However, lately, the Mixture Model technique has been extended to estimate the number of clusters in the population automatically (Fraley and Raftery, 2002). Although this extension can be applied to problems requiring density estimation, the results obtained are often sensitive to the assumption of the parametric family used in the model. Additionally, it is more suitable for ellipsoidal-shaped clusters (Stuetzle, 2003). Thus, in those scenarios where we cannot assume that the underlying data structure can be adequately represented in statistical terms, heuristic methods like proximity-based algorithms are of interest.

4.2.3 Proximity-based Clustering

Proximity-based algorithms are widespread in real applications due to their simplicity and ease of implementation. K-means (MacQueen and others, 1967) is one of the most widely applied technique in this family, and a large number of variations have been proposed (Aggarwal and Reddy, 2013). This partitional algorithm following a *greedy* approach iteratively finds k clusters, each of which characterized by its respective centroid. The clustering process starts by picking k representative points from the data set, being each the initial centroid of its associated cluster. Then, the technique determines the best centroids and consequently, the clusters by alternating between two steps:

- Each point in the dataset is assigned to one of the k clusters based on the current centroids and a particular proximity measure chosen.
- The centroids of the clusters are updated based on the current assignment of data points to clusters.

These steps are sequentially repeated until the centroids do not change or any other alternative relaxed convergence criterion is satisfied.

The basic K-means algorithm uses Euclidean distance as a proximity measure, which makes its applicability to astronomy quite limited (Bruzzone et al., 2004; Chekanov, 2006). In principle, approaches employing Ward minimum variance (Lindegren, 2005; Ward Jr, 1963) are more useful in astronomical applications, especially when paired with a hierarchical clustering technique; see section 4.2.4. Additionally, the K-means approach presents a few limitations. It is very sensitive to the initialization method and requires the specification beforehand of the number of distinct clusters to extract from the input dataset, which is what we precisely want to determine in MIRADAS. In practice, the former can be overcome by running several times the algorithm with different initial populations. However, the latter is more challenging, and many researchers have studied ways to address it (Aggarwal and Reddy, 2013). Luckily, in other techniques such as hierarchical clustering, the number of clusters can be automatically or analytically estimated.

4.2.4 Hierarchical Clustering

Hierarchical clustering, unlike other families of algorithms, does not require the explicit specification of the number of clusters to find. They produce a hierarchic relationship between the different elements of the population, which is frequently represented utilizing a *dendrogram*. Depending upon whether the hierarchy is built following a bottom-up or a top-down approach, the algorithms are respectively classified as *agglomerative* or *divisive*.

In agglomerative clustering, all points of the population begin as clusters of their own, and then those showing greater proximity are progressively merged until all points belong to the same cluster. On the other hand, in the divisive approach, the whole data set is treated as a single cluster that is iteratively partitioned.

4.2.4.1 Dendrogram

A characteristic of both approaches is that during the clustering process, the corresponding hierarchical algorithm builts a dendrogram by recording the proximity values whenever clusters are combined or split up. A dendrogram provides an overview of the distinct sets of clusters existing at different levels of proximity. It can be seen as a tree specifying the hierarchic relationships between the data in the set. This tree has many leaves at one end, corresponding to all points in the input data set. As we move toward the other end of the tree, similar data is progressively merged into branches until we reach the root, which contains all points. Sometimes, for convenience, the tree is displayed horizontally.

4.2.4.2 Selecting a Clustering from the Hierarchy

Unfortunately, popular algorithms do not automatically provide a single clustering[4] but rather a set of possible solutions. That is because the algorithms construct the whole hierarchy, not allowing the sudden interruption of the building process when the tree reaches a particular height.

A solution is often obtained by visually inspecting or analytically analyzing the dendrogram and then cutting it at the desired height, yielding this way a particular set of clusters. Alternatively, once computed the complete dendrogram, the user decides the number of clusters, and the tree is cut accordingly. However, in the case of MIRADAS, as we will see, there is no need to construct the full tree since we can determine well in advance the height where it has to be trimmed.

There exists in literature different optimization-based methods for branch pruning [5] such as the within cluster sums of squares or the average silhouette, named *elbow* and *silhouette* methods respectively (Jung et al., 2003; Ketchen Jr. and Shook, 1996; Rousseeuw, 1987). But there is no magic formula for selecting one from the many available since the suitability of each is context-dependent. The most common dendrogram partition methods consider each contiguous branch below a fixed cut-off height a separate cluster. This is the case of the one based on the *self-proximity* measure. For a given cluster Γ, self-proximity is defined as

$$h(\Gamma) = \max_{x,y \in \Gamma} d(x,y) \tag{4.16}$$

where d can be any proximity function, but often Euclidean distance is employed. Then, for a threshold value θ, the clustering selected is the one at level L in the hierarchy if at level $L+1$ there is at least one cluster Ω such that $h(\Omega) > \theta$.

4.3 Segmentation of Scattered Fields in MIRADAS

Although different hierarchical agglomerative methods can be found in literature, all employ a *similarity* matrix based on proximity metrics like the ones introduced in section 4.2.1. Additionally, all, as already stated in section 4.2.4, follow a common bottom-up strategy that starts by separating each component of the input data set into a different cluster. Then, at successive iterations, the two most similar clusters are combined into a new one. This sequential process ends when there is only one cluster left comprising the complete initial data set. The approaches only differ in two aspects: (i) the measure used to compute the *similarity* between two elements in the data set and (ii) the inter-cluster *linkage criterion*, which specifies how similarity between each pair of clusters is determined; see section 4.2.1.2 for more details. The choice of these both metrics influences the shape of the clusters found.

[4] A particular clustering comes determined by the number of groups (or clusters) in the dataset and their components.

[5] Automatic detection of the height where a tree should be cut.

The classical hierarchical agglomerative clustering procedure basically follows the next steps:

Step 1: Make one cluster for each of the n points in the initial data set. The number of active clusters is $M = n$.

Step 2: Determine the similarity matrix for the M active clusters, requiring $M(M-1)/2$ similarity calculations.

Step 3: Find the two active cluster having greater similarity, agglomerate them, and decrease M by 1.

Step 4: Repeat steps 2 and 3 until only one cluster is left ($M = 1$).

Although intuitive, the above greedy strategy presents two computational problems. First, it requires a significant amount of storage since it works with a $O(n^2)$ similarity matrix. Secondly, the time complexity is $O(n^3)$ as a result of the repeated computation of similarities for all possible pairs of points in conjunction with $O(n)$ iterations (Murtagh, 1983).

In the 1970s, more efficient algorithms appeared for some particular linkage criterion, computing the results in $O(n^2)$. In particular, implementations for the single (Sibson, 1973) and complete linkage criterion (Defays, 1977) were proposed. Nowadays, most of the current implementations for those linkage criterion fulfilling the Bruynooghe's *reducibility property* (Bruynooghe, 1977) often rely on the detection of *reciprocal nearest neighbors* (RNNs). RNNs are pairs of points i and j such that i is j's nearest neighbor (NN) and j is i's nearest neighbor. A given proximity metric expressed as a distance function d is considered to be reducible if, for every three clusters Γ, Δ and Ω where Γ and Δ are RNN, the following inequality is met (Murtagh, 1983, 1984)

$$d(\Gamma,\Delta) \leq \min(d(\Gamma,\Omega), d(\Delta,\Omega)) \Rightarrow \min(d(\Gamma,\Omega), d(\Delta,\Omega)) \leq d(\Gamma \cup \Delta, \Omega) \tag{4.17}$$

Unlike the classical approach, the RNN-based strategy does not strictly agglomerate clusters in the sequence of decreasing similarity. However, the resulting hierarchy is the same but computed in $O(n^2)$ time (Murtagh, 1983; Murtagh and Contreras, 2012).

On the other hand, the agglomerative implementations found in the most well-known cluster analysis packages/libraries[6] either (a) return the complete dendrogram, and it is the analyst who decides the cut-off point or (b) the number of clusters to be found is specified as an input parameter. Considering the geometrical constraints in MIRADAS, it makes no sense building the whole hierarchy. In contrast, automatically stopping the merging process as soon as the dissimilarity of the remaining clusters is larger than a given value (the diameter of the instrument FOV) seems more appropriate. That is why, for MIRADAS, we propose a threshold-based agglomerative clustering using RNN and the following metrics: (a) the Euclidean distance for proximity measures between any pair of points and (b) complete-linkage, which holds the reducibility criterion, for inter-cluster measures.

[6] These are *scikit-learn* or *scipy* in Python, *cluster* in R, and *fastcluster* in C++, which provides interfaces for Python as well as for R.

4.3.1 Threshold-based Agglomerative Clustering using RNN

By employing an algorithm building a nearest neighbor-chain (Benzécri, 1982; De Rham, 1980), RNNs can be found in an efficient manner. A NN chain is a sequence consisting in an initial arbitrary cluster followed by its NN, which is then followed by its NN from the remaining agglomerable clusters, and so on. A NN-chain satisfies that (i) similarities between consecutive clusters are not decreasing and (ii) it ends in a pair of RNNs. Starting from an arbitrary point, the algorithm grows a chain of NNs until a pair of RNNs is found. Then, this pair is removed from the chain and merged. The iterative construction of the chain continues until there is only one cluster left in the chain. Then, a new chain begins, and so on. The process stops when all initial points belong to the same cluster.

But, in the particular segmentation case of MIRADAS, we need to incorporate a threshold limiting the tree height to the general RNN clustering algorithm commented above. Such an approach is frequently utilized in image analysis. For instance, Leibe et al. (2008) discusses how RNNs together with average-linkage criterion can be effectively applied for object recognition. There, the authors provide pseudo-code as well, enabling this way a direct implementation of the solution in any programming language. However, although it shows how the NN-chains are grown, pseudo-code does not return the resulting clusters since they are not adequately stored once they cannot be agglomerated further. A more recent paper refines the previously mentioned work by introducing a faster way to construct the chain (Lopez-Sastre et al., 2012). This extension is particularly suitable for high-dimensional data and, in the proposed solution, the final clusters are, this time, explicitly returned.

Pseudo-code for the threshold-based agglomerative clustering approach used in MIRADAS is given in algorithm 1. As can be appreciated, the algorithm arbitrarily chooses a first cluster (line 2) and then, with the help of a routine that finds nearest neighbors (line 5), iteratively builds up an NN-chain (line 7). Once an RNN pair is discovered (line 8), the involved clusters are agglomerated if they show a similarity compatible with a given threshold (line 10). Otherwise, the chain construction is interrupted, and all its elements are included in the final cluster sequence (line 13). What really makes this implementation more efficient than the standard greedy approach is that the NN-chain can be reused in successive iterations. That is due to the reducibility property, which ensures that the merging of a given RNN pair does not affect the relations among the NN discovered before. Note that the pseudo-code provided does not consider what has to be done if a premature end occurs. In fact, it is possible that the while loop (line 4) exits when the current NN-chain is not empty. The works previously mentioned do not say anything about how to proceed in such a special situation. As practical information, it is worth commenting that the remaining clusters in the NN-chain list should be inspected again to check if they can be agglomerated further. If they cannot, the set of discovered clusters is updated with the corresponding clusters coming from the NN-chain.

An important point should be emphasized when applying this agglomerative algorithm to

Algorithm 1 Threshold-based agglomerative clustering using NN-chains

Require:
$\mathcal{P}$: a sequence with all points in the data set, where $p \in \mathcal{P} = (p_1, \ldots, p_n)$
t: agglomeration threshold
Ensure:
$\mathcal{C}$: a sequence with the clusters found

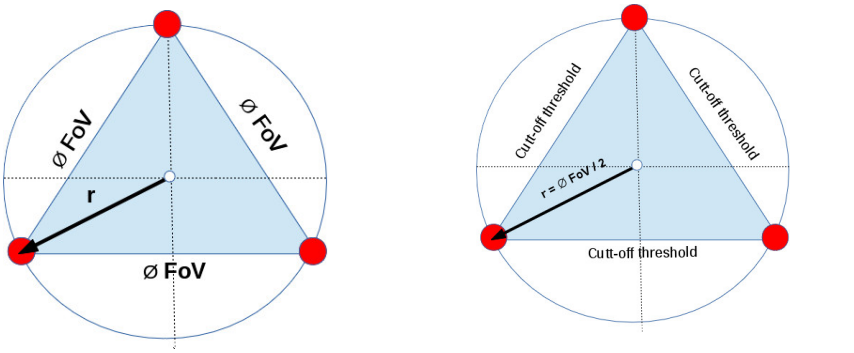
$\mathcal{C} \leftarrow \varnothing; last \leftarrow 0; lastSim[0] \leftarrow 0$
$\mathcal{N}[last] \leftarrow p \in \mathcal{P}$ ▷ Initialize NN-chain with an arbitrary point
$\mathcal{R} \leftarrow \mathcal{P} \backslash p$ ▷ Sequence with the remaining agglomerable clusters
while $\mathcal{R} \neq \varnothing$ **do**
  $(c, sim) \leftarrow$ GETNEARESTNEIGHBOR($\mathcal{N}[last]$, $\mathcal{R}$)
  if $sim > lastSim[last]$ **then** ▷ Update NN-chain with the cluster
    $last \leftarrow last + 1; \mathcal{N}[last] \leftarrow c; \mathcal{R} \leftarrow \mathcal{R} \backslash c$
  else ▷ RNN pair found
    if $lastSim[last] > t$ **then**
      $c \leftarrow$ MERGE($\mathcal{N}[last]$, $\mathcal{N}[last-1]$)
      $\mathcal{R} \leftarrow \mathcal{R} \cup c$; $last \leftarrow last - 2$
    else ▷ Update cluster sequence with the content in NN-chain
      $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{N}$; $last \leftarrow -1$; $\mathcal{N} \leftarrow \varnothing$
    end if
    if $last < 0$ **then** ▷ Initialize a new NN-chain with a random cluster
      $last \leftarrow 0$; $\mathcal{N}[last] \leftarrow p \in \mathcal{R}$; $\mathcal{R} \leftarrow \mathcal{R} \backslash p$
    end if
  end if
end while
return $\mathcal{N}$

MIRADAS. Its worst-case scenario is that presenting three points separated all of them by a distance equal to the MIRADAS FOV diameter, as shown in fig. 4.3a. To guarantee, even in such case, that all points in a cluster are contained in a surface with the area of the real FOV, the cut-off threshold should be equal to the side of the equilateral triangle inscribed in a circle with a diameter equal to the FOV diameter; see fig. 4.3b.

4.3.2 Smallest Bounding Circle for a Set of Points

Determining the circle of smallest radius enclosing a set of points is a classical optimization problem of computational geometry. Since at most three points uniquely determine a circle, a brute force approach can be attempted. It will consist in checking every circle defined by three and two points and keeping the minimum-sized disk containing all points of the set. This is by far the worst solution. There exist $O(n^3)$ circles, being $O(n)$ the time required to check each of them. Consequently, the total execution complexity of this approach is $O(n^4)$.

Although alternative methods can be found in literature, it was Welzl the first proposing a fast and easily implementable algorithm for 2D (Welzl, 1991). His solution was inspired by a previous work solving a linear program with n constraints and d variables in *expected* $O(n)$ time, provided d is constant. Welzl relied on a randomized incremental construction, nowadays a powerful



(a) Three points separated by a distance equal to the FOV diameter. The radius of the circumscribing circle about the triangle is larger than the FOV diameter.

(b) An equilateral triangle inscribed in a circle with radius equal to the instrument FOV radius. The side of such a triangle determines the cut-off threshold of the RNN algorithm.

Figure 4.3: Inscribed triangles to determine the RNN cut-off threshold.

technique widely employed to efficiently solve optimization problems found in computational geometry. Although that way of constructing a solution provides a useful framework to address the problem at hand, it comes at a cost. The running time of $O(n)$ is only an *expected* bound that might be larger for some data sets. Generally, this technique applies to those optimization problems where: (i) the solution does not change if a new constraint is added or (ii) the solution is partially defined by the new constraint so that the dimension of the problem is reduced (Berg et al., 2008). Both conditions are met in the smallest enclosing disk problem.

Given a set of n points $P_n = \{p_1, p_2, \ldots, p_n\}$, the Welzl approach iteratively selects one point and grows the smallest bounding disk that contains that point and the previously processed. The algorithm is based on the observation that when the smallest disk D_i enclosing the subset $P_i = \{p_1, p_2, \ldots, p_i\}$, where $1 \leq i \leq n$, is updated to include another point p_{i+1}, then the new optimal disk $D_{i+1} = D_i$ if the point belongs to D_i. Otherwise, D_{i+1} must pass through p_{i+1}. In the former case, the algorithm directly proceeds to the next point while in the latter a new method computing the smallest bounding circle of a set of points P with a given point p_i on its boundary is required. Relying on this additional method that we will call `minidiskWithSupport`(P, p_i), Welzl incrementally computes the optimal disk for the initial data set.

The algorithm for `minidiskWithSupport`(P, p_i) follows a similar iterative approach. It sequentially checks the points of the set for inclusion, calling, when necessary, a subroutine determining the smallest enclosing circle of a set of points with two given points in its boundary. This new subroutine is also constructed using the same framework, but this time it will require another subroutine computing a bounding disk of a set of points with 3 points in its perimeter. As a circle is determined by at most three points, there is no need to expand further

Algorithm 2 Smallest enclosing disk

Require:
$\mathcal{P}$: a set with all points to be enclosed
$\mathcal{R}$: a set with all points that must be on the boundary of the smallest circle
Ensure:
$\mathcal{D}$: The smallest circle enclosing the set $\mathcal{P}$

```
1: function MINIDISKWITHSUPPORT(P, R)
2:     if P ≠ ∅ or |R| = 3 then
3:         D ← BUILDCIRCLE(R)    ▷ Construct a circle with the given points in
   its boundary
4:     else
5:         select p ∈ P randomly
6:         D ← MINIDISKWITHSUPPORT(P\p, R)
7:         if p ∉ D then                    ▷ point p does not lie inside D
8:             D ← MINIDISKWITHSUPPORT(P\p, R ∪ p)
9:         end if
10:    end if
11:    return D
12: end function
```

the search. Although this scheme of nesting routines is quite intuitive, a more compact recursive solution can be obtained relying on a generalized version of `minidiskWithSupport(`P`,`R`)`. This version determines the smallest disk containing the points in P with the points in R on its perimeter; refer to Welzl (1991) for proofs. The pseudo-code for the recursive approach is shown in algorithm 2. The smallest enclosing disk for a set of points P can be solved by the call `minidiskWithSupport(`P`,`$\varnothing$`)`.

4.3.3 Simulations and Results

The proposed threshold-based RNN clustering algorithm for *complete-linkage* as well as the minimum bounding disk, respectively described in section 4.3.1 and in section 4.3.2, were implemented in C++. Several simulations with synthetic and real science targets were conducted to study the performance of our method. First of all, we will focus on random scenarios, which will help us to show the reader what we want to accomplish in MIRADAS by employing the proposed algorithms. In section 4.3.3.2, our solution is tested with real celestial objects.

4.3.3.1 Synthetic Scenarios

In the first of the two synthetic scenarios, a total of 20 points were arbitrarily arranged inside a square of side 50 mm, as seen in fig. 4.4a. Then, the clustering algorithm with a threshold value of 25 mm (the FOV diameter for this example) is run to detect clusters automatically. Once known these clusters and the points included in each of them, the smallest enclosing circle is determined for every cluster. The center of this circle will be used as telescope's pointing information. Also,

Table 4.1: *Intra-cluster* and *inter-cluster* distances for the example in Figure 4.4a

(a)

	Intra-Cluster dist. (mm.)	
Cluster	min.	max.
1	3.1622	19.9249
2	6.4031	20.2237
3	6.4101	17.088
4	4.1231	4.1231
5	7.8102	18.4391

(b)

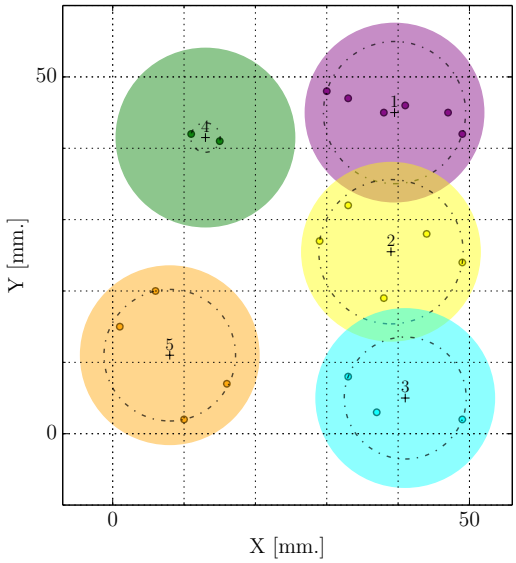
	Inter-Cluster distance (mm.)				
	cluster 1	cluster 2	cluster 3	cluster 4	col5
cluster 1	…	30.6105	49.7695	30.2655	36.8782
cluster 2	30.6105	…	34	42.0476	29.5466
cluster 3	49.7695	34	…	55.1725	46.6154
cluster 4	30.2655	42.0476	55.1725	…	28.7924
cluster 5	36.8782	29.5466	46.6154	28.7924	…

the members of each cluster will be delivered to a different instance of the *target allocator*, discussed in chapter 5. Intra-cluster and inter-cluster distances for this example are given in table 4.1. The former refers to distances between elements in a cluster, while the latter relates to the distance between two clusters, which is the distance between their two farthest-apart members. As expected, the dissimilarity (distance) between any pair of members of one cluster is inferior to the specified threshold while the dissimilarity between clusters remains greater than the threshold. In addition, the results show how the algorithm generates compact groups. That is due to the tendency of the selected linkage scheme to combine those cluster pairs whose merge has the smallest diameter. Finally, we plot the full dendrogram, given in fig. 4.5, for this scenario. As seen there, if the tree is properly cut at a height equal to 25, we obtain the 5 clusters found in fig. 4.4a.

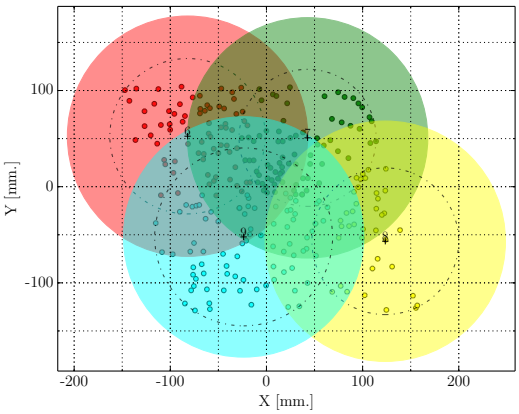
A second random scenario, but this time with 270 targets more densely distributed was generated. Clustering with a cut-off threshold of 250 mm was performed. As can be appreciated in fig. 4.4b, four clusters were needed to contain all points in the input data set. For some clusters, the corresponding circles that represent the FOV overlap each other. That means that targets in the intersection area of two or more FOV, although belonging to different clusters, can be delivered to two or more distinct instances of the target allocator. This degree of flexibility might be conveniently exploited when trying to compute safe trajectories for the targets. In fact, as we will see, this superposition is normal when clustering real science targets since they are frequently arranged in a quite compact surface.

4.3.3.2 Real Data Scenarios

We have also performed a series of tests with real data. They contain groups of celestial objects that, according to the MIRADAS science cases, can be targeted by the instrument. These targets were determined initially in equatorial coordinates. To be represented in the surface of the MIRADAS FOV, they were previously converted into standard coordinates using the MIRADAS plate scale. Since each input data set does not implicitly specify the center/reference of the targets to observe, we considered it to be the mean value of the samples, as explained in section 4.1.2.1. Once known the ξ and η values, the X- and Y-axes respectively, the data was delivered to the clustering algorithm. In fig. 4.6a, the cluster analysis for several objects belonging to Messier

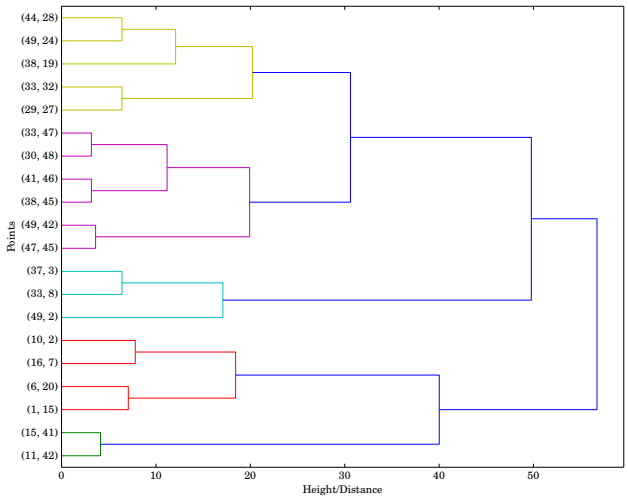


(a) Five different clusters, each comprising the small dots of the same color. For each of the clusters, we show its center (black cross), the smallest enclosing disks (dashed circle) used to compute the center as well as a larger colored circle centered at the corresponding cross with a radius equal to *threshold*/2 (12.5mm). The numbers above the cluster centers specify the cluster name.



(b) Four overlapping clusters (in bold colors) were required to completely cover the 270 targets in the figure. In fact, many targets that were initially assigned by the algorithm to a particular cluster lay inside more than one *threshold* circle (in bold colors).

Figure 4.4: Synthetic data scenarios.



(a) The tree has as many leaves as points are in the scenario and they are merged up to the root, where all the initial points are members of a single cluster. A cut at a distance of 25 gives the 5 clusters automatically obtained by our algorithm.

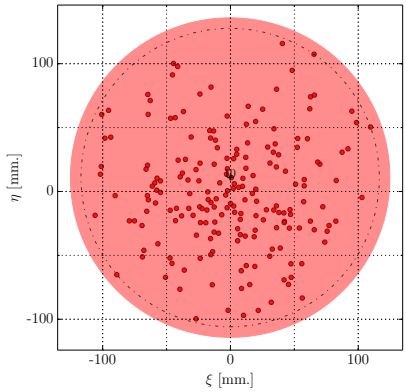
Figure 4.5: Dendrogram for the clustering example in Figure 4.4a.

13 (M13) is given. All targets fit in the MIRADAS FOV area; therefore, a single cluster, with its center at the denser zone of the data set, was found.

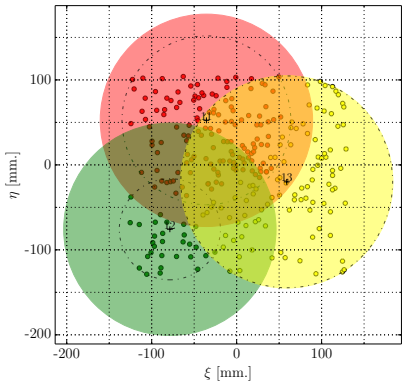
A different scenario is shown in fig. 4.6b. This time targets represent a group of objects in Czernik 3. The clustering algorithm returns more than one cluster since the targets occupy an area slightly bigger than the MIRADAS FOV. Note that these clusters present bias. That means that the members of a cluster can be concentrated in a peripheral zone of the FOV. That is the case of the red cluster, where its components are mainly concentrated in the center and toward the inferior semicircle of the FOV. This bias, if necessary, could be mitigated by adequately rebalancing the load of those clusters presenting overlapping areas or, alternatively, by appropriately shifting the center of the cluster. The effect of bias tends to be more pronounced in scenarios with a larger number of targets. Figure 4.6c shows 1354 objects from Sagittarius A (Sgr A), all densely distributed in a surface moderately bigger than the MIRADAS FOV. The corresponding circles associated with each cluster are given in fig. 4.6d. Note how now the clusters are more significantly biased than in the previously analyzed cases.

Finally, we retrieved two more sparsely distributed sets of sky objects from the UKIRT Infrared Deep Sky Survey (UKIDSS)[7]. The first of them contains 2000 selected targets from Sgr A (~0.5×0.5 deg) and the second the same number of targets but this time from a particular

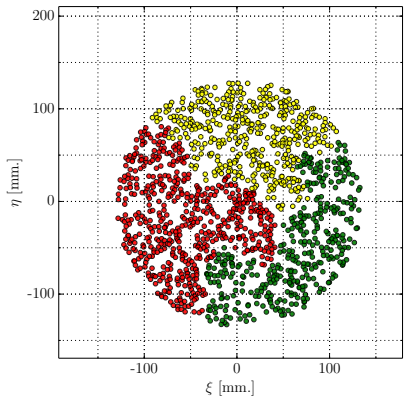
[7] http://www.ukidss.org



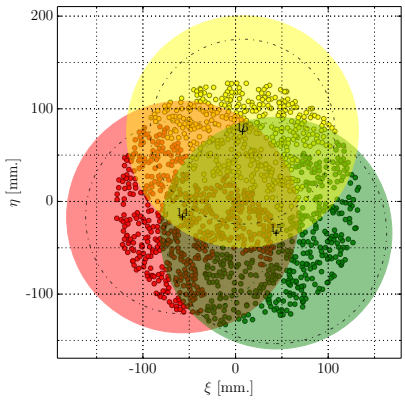
(a) Sky objects (dots) from M13 occupying and area smaller than the MIRADAS FOV (red circle). The algorithm groups all points in a single cluster centered at the black cross.



(b) Targets from Czernik 3 occupying a wider area than the MIRADAS FOV. Here, the agglomerative algorithm returns 3 clusters. The members of the green one are grouped around the cluster center, while those in the yellow cluster are spread over the whole FOV surface. Note that the bounding disk of the yellow cluster has the same diameter as the FOV.

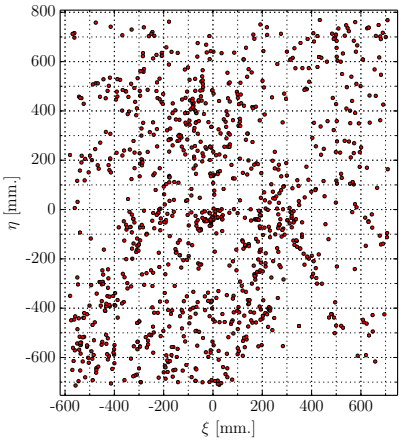


(c) More than 1300 targets from Sgr A occupying a circular area slightly wider than the MIRADAS FOV. For this compact scenario, the clustering algorithm returns three different sets. The dots with the same color belong to the same cluster.

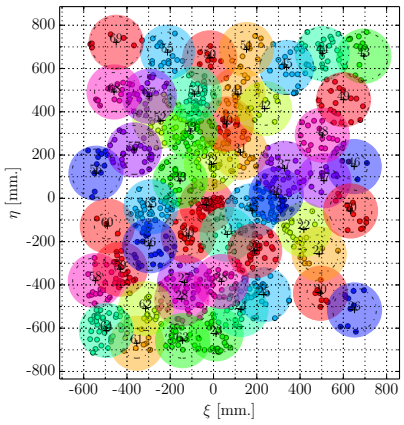


(d) The same points found in fig. 4.6c, but now the circles specifying the FOV for each cluster are also shown. As can be appreciated, in each cluster there is a bias. The members are not uniformly distributed, leaving a considerable region of the respective FOVs empty.

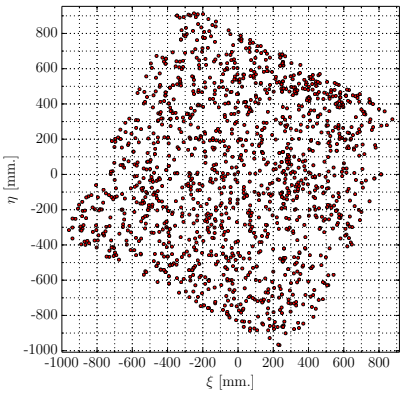
Figure 4.6: Clustering results for targets from M13, Czernik 3 and Sgr A.



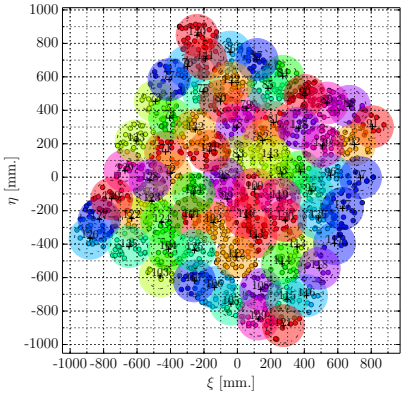
(a) 2000 targets from a wider area of SgrA.



(b) Clusters obtained from the 2000 targets shown in fig. 4.7a.



(c) 2000 targets from a particular region of the Scutum-Centaurus arm.



(d) Clusters obtained from the 2000 targets shown in fig. 4.7c.

Figure 4.7: Clustering results for scattered fields belonging to portions of Sgr A and Scutum-Centaurus arm.

portion of the Scutum-Centaurus arm. The celestial targets, as well as the resulting clusters, are shown in fig. 4.7. In scenarios of this kind, bias can also occur but, since the points of interest are scattered over a wider area, it is not as severe as in denser scenarios. Overlapping is also present, but, as expected, superposed regions enclose few points.

4.4 Conclusion

In this chapter, we looked at how astronomers define the locations of celestial bodies in space as well as how they can be properly transformed into coordinates in the FOV of an astronomical instrument. We also introduced several analysis techniques to partition a data set into distinct subsets. Despite being methods broadly employed in many areas, including the analysis of astronomical surveys, we believe that only a reduced number of instruments require them to prepare observations. Indeed, the authors have been only capable of identifying one MOS needing field segmentation. We are referring to EMIR, a multi-slit based spectrograph installed at GTC a few years ago.

Designed as a GTC common-user instrument for many years in the future, MIRADAS should be capable of working with a large variety of celestial fields. Targets will be mainly selected considering their sky locations, yielding, in many cases, fields of scattered objects not fitting in the FOV of the instrument. In this chapter, we demonstrated how fields of this kind can be arranged to be conveniently observed by MIRADAS. Specifically, a preprocessing step based on cluster analysis run before the target allocator was proposed. In this step, user-defined objects were combined so that all those in the same group are contained in a circular area smaller than a given FOV. That was achieved by employing a hierarchical agglomerative clustering algorithm returning also the center of each cluster, which can be used as telescope's pointing information. This technique successfully partitioned the data sets used in several tests. Also, it created some overlapping regions that might help a higher-level logic (or scheduler) to redistribute load among different clusters. However, our solution shows a tendency toward clusters whose members are not homogeneously arranged. This occurs especially in those clusters located in the remotest areas of the field. Thus, future work should include the exploration of: (i) alternative clustering techniques and (ii) alternative ways of modeling the center of each cluster so that the load of each group is distributed evenly.

CHAPTER 5

TARGET ALLOCATION

Fundamentally, target allocation in MIRADAS is a combinatorial optimization problem (COP). In applied mathematics and theoretical computer science, COP is a lively field integrating techniques from combinatorics, linear programming and theory of algorithms (Cook et al., 1998). An optimization problem is often expressed employing a particular mathematical representation of a decision process such that a piece of software can effectively solve it. Besides, these problems are constructed around decision variables, representing the decisions to be taken, and constraints, restrictions on the values that can be given to the variables. This representation process is commonly known as modeling the problem, and it is probably the most crucial task when addressing a COP.

Combinatorial problems can be found in many areas of knowledge such as management, operations research and robotics, and arise in a variety of situations as well. For instance, the Traveling Salesman Problem (TSP) is a well-known COP consisting in finding the shortest round trip such that n customers are visited exactly once (Lawler et al., 1985). All COPs have in common that the decision variables characterizing the respective models have discrete and finite domains. Hence, these problems have a finite number of potential solutions. A solution is an assignment of all n decision variables, while the search space is the set of all possible assignments to these variables. This search space is typically exponential in the number of variables. However, fortunately, due to the constraints affecting the problems, many combinations are infeasible.

Like in any other assignment problem, in MIRADAS there is a set of *workers* -arms- for performing a set of *jobs* -target observations. Considering the two sets involved and some particular constraints, then we need to determine how to assign workers to jobs. Due to its underlying combinatorial structure, the target allocation problem might have more than one feasible solution satisfying the constraints. While for general search or decision problems is

enough to find any of them, here we want to determine the best (or optimal) one. The goodness of a solution with respect to the others is evaluated with the help of a performance metric, often referred to as objective function.

Parts of this chapter have previously been published as: "Target allocation and prioritized motion planning for MIRADAS probe arms", Proc. SPIE 9913: Software and Cyberinfrastructure for Astronomy IV, 99132P, 2016 (Sabater et al., 2016).

5.1 Problem Statement

In MIRADAS, each celestial target comes specified by two different elements: its position on the sky and a priority (user-defined integer value) showing the importance of the target. One remarkable difference between MIRADAS an other MOS is the nature of the priority field. To the best of our knowledge, observation preparation tools for instruments of this kind work always with a prefixed set of priorities. That is, the range of values that the priority field can hold is very restricted. Typically, instruments support three levels of priority. Such is the case in EMIR, a cryogenic multi-slit MOS in operation at GTC (Garzón et al., 2006, 2014). Others offer the user a little bit larger range. To name one, in the Two-degree Field (2dF), a positioning system specially designed for fibers by the Australian Astronomical Observatory (AAO), nine priority levels are available (Lewis et al., 2002). In contrast, priority in MIRADAS can hold any natural number that can be represented using 32 bits. Therefore, the values are practically unlimited, leading that way to very diverse priority policies. In fact, the user could choose from assigning to every target the same priority to assigning a different priority to every target.

Since a particular sky position can be reached by more than one probe in the system, it is crucial to obtain a rational assignment of targets to probes, which requires solving a COP. Broadly speaking, given N targets to be observed, the MIRADAS target allocator will have to determine a total of N pairs <target, arm>. As the number of targets of interest is often higher than the number of arms in service in the MXS system, not all targets could be observed at once. Therefore, more than one *observational configuration*, considered as the different exposures needed to acquire all targets belonging to the same telescope pointing, might be required. In this context, then, each observational configuration can be thought of as an iteration to any of the particular target allocation algorithms proposed in the following sections. First, an observational configuration is determined by the algorithm. If not all science targets of interest could be fitted in this initial configuration, then another one is planned considering the remaining targets. This process is repeated until all targets have been assigned to a probe arm. Consequently, the output of the allocation process can be a single observational configuration or a sequence of them.

Ideally, this allocation process will deliver a set of observational configurations, where, in each of them a maximum of 12 science targets will be observed simultaneously. However, in practice, maximum efficiency in terms of arms successfully allocated could not always be achieved.

Moreover, depending on the precise needs, several different objective functions can be optimized when planning these allocations. In particular, the elemental decision criterion to be followed by the MIRADAS target allocator is:

- To group targets in observational configurations in such a way that the sum over all available targets priority is maximized.

In other words, for a given set of targets, if only one observational configuration should be determined, this would contain the 12 targets whose sum of priorities is the greatest. Finally, in MIRADAS, three other criteria are desirable but not required. These are:

(a) **High efficiency**: maximize the number of arms assigned in each observational configuration.

(b) **High priority to observational configuration ratio**: minimize the number of observational configurations needed to observe the highest priority targets.

(c) **Path-aware**: favor those assignments leading to shorter potential paths with the hope of avoiding collisions among probes.

The length of an arm path is estimated by the relative displacement of the two actuated probe joints:

$$d = |\theta_1^f - \theta_1^i| + |\theta_2^f - \theta_2^i| \tag{5.1}$$

where θ_1^f and θ_1^i represent the final position (the one returned by the allocation algorithm) and the initial one for the joint J_1. Furthermore, the parameters θ_2^f and θ_2^i respectively denote the final and the initial position for the joint J_2. The higher the value of d, the longer a path is considered. Therefore, an assignment plan u, in terms of path distance, is said to be more convenient than another assignment v, if the sum of all arm path lengths in u is inferior to the sum in v.

An observational configuration includes the following information: (i) a natural number indicating the sum of the priorities of the targets in the configuration and (ii) as many assignments as probes are in the MXS system. An assignment is expressed through a triplet consisting of a target, an arm, and the configuration q of the arm positioning the tip on the corresponding target. An observational configuration always provides a static picture of the state of the MXS system, especially of the placements of all arms. Since there is an area of the arm workspace that can be reached with two different arm configurations, as discussed in section 3.3, the value of q is necessary to determine the arm location unambiguously. In fact, if an arm is broken, it has to be also present in a configuration. In this case, the value of q will indicate the current arm location. Finally, if for any other reason, the allocator could not assign a working arm to a real target, the arm will rest in its actual position or be assigned to any other blank sky position.

Taking into account all the previous considerations, we can say that the MIRADAS target allocation problem consists in planning a set of observational configurations maximizing in each

of them the aggregated priority. Besides, in each configuration, a set of hard constraints has to be satisfied as well:

(a) An arm is assigned to one and only one location.

(b) A given FOV point is only assigned to one arm.

(c) The arms of the system do not collide to each other.

5.2 Related Work

There is an impressive body of literature about assignment problems related to logistics, operations, and other similar areas of study. Besides, the problem has been approached following a vast number of methods. The interested reader is referred to Korte and Vygen (2012) for a comprehensive discussion. However, it is worth mentioning that, in some particular cases, the problem can be formulated using a linear model and, consequently, integer or mixed integer linear programming can be used to determine a solution (Schrijver, 1986). In contrast, in others not showing that linear nature, metaheuristics are of great help (Gendreau and Potvin, 2010). Regarding astronomy, this sort of problems frequently appears in modern MOS. In these instruments, there exist a limited number of light-collecting resources -fibers, slits, or probe arms- to observe a greater number of celestial bodies. Therefore, some arrangement -an allocation plan- needs to be determined to acquire all the desired sky targets properly. We present here an overview of the most important works in the field of astronomical instrumentation.

Lately, a new concept of positioners has been developed for fiber-based MOS. These positioners, named *star bugs*, consist of a pair of concentric piezo-ceramic actuators carrying an optical fiber payload. The *star bugs* are adhered via vacuum force to a glass plate and rotate and translate over the field-of-view of the instrument, positioning the fibers onto the desired locations. TAIPAN, a 150-fiber spectrograph for 1.24-m UK Schmidt Telescope (UKST) at Siding Spring Observatory (Australia), is the first instrument equipped with this kind of mini mobile robots (Kuehn et al., 2014; Staszak et al., 2016). The MOS configuration software of this instrument has two different allocation algorithms: Closest Distance Allocator (CDA) and Advanced Version of CDA (AVCDA) (Goodwin et al., 2014; Lorente et al., 2016). Both methods are iterative in nature. Specifically, CDA assigns to each star bug the nearest target to its home position (called *park* position) that is in its range of motion. Those positioners not assigned to their nearest targets are considered in later executions (revisits). This approach shows performance similar to that of an ideal positioner for field completeness inferior to 70%. However, the algorithm does not consider the use of target priorities. Moreover, it assumes that the bugs always start their trips from their park positions. And finally, the assignments delivered might lead to robot collisions if they follow direct paths, those connecting the corresponding initial and destination points. On the other hand, AVCDA considers priorities and attempts to avoid those allocations leading to intersecting direct paths.

In addition, unlike the basic CDA, it also works with robots whose starting locations are different from their respective park positions. This advanced variant generates theoretical longer paths and possesses poorer completeness than the basic CDA.

MANIFEST, an under construction fiber feed and positioner for a series of Giant Magellan Telescope (GMT), will have about 500 starbugs operating in parallel spectrographs (Lawrence et al., 2018, 2014a). Designed to have seven 8.4m diameter primary segments, this extremely large telescope will be operated at Las Campanas Observatory (Chile) and is planned for completion in 2025. It is expected to have a configuration system similar to the one present in TAIPAN.

Other instruments utilize metaheuristics. The 2dF positioning system essentially consists in a *pick and place* robotic system placing 392 fibers at the locations previously determined by the field configuration algorithm. This algorithm is based on the *simulated annealing* technique (Miszalski et al., 2006). The allocator pre-calculates all potential conflicts for a particular field and stores them in a collision matrix. This data structure enables faster calculations, especially if the user wants to determine distinct assignments for the same field considering different values of the algorithm parameters. The positioner of WEAVE, a fiber-based MOS survey spectrograph for 4.2-m William Herschel Telescope (WHT) at the Observatorio del Roque de los Muchachos (Spain), is conceptually similar to the one in 2dF. It comprises two positioning robotic arms that configure the field previously allocated by a target assignment tool resembling the one used in 2dF. Sadly, adding the concept of target priority in this particular metaheuristic is challenging. In Miszalski et al. (2006), an attempt is done in that direction[1] by tuning the temperature parameters - T_i, ΔT and T_f - of the annealing algorithm. Finally, LAMOST, a multi-object fiber spectroscopic telescope located at Xinglong Station (China) also employs simulated annealing to extend and enhance the draining algorithm (Yuan et al., 2013).

The draining allocation algorithm was conceived for fiber-based large survey spectrographs (Morales et al., 2012). The aim of this naturally rebalancing method was to increase the fraction of targets observed in the same number of configurations (or tiles). For each positioner, it creates a different list containing the targets that can be reached by that particular fiber. In those cases where a target can be reached by more than one positioner, the target is only pushed into the list containing fewer elements. Although the results were more promising than those obtained from a random allocation, a point to consider is that the algorithm does not check if fibers collide with each other once assigned. Based on the draining concept and the positioner of MOONS, a fiber-based MOS for ESO's 8.2-m VLT located at Observatorio Paranal (Chile), a slightly different solution is proposed in Schaefer et al. (2016). Instead of allocating targets to positioners, this time, positioners are allocated to targets. This algorithm determines the pairings iteratively, but now checking if a potential target-fiber assignment collides with those previously determined. One crucial difference between the two drain-based approaches is that the one utilized in MOONS returns an allocation plan for only one tile. Thus, it has to be run multiple times to prepare

[1] Only 9 levels were considered since that were the requirements for 2dF.

consecutive tiles. Finally, although a hint is given in this last work, neither of both variants considers the priority[2] of the targets in the allocation process.

Before closing this section, we will dedicate a few lines to KMOS, a MOS for ESO's VLT (Sharples et al., 2010). Although KMOS and MIRADAS employ deployable probe arms distributed around a circular bench to observe several user-defined targets simultaneously, they differ technically. For instance, as we will see, in the target allocation phase. In KARMA, the observation preparation tool for KMOS, the astronomer can prioritize sky targets using three distinct levels (Wegner and Muschielok, 2008). Contrarily, in MIRADAS the values the user can employ in the priority field are not restricted to a predefined set. This is a consequence of the target allocation and the motion planning steps being decoupled in MIRADAS, that seems not to be the case for KMOS.

KARMA offers two different choices to prepare the field to observe. The first uses the Hungarian algorithm, an optimization method solving the fundamental assignment problem in polynomial time (Kuhn, 1955). The cost associated with each pairing corresponds to the degree of motion required for each arm to reach a given target. This condition would naturally meet the constraint of collision-free assignments if arms were dimensionless. Therefore, as that is not realistic, a collision checking phase is performed afterward, discarding those conflicting assignments. As it will be commented in section 5.4, although the Hungarian algorithm is efficient, it is only adequate in those cases where the number of jobs (targets) is equal to the number of workers (arms). Hence, its use imposes in some way that the set of targets to be allocated should be successively partitioned in batches of size equal to the number of arms to consider. This might result in a suboptimal sequence of targets exploration. As the authors assert, the Hungarian method is best suited for equally-weighted targets. For fields showing heterogeneous priorities, the second option offered by the KARMA tool is preferred. This relies on an algorithm solving the stable marriage problem (Gale and Shapley, 1962). In this approach, in addition to the collision-free constraint for dimensionless arms, priority is also taken into account. However, the technique proposed is far from being optimal. What is more, the subsequent collision-checking phase is still required.

Finally, a remarkable characteristic of KMOS is that its 24 arms are evenly distributed in two different planes, one above the instrument FOV and other below. Due to this particular arrangement, eclipses of lower plane IFU's might arise if the upper arms are not appropriately positioned. These awkward situations can be easily avoided by the assignment of lower plane arms to targets in the FOV center, whereas upper plane arms are allocated to more outward targets. This leads to the usage of a two-stage assignment method, which allocates the bottom layer arms in a first run and the top layer arms in a second one.

[2] In MOONS there are only four prefixed values, one for each priority level.

5.3 A Metaheuristic for MIRADAS

In many industrial and scientific applications, finding an optimal solution becomes intractable. However, in most cases, it is not necessary to determine the best possible solution since we will be satisfied with an acceptable one. Heuristics and metaheuristics are approximate techniques, which means that they compute a good solution in a reasonable time. The popularity of the second of them has significantly increased since it provides a template that if suitably adapted, helps to the design of underlying heuristics.

In this section, we present a target allocator based on a frequently employed metaheuristic that, given a list of user-defined targets and considering the objectives commented in section 5.1, returns an observational configuration. Our method requires that the targets of interest fit inside the area delimited by the instrument FOV. Besides, it is also necessary to provide them with the telescope pointing information. Therefore, before delivering a set of targets to our algorithm, the targets should be processed by a field segmentation algorithm like the one presented in section 4.3. Then, once the different clusters are known, each can be sequentially explored by calling the method proposed here. This task can be easily parallelized by running different instances of the allocation process concurrently, each of them in charge of a distinct group of targets.

5.3.1 Semi-Greedy Patrol-Based Target Allocator

As its name indicates, the Semi-Greedy Patrol-Based Target Allocator (SGPBTA) applies a *semi-greedy* strategy to try to find a global optimum. The proposed allocator is inspired by the Greedy Randomized Adaptive Search Procedure (GRASP), a metaheuristic algorithm for combinatorial optimization problems (Resende and Ribeiro, 2010, 2016). GRASP extends the traditional greedy approach by incorporating randomness to allow this way some degree of diversification. Moreover, GRASP also includes a local search phase with the aim of intensifying the exploration of given neighborhoods.

Greedy algorithms are constructive in nature, iteratively building a solution. They progress in a top-down fashion, making step by step choices and consequently reducing in each iteration the given problem to a smaller one. In each of these iterations, it is selected from all available choices the one looking best at the moment. In other words, they decide what to do next by grabbing the most promising local option hoping that it will lead to a global maximum. However, if the problem to be solved does not exhibit optimal substructure, this approach of focusing only on the most immediate benefit might not yield an optimum (Cormen et al., 2009). But, in many cases, it produces a good enough solution. On the other hand, semi-greedy schemes, although greedy in essence, they rely on an additional stage that incorporates some sort of stochastic behavior in the selection process (Hart and Shogan, 1987). In each step of the solution construction, a limited pool of candidates (LPC) is built from all available choices. The components of this pool are determined according to a greedy evaluation function, which measures the relative cost (or

benefit) of individually including each particular element into the solution under construction. Then, one of the individuals in the LPC is randomly picked and incorporated to the partial solution. As we will see, this selection technique enables obtaining different solutions at each iteration of the process.

Algorithm 3 presents the pseudo-code for the SGPBTA heuristic. It is based on a multi-start process comprising three distinct phases. In the first of them, a solution to the problem is constructed following a *target-oriented* semi-greedy approach. This solution might be incomplete if the algorithm has not managed to discover a feasible allocation for each arm in the MXS bench. If such is the case, the second phase tries to complete it, assigning the unallocated arms to blank sky. Unfortunately, this solution is not guaranteed to be locally optimal for a simple neighborhood definition. Therefore, a local search is then carried out to attempt to improve the constructed solution. Finally, the solution is analyzed, and the best overall from all iterations is kept as a result. A total of `n_iter` executions are run, each employing the same value for the parameter `sd` representing the initial seed for the pseudo-random number generator.

Algorithm 3 Semi-Greedy Patrol-Based Target Allocation

Require:
$\mathcal{T}^i$: a set of targets in a cluster
$\mathcal{P}$: telescope pointing information for the cluster
$\mathcal{C}$: the cost/penalty matrix for all arms and SoPs
n_iter : number of iterations
sd : seed for random selections
Ensure:
$\mathcal{S}$: a set with *arm-target* assignments
$\mathcal{T}^o$: the set of targets not allocated

1: **function** SGPBTA($\mathcal{T}^i$, $\mathcal{P}$, $\mathcal{C}$, n_iter, sd)
2: $\mathcal{T}^o \leftarrow \mathcal{T}^i$, $\mathcal{S} \leftarrow \varnothing$
3: $\mathcal{K} \leftarrow \{1, \ldots, n\_iter\}$
4: **for all** $k \in \mathcal{K}$ **do**
5: $\mathcal{S}_k$, $\mathcal{T}_k^o \leftarrow$ SGPB-Construction($\mathcal{T}^i$, $\mathcal{P}$, $\mathcal{C}$, sd)
6: **if** $|\mathcal{S}_k| < 12$ **then**
7: $\mathcal{S}_k \leftarrow$ Solution-Padding($\mathcal{S}_k$, sd)
8: **end if**
9: **if** $|\mathcal{S}_k| = 12$ **then**
10: $\mathcal{S}_k$, $\mathcal{T}_k^o \leftarrow$ Local-Search($\mathcal{S}_k$, $\mathcal{T}_k^o$)
11: $\mathcal{S}$, $\mathcal{T}^o \leftarrow$ Solution-Update($\mathcal{S}_k$, $\mathcal{T}_k^o$, $\mathcal{S}$, $\mathcal{T}^o$)
12: **end if**
13: **end for**
14: **return** $\mathcal{S}$, $\mathcal{T}^o$
15: **end function**

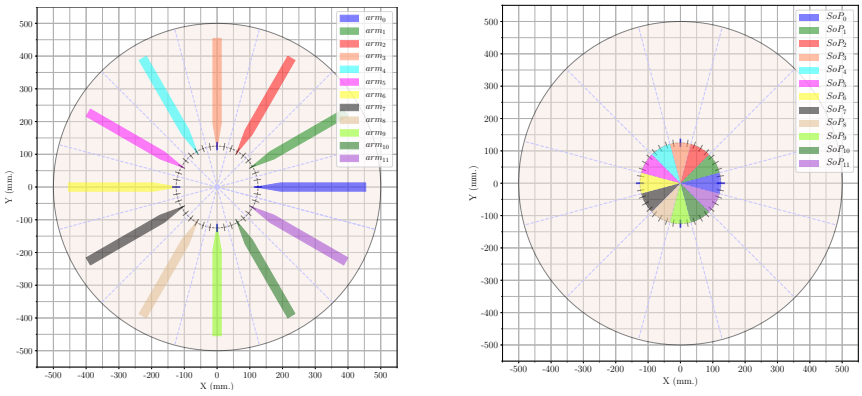
5.3.1.1 Construction Phase

In every iteration of the construction phase the targets awaiting allocation are considered and a LPC is built with those having greater benefit values. Afterward, an ordered list containing all not previously assigned arms reaching the selected target is built and explored. The position of a given arm in the list comes determined by the $m \times n$ cost matrix $\mathbf{C}$, where m represents the number of probe arms in the MXS system and n the number of SoPs in the FOV. Each element $\mathbf{c_{ij}}$ contains an integer indicating a penalty associated to the pair $< i, j >$, where the index i and j refer to the i^{th} arm and the j^{th} SoP respectively. The higher the value, the higher the penalty and, consequently, the farther from the origin of the list an arm will be. Principally, this matrix is used to prioritize some target-arm assignments over others. Typically, it is square. But, being the SoP an abstract construction to reduce the effective range of motions of an arm, nothing is preventing us from having a number of SoPs lower than the actual number of arms.

The penalty matrix can also be utilized to effectively limit the patrol range of every arm in the system. Implementing a SoP strategy, as commented in section 3.7.2, is as simple as defining a matrix where all elements of its main diagonal are 0. For instance, the following 12×12 matrix shows that each arm can only patrol its associated SoP (see fig. 5.1):

$$\mathbf{C} = \begin{bmatrix} 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix} \tag{5.2}$$

All negative components of the matrix in eq. (5.2) will be skipped by the algorithm since they indicate that the points of the FOV belonging to those SoPs cannot be patrolled by a given arm. On the other hand, hybrid patrolling schemes can be straightforwardly implemented as well. By way of example, the following sample matrix can be useful if we want to allow an arm to patrol more than one SoP and, at the same time, to give to each of them different selection-preferences:



(a) Arms numbering (b) SoP numbering

Figure 5.1: The SoPs employed in the SGPB Target Allocator and the arms of the system numbered.

$$\mathbf{C} = \begin{bmatrix} 0 & 10 & 20 & 30 & -1 & -1 & -1 & -1 & -1 & 35 & 25 & 15 \\ 15 & 0 & 10 & 20 & 30 & -1 & -1 & -1 & -1 & -1 & 35 & 25 \\ 25 & 15 & 0 & 10 & 20 & 30 & -1 & -1 & -1 & -1 & -1 & 35 \\ 35 & 25 & 15 & 0 & 10 & 20 & 30 & -1 & -1 & -1 & -1 & -1 \\ -1 & 35 & 25 & 15 & 0 & 10 & 20 & 30 & -1 & -1 & -1 & -1 \\ -1 & -1 & 35 & 25 & 15 & 0 & 10 & 20 & 30 & -1 & -1 & -1 \\ -1 & -1 & -1 & 35 & 25 & 15 & 0 & 10 & 20 & 30 & -1 & -1 \\ -1 & -1 & -1 & -1 & 35 & 25 & 15 & 0 & 10 & 20 & 30 & -1 \\ -1 & -1 & -1 & -1 & -1 & 35 & 25 & 15 & 0 & 10 & 20 & 30 \\ 30 & -1 & -1 & -1 & -1 & -1 & 35 & 25 & 15 & 0 & 10 & 20 \\ 20 & 30 & -1 & -1 & -1 & -1 & -1 & 35 & 25 & 15 & 0 & 10 \\ 10 & 20 & 30 & -1 & -1 & -1 & -1 & -1 & 35 & 25 & 15 & 0 \end{bmatrix} \tag{5.3}$$

The first row of the above matrix contains the patrolling preferences of arm_0. In order, the allocator will try to assign this arm to SoP 0, 1, 11, 2, 10, 3, 9. Just as a reminder, the lower the value (penalty) of a matrix element, the higher its patrolling preference. The other SoPs are forbidden since they contain negative values.

The pseudo-code of the construction function `SGPB-Construction` is illustrated in algorithm 4. In the `Require` and `Ensure` sections are specified the different sets and information required by the function to work as well as the data it returns. Among the different parameters, we find the set of input and output targets, represented by $\mathcal{T}^i$ and $\mathcal{T}^o$ respectively. The former contains the

targets that need to be allocated, whereas the latter, which is populated during the execution of the construction method, will have those targets that were not successfully assigned. We denote by $\mathscr{A}^a$ the set of all assigned arms, which is initialized in line 2. In addition, the set $\mathscr{A}^c$ represents the arm candidates to be considered for a given target. This set is initialized in line 9. The algorithm uses the function `Cost(C,a,s)` returning the cost or penalty associated to arm a for accessing the SoP s. The loop comprising lines 4-21 is performed while there are targets remaining and not all 12 arms available in the system have been successfully allocated. A semi-greedy approach is applied in line 5-6 to pick the target to explore by first building an LPC and then selecting an element randomly. On the other hand, a probabilistic selection or a pure greedy one (note that the *pseudo* prefix does not appear here) could be carried out in line 11. In

Algorithm 4 Semi-Greedy Patrol-Based Construction

Require:
$\mathcal{T}^i$: a set of targets in a cluster
$\mathcal{P}$: telescope pointing information for the cluster
$\mathcal{C}$: the cost/penalty matrix for all arms and SoPs
sd : seed for random number generation
Ensure:
$\mathcal{S}$: a set with *arm-target* assignments
$\mathcal{T}^o$: the set of targets not allocated

1: **function** SGPB-CONSTRUCTION($\mathcal{T}^i$, $\mathcal{P}$, $\mathcal{C}$, sd)
2: $\mathcal{T}^o \leftarrow \varnothing$, $\mathcal{S} \leftarrow \varnothing$, $\mathcal{A}^a \leftarrow \varnothing$
3: $\mathcal{A} \leftarrow \{1, \ldots, 12\}$
4: **while** $\mathcal{T}^i \neq \varnothing$ **and** $|\mathcal{A}^a| < 12$ **do**
5: Built LPC^t with those targets $r \in \mathcal{T}^i$ having the greatest benefits
6: Select an element $t \in LPC^t$ at random
7: $\mathcal{T}^i \leftarrow \mathcal{T}^i \setminus t$
8: Let s be the SoP where t belongs to
9: $\mathcal{A}^c \leftarrow \{a \in \mathcal{A} \mid (a \cap \mathcal{A}^a = \varnothing) \wedge (cost(\mathcal{C}, a, s) \geq 0)\}$
10: **while** $\mathcal{A}^c \neq \varnothing$ **do**
11: Select an element $i \in \mathcal{A}^c$
12: Let q be the configuration positioning the arm i at t
13: **if** arm i configured at q does not collide with any $j \in \mathcal{A}^a$ **then**
14: $\mathcal{S} \leftarrow \mathcal{S} \cup < i, q, t >$
15: $\mathcal{A}^a \leftarrow \mathcal{A}^a \cup i$
16: **go to** 4
17: **end if**
18: $\mathcal{A}^c \leftarrow \mathcal{A}^c \setminus i$
19: **end while**
20: $\mathcal{T}^o \leftarrow \mathcal{T}^o \cup t$ ▷ Cannot find allocation for t
21: **end while**
22: **if** $\mathcal{T}^i \neq \varnothing$ **then**
23: $\mathcal{T}^o \leftarrow \mathcal{T}^o \cup \mathcal{T}^i$ ▷ Fill output target list with those remaining
24: **end if**
25: **return** $\mathcal{S}$, $\mathcal{T}^o$
26: **end function**

this line, an arm patrolling the current target is picked, and the feasibility of this assignment checked in line 13. The particular selection method to use has not been explicitly specified in algorithm 4, since we propose different options in section 5.3.1.1. Once an arm has been found for the current target not colliding with the previously planned assignments, the 3-tuple $< i, q, t >$ defining the new allocation is included in the solution (line 14). Next, in line 15, the list of arms already assigned is appropriately updated, the search for a suitable arm is abruptly interrupted in line 16 and a new target is planned. Contrarily, if the arm just selected presents a conflict, it is discarded in line 18, and a new one is chosen. If, unfortunately, after inspecting all unassigned arms patrolling the target, we cannot find a valid assignment, the target is included in the set $\mathcal{T}^0$. As commented above, this set contains those targets not allocated in the current run of the construction phase. If for any reason, there are targets still pending allocation, but all arms are already assigned, the processing loop in line 4 breaks and the targets left are pushed into $\mathcal{T}^0$ (lines 22-24). Finally, the solution formed by the assignments found is returned in line 25.

There is an important aspect to consider. Algorithm 4 might return an incomplete solution. This will happen if it is unable to allocate all arms available. In that case, we will attempt to complete the solution in the second step of the target allocation procedure, whose pseudo-code can be seen in algorithm 5. Finally, note that the function `SGPB-Construction` takes in an integer named sd, which specifies the seed to be employed in those parts of the algorithm generating a pseudo-random number.

Greedy Heuristic Function and Construction of the Limited Pool of Candidates For the construction of an LPC like the one used in line 5 of algorithm 4, two different approaches have been followed traditionally. The size of the pool can be restricted either by the number of elements or by their quality (Resende and Ribeiro, 2003). Let $c(x)$ be the incremental cost of the candidate x associated with the inclusion of that element into the solution. In addition, we respectively denote by c_{min} and c_{max} the minimum and maximum relative costs found in the remaining population at each iteration. Then, in the former scheme, the LCP consists of the n elements having the best incremental cost. However, in the latter case, the size of the pool is variable. It is populated according to a threshold value $0 \leq \alpha \leq 1$ modulating the greediness of the approach. Specifically, the authors of the seminal work in GRASP proposed that at each iteration the LPC is filled with those pending elements x with a cost value $c(x) \in [c_{min}, c_{min} + \alpha(c_{max} - c_{min})]$. The parameter α can be seen as a diversification–intensification trade-off control component. When it is equal to 1, a completely random choice is obtained, consequently, producing the maximum diversification. Contrarily, when $\alpha = 0$, low diversification is obtained since selection is made following a pure greedy scheme.

Now, we show how the LPC is populated in MIRADAS. Let η_i be heuristic information for a target i, giving a measure of the relative benefit of adding this target to the partial solution. This

heuristic is made up of a mixed rule of target priority and arms reachability:

$$\eta_i = p_i + \gamma \frac{r_i}{n} \tag{5.4}$$

where p_i is the target priority, n the arms available in the MXS system, r_i the number of arms reaching the target and $\gamma \in [0, 10]$ a parameter governing the influence of the reachability factor in the expression. Note that in our case, we are working with a benefit function instead of a cost function. Hence, the higher the value of a candidate, the more appropriate it is for incorporation. Then, a decision is made about which targets still pending an assignment are selected to be in the pool of candidates according to the following greedy function:

$$q(\alpha) = \eta_{i_{max}} - \alpha(\eta_{i_{max}} - \eta_{i_{min}}) \tag{5.5}$$

All targets with a heuristic value η_i greater or equal to $q(\alpha)$ are included in the LPC, being $\alpha \in [0, 1]$.

The performance of the GRASP metaheuristic is very sensitive to the α parameter, and many strategies may be applied for its initialization (Talbi, 2009). One of them is the *static* approach, where the value of the parameter is initialized to a constant value before the search for a solution is carried out. The most frequently employed range is naturally closer to the greedy choice: [0.7, 0.9] (Resende and Ribeiro, 2003). We here will follow a static approach.

Selecting an Arm for a Given Target As commented before, in line 11 of the construction phase, see algorithm 4, an arm is chosen from those still pending an assignment. Here we describe two different methods to perform this selection. The first of them is the most obvious and straightforward, as well as the one focusing on the most immediate benefit. The arms are sorted in ascending order according to their associated penalty value and explored sequentially, starting from the most promising one to the least. This option, from now on, will be referred to as greedy arm selection (GAS).

On the other hand, the second method, that we will call penalty-based arm selection (PBAS), includes randomness. Several lists are constructed, one for each penalty value distinct to -1. Every list is populated with the arms showing the corresponding penalty value for patrolling the target currently under process. In this way, if an arm has different patrolling penalties, it will be present in several lists. Then, each list is shuffled. Afterward, an arm is picked by systematically exploring these shuffled lists, beginning with the one storing the arms with the lowest penalty. If all arms there are unsuccessful, then the list with the immediately higher priority is explored. This process continues until a no conflicting arm is found or all assignments proposed by all the lists have been inspected without success.

5.3.1.2 Solution Padding Phase

The function `Solution-Padding` found in line 7 of algorithm 3 tries to allocate those arms not paired with a real target to a *fake* target (blank sky). That is, it attempts to sequentially find a

position for each unassigned arm so that it does not collide with those previously assigned. The first position that will be explored is the current arm position, secondly the arm home location, and if the algorithm does not succeed, then a more sophisticated local search begins. If the arm i to be positioned to blank sky happens to have its neighbors allocated, as in fig. 5.2, the area connecting the points A, B, C, and D is randomly explored. Every time a point of this area is drawn, it is substituted into the inverse kinematics expressions to determine the configuration q positioning the mirror of the unassigned arm in that particular Cartesian coordinate. If the arm i configured at q does not collide with the others already allocated, the partial assignment plan is consequently updated with the pair <*i*,*q*>. But, if after a given number of attempts, the local search for blank sky fails, the function `Solution-Padding` terminates returning an incomplete solution. Analogously, if any of the adjacent arms has not been previously allocated a similar area is determined and progressively sampled.

5.3.1.3 Local Search Phase

The solutions generated are not guaranteed to be locally optimal. Therefore, a local search algorithm can be applied as an attempt to improve them. An algorithm of this sort works in an iterative fashion. It successively replaces the current solution by a better one in a neighborhood of the first, ending at a local optimum. The degree of success of this algorithm is directly connected with the choice of the neighborhoods as well as the starting solutions delivered by the construction phase. For a general survey on local search, see Aarts et al. (2003).

A neighbor to a given solution is built by the application of a *move* operator performing a small perturbation to the solution. The nature of these operators strongly depends on the optimization

Algorithm 5 Semi-Greedy Patrol-Based Solution Padding

Require:
sd : seed for random number generation
Ensure:
$\mathcal{S}$: a set with *arm-target* assignments

1: **function** SOLUTION-PADDING($\mathcal{S}$, sd)
2: Let $\mathcal{A}^a$ be the set of all arms in $\mathcal{S}$
3: $\mathcal{A} \leftarrow \{1, \ldots, 12\}$
4: $\mathcal{A}^p \leftarrow \mathcal{A} \setminus \mathcal{A}^a$ ▷ All arms pending
5: **for all** $i \in \mathcal{A}^p$ **do**
6: Find a position q for arm i not colliding with all $j \in \mathcal{A}^a$
7: **if** $q \neq \varnothing$ **then**
8: $\mathcal{S} \leftarrow \mathcal{S} \cup < i, q, \varnothing >$ ▷ Assigning arm i to *blank sky*
9: $\mathcal{A}^a \leftarrow \mathcal{A}^a \cup i$
10: **end if**
11: **end for**
12: **return** $\mathcal{S}$
13: **end function**

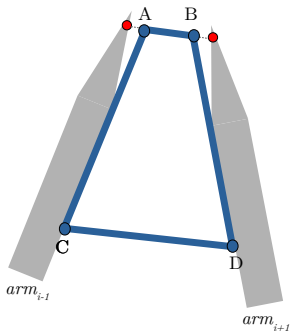


Figure 5.2: The area delimited by the points A, B, C and D is sampled when searching for a feasible location for the unassigned arm_i (the one located between the two shown in the figure).

problem to solve and on the associated representation of the problem (Talbi, 2009). In the case of the MIRADAS allocation algorithm, we can consider a permutation-based representation. A neighborhood often employed in these representations is generated by the pair-wise *swap* operator, which consists in exchanging the location of two elements s_i and s_j of the permutation. That is, tasks j_1 and j_2 (sky targets in our problem) are swap between different agents a_1 and a_2 (probe arms). For a permutation of size n, theoretically, the size of this neighborhood is $n(n-1)/2$, being n, in our case, equal to the number of arms in the instrument. Nevertheless, in practice, the neighborhood can be significantly reduced when considering the inherent physical constraints of the MIRADAS arms and the patrolling strategy followed. In addition to the use of the swap operator, we also consider the *replace* operator for local search, which has been employed in field allocation for fiber-based instruments as well (Miszalski et al., 2006). With this operator, the target previously assigned to an arm is replaced by another target not still allocated. As a reminder, the sky targets still awaiting for allocation are in the set $\mathcal{T}_k^o$, which was initialized during the construction phase; see algorithm 3.

Taken into account practical considerations, the swap operator will be only applied in those cases where the targets to exchange can be patrolled by the two arms involved in the move. The term *patrolled* here has not to be seen as if a given arm is mechanically able to reach a target, but with regard to the presence or no of a valid value in the penalty matrix previously introduced. Although other choices are checked at random, preference is always given to swaps involving adjacent arms. Finally, the replace operator randomly generates neighbors where a given arm is assigned to a pending target lying either: (i) on the default SoP associated to the arm or (ii) on any of the two adjacent SoPs (the ones immediately above and below) to the default one.

The local search is divided into two separate steps. In the first of them, only swap moves are considered, whereas in the second only replace moves are applied. In both, a *first improving* strategy is employed: the current solution is replaced by the first neighbor yielding an improvement

in the aggregated priority or in any of the three desirable aspects listed in section 5.1.

5.3.1.4 Solution Update

Once a complete solution is returned by either the construction or the solution padding phase, as shown in line 11 of algorithm 5, it is checked to see if it improves the solution previously stored. In order to keep the newly generated solution, it should satisfy one out of three conditions. The first of them refer directly to the objective function to maximize by the target allocator, the total aggregated priority. Therefore, if the sum of all the targets allocated in the current solution is greater than the sum in the best solution, then the function `Solution-Update` keeps the just generated assignment plan. The best solution is also updated if the new solution has the same aggregated priority and, additionally, has higher efficiency. That is, the number of arms effectively used in the plan is larger than the arms used in the one stored. Finally, if the new solution presents the same values for the sum of priority as well as the efficiency criterion that the solution stored, then the path-aware criterion mentioned in section 5.1 is checked.

5.4 Integer Programming Based Allocator

Although metaheuristics are preferable in many practical applications, classical optimization models are still useful in domains where the constraints can be appropriately defined and where the search space is not large.

The main goal of classical exact methods is to determine a global optimal solution. Unlike metaheuristics, exact techniques guarantee the optimality of the obtained solutions. In fact, many global optimal solutions may exist for a given problem, and consequently, if properly formulated, then exact methods can find all of them. Different families of optimization models are used in practice; however, one of the most successful is based on mathematical programming. Mathematical programming, and especially linear programming (LP), is one of the best developed and most utilized branches in operational research.

In LP, problems are formulated in canonical form using exclusively linear expressions as follows:

$$\max\ c^T x \tag{5.6}$$

subject to

$$Ax \leq b \tag{5.7}$$

$$x \geq 0 \tag{5.8}$$

where x denotes the vector of variables to be determined, c and b are vectors of known coefficients, and, finally, A is a known matrix of coefficients.

Equation (5.6) represents the objective function to be optimized, in this case maximized, although it could also be minimized if required. Equation (5.7) and eq. (5.8) express the constraints

imposed by the problem of study. For problems where the domain of the decision variable is continuous, there exist algorithms like the *simplex method* to efficiently determine a solution (Matousek and Gärtner, 2007). Efficiency is achieved because the feasible region of the problem is a convex set, and the objective function is convex as well. However, in many real optimization problems, the resources to consider are indivisible and, consequently, the decision variables are discrete. In this particular case, the linear program modeling the problem is popularly referred as *integer program* (IP) (Schrijver, 1986).

The classical assignment problem is a very special case of the more general transportation problem due to its particular structure (Schrijver, 2002). In literature, this problem is often seen from the perspective of a bipartite graph. Using the specific language of graph theory, we have a graph $G = (V, E)$ with a vertex set V divided into two disjoint sets X and Y, and an edge set E. Each edge $e_{ij} \in E$ connects one vertex i of X to one vertex j of Y; see fig. 5.3. Besides, the sets X and Y have the same cardinality and each edge $e_{ij} \in E$ has a non-negative weight w_{ij} . Then, we want to find a *maximum-weight perfect matching*. In other words, a subset $M \subseteq E$ such that each vertex of both X and Y is incident to exactly one edge of M and at the same time maximizes the sum of weights. Analogously, the problem can also be formulated as a minimization problem, consisting then in the determination of the minimum-weight perfect matching.

If we introduce a binary decision variable $x_{ij} \in \{0, 1\}$ for each edge e_{ij}, then the assignment problem can be modeled as an IP. These variables will take the value 1 if the corresponding edge belongs to the subset M. Otherwise, they will hold a 0. That is, $x_{ij} = 1$ if the matching represented

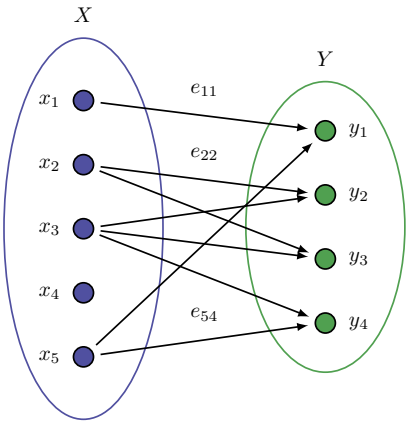


Figure 5.3: A bipartite graph with two independent sets U and V, and edges connecting elements of both sets.

by the edge e_{ij} leads to the maximization of the following objective function:

$$\sum_{e_{ij} \in E} w_{ij} x_{ij} \tag{5.9}$$

subject to

$$\sum_{i \in X} x_{ij} = 1 \qquad \text{for } j \in Y \tag{5.10}$$

$$\sum_{j \in Y} x_{ij} = 1 \qquad \text{for } i \in X \tag{5.11}$$

$$x_{ij} \in \{0, 1\} \qquad \text{for } i \in X, j \in Y \tag{5.12}$$

where the constraints that a vertex $v \in V$ have exactly one incident edge of M is expressed by eq. (5.10) and eq. (5.11). By relaxing the constraints in (5.12) to the weaker ones $0 \leq x_{ij} \leq 1$, an assignment plan can be found by any algorithm solving an LP (e.g. the simplex method). However, there exist specific algorithms solving this assignment problem in polynomial-time, being the oldest of them the Hungarian algorithm (Kuhn, 1955; Munkres, 1957).

As commented in section 5.2, the Hungarian method is one of the techniques used in KARMA, the field configuration application employed in KMOS, to allocate the 24 probe arms of the instrument (Wegner and Muschielok, 2008). As a matter of fact, this algorithm is not suitable for assignment scenarios where additional constraints need to be considered. It is the case in KMOS as well as in MIRADAS, where the integer programming formulation should take into account, in addition to the priorities of the targets, a collision-free constraint.

In MOS allocation scenarios with multiple constraints of diverse types, intuition says that the natural choice would be employing the weights w_{ij} in eq. (5.9) for the target's priority. However, in KMOS, this matrix of weights has been utilized to enforce assignment plans with no collisions. Unfortunately, in their case, it can be only guaranteed if arms were dimensionless. On the other hand, as stated above, the classical assignment problem as modeled above assumes that both independent sets X and Y are of equal size. But, that is not the more frequent scenario in MOS instruments, where users typically want to observe more targets than manipulators are available. In the following section, we extend the classical assignment model to naturally satisfy all considerations mentioned as well as others related to MIRADAS.

5.4.1 MIRADAS Problem Formulation

Here, we provide a mathematical definition for the MIRADAS Integer Programming Based Allocator (IPBA). Let be $K = \{1, \ldots, m\}$ a set of probe arms, and $J = \{1, \ldots, n\}$ a set of observation jobs (or targets) for a given telescope pointing. For each job $j \in J$, we define a non-negative prize p_j representing its priority for observation (target priority). Moreover, we denote by $V = \{v_1, \ldots, v_n, v_{n+1}, \ldots, v_{n+m}\}$ the set of locations in the instrument FOV reachable by the arms in K. The first n elements of that set correspond to those locations associated with the observation jobs

in J, whereas the remaining m elements are the parking locations associated with each arm in K. Due to the mechanical constraints of the arms and their positions in the MXS bench, no all arms in the system can reach all observing locations. Therefore, we define for each arm in K, the set $V_k \subset V$ representing the locations reachable by the arm k. Note that, the parking location v_{n+k} is only available at V_k. Similarly, we define an additional subset $K_j \subset K$ for every location $v_j \in V$ as the set of arms capable of reaching such a location. Finally, the prize b_j for every location $v_j \in V$ associated to a real science target is assumed to be in the range of $\mathbb{N}$ (taken as 1, 2, 3, ...), whereas the prize for all park locations is 0.

The set $A = \{(v_j,k) \subset V \times K \mid v_j, v_{n+k} \in V_k,\ k \in K\}$ describes the set of all potential assignments of jobs to arms. For each pair (v_j,k), we define d_{jk} as the distance traveled by the arm k to get to the location v_j from its parking location v_{n+k}. Although the set of all potential assignments A is significantly large, many of them are incompatible configurations that reduce the cardinality of this set -for example, all those pairs involving arms colliding against each other. Finally, as there may be more than one set of pairs fulfilling the previous conditions, a metric is required to select the best of them. With the purpose of gathering these constraints, we denote by $\mathscr{H} \subset A \times A$ the set of all pairs of assignments making two different arms crash into each other.

A feasible solution for this problem is an assignment plan $\mathscr{A} \subseteq A$, where each arm in K has been assigned to exactly one location in V, each location in V has been assigned to at most one arm, and no incompatible pair of an assignment is allowed. The here-proposed model is based on two families of decision variables. For each potential assignment $(v_j,k) \in A$, the binary variable x_{jk} specifies whether the arm k reaches the location v_j from the location v_{n+k}, in the solution. The continuous variable x_l takes the value of the most prolonged arm movement in a solution. This value, once multiplied adequately by the maximum velocity an arm can move, gives a lower bound for the system *makespan* -the time needed to finish all jobs[3]. The following set of constraints $\mathscr{P}$ represents the set of all those feasible assignments $\mathscr{A}$ described by the variables mentioned before:

$$\sum_{j \in V_k} x_{jk} = 1 \qquad \text{for } k \in K \tag{5.13}$$

$$\sum_{k \in K_j} x_{jk} \leq 1 \qquad \text{for } v_j \in V \tag{5.14}$$

$$x_{ik} + x_{jk'} \leq 1 \qquad \text{for } ((v_i,k),(v_j,k')) \in \mathscr{H} \tag{5.15}$$

$$d_{jk} x_{jk} \leq x_l \qquad \text{for } j \in V_k,\ k \in K \tag{5.16}$$

$$x_{jk} \in \{0,1\} \qquad \text{for } v_j \in V_k,\ k \in K \tag{5.17}$$

Constraints (5.13) enforce that each arm in K is assigned to exactly one location in V, which can be either a real target or its corresponding park position. Inequalities (5.14) guarantee that at most one arm is assigned to each location. Indeed, these expressions enable the allocator to

[3]The time required for all arms to reach their corresponding destinations

work with a number of sky targets larger than the number of arms in the system. In addition, note that there could be a non-motion assignment where the origin and destination of the arm are the same locations (i.e. park). Equations (5.15) prevent assignments incompatible. In other words, they avoid that any pair of arms in the system collide among them once they have been assigned a location in the FOV. Constraints (5.16) define variable x_l as the value of the longest move performed by the arms. Bounds (5.17) establish that the decision variables x are binary. We notice that constraints (5.13) and (5.14) are particular cases of those modeling a generalized assignment problem (Fisher et al., 1986).

Our aim is the optimization of the following objective function:

$$\max_{x \in \mathscr{P}} \sum_{(j,k) \in J \times K_j} p_j \, x_{jk} \tag{5.18}$$

Equation (5.18) maximizes the total priority associated with the pairs in the solution. The optimization process will favor assignments with real sky targets over park locations, since, as commented above, targets will always posses higher priorities than park placements. In fact, there may exist more than one set of assignments maximizing this expression. Therefore, similarly to what has been done in section 5.3.1.4, one of them will be selected according to additional criteria such as high efficiency and small makespan.

In practice, for the implementation of the model proposed above, three different data structures need to be pre-computed. First, a reachability data structure holding all those sky locations (targets) that every arm in the system is capable of reaching. This structure is populated with the information obtained by applying the inverse kinematics equations of all arms for all targets. A sparse matrix of $n \times (m+n)$ positions, where n denotes the number of arms and m the user-defined targets, can be used to store this information. As a reminder, just to mention that the park locations of each arm have to be also stored. However, if the target field is dense many of the elements of the matrix will contain non-representative data since not all arms might be able to reach all targets. Hence, other data structures more efficient in terms of memory consumption should be considered.

The second structure to be pre-calculated is that containing all possible pairs of conflicting assignments. This involves the construction of a simple characteristic polygon for each possible location in the set V that every arm can reach. Then, all possible pair-wise collisions between them need to be checked and the results adequately stored. This operation might be time-consuming if the density of the field is large. Other field configuration tools for MOS also pre-compute this information, which is employed as a look-up table when determining whether a proposed allocation is valid (Miszalski et al., 2006). Although in the definition of V it has been considered only a single rest location (park) for each arm, additional ones could be included (e.g. the current arm positions). Finally, a matrix containing the arm displacements [4] need to be also pre-computed.

[4] To be precise, the active joints' displacements.

Without loss of generality, here we have been assuming that all arms start their corresponding motions from park positions. Consequently, the last matrix mentioned in the previous paragraph will contain for each arm the aggregated displacement from its initial position to the final one of every actuated joint when going from park to every reachable target. However, nothing prevents us from adding as well the displacement of the arms from their current positions. In fact, these displacements provide a lower bound as they are modeling a straight line in the joint space, which is the shorter path connecting two points.

5.5 Simulations and Results

We implemented the SGPBTA solution in Python, while the one based on integer programming was coded in C++ and the associated model solved with the help of the CPLEX optimization library. We conducted several experiments with random catalogs as well as fields of real targets. Although some of the fields obtained probabilistically might not reflect the characteristics of real MIRADAS science cases, they are valuable to blindly determine how our methods behave. The stochastic fields are also studied in conjunction with distinct target priorities schemes. Finally, the arms and SoPs arrangements employed in all of our experiments are illustrated in fig. 5.1.

The process to demonstrate the performance of the two approaches presented is the following: given a particular dataset, we successively run the selected algorithm until all targets have been assigned to an arm. That is, we initially call the allocator with the complete set of targets. If after its execution, not all targets were successfully assigned, we iterate again over the list of pending targets. In this way, depending on the number of targets in the dataset as well as their locations in the instrument FOV, one or more observational configurations might be required to complete the observation of the field of interest. Additionally, if instead of target completeness, we turn our attention to priority, it could also be possible that more than one configuration might be necessary to acquire the targets with the highest priority.

5.5.1 Metrics

In this section, we introduce a ratio to assess the complexity of an experiment and the quantitative metrics that will be extensively employed to study the performance of the allocation methods proposed.

Difficulty As its name indicates, this ratio gives an estimation of the patrolling complexity of the problem. The difficulty of a given field is defined as

$$dif = \frac{1}{n}\sum_{i=1}^{n}\frac{t-t_i}{t} \tag{5.19}$$

where t represents the total number of targets in the field and t_i the number of targets that can be reached by arm i. Those datasets where most of the targets are concentrated in a reduced area

of the instrument FOV are said to be biased. As more biased a field is, the more complex it is. The values obtained by this metric are highly dependable of two distinct factors: the location of the targets and the patrolling strategy used by each arm in the system.

Field Completeness It is defined as the minimum fraction of targets to be observed in a given field. It does not take into account the priority of the targets. Field completeness of 100% indicates that we are interested in the allocation of all elements in the data set, whereas 50% completeness indicates we want only allocate half of the targets. The higher the difficulty ratio of a given dataset, in principle, the greater the number of configurations required to achieve 100% field completeness.

Priority Completeness It is defined as the minimum fraction of the aggregated priority to be observed in a given field. Completeness of 75% indicates that 75% of the sum of the priorities of all targets is allocated. The higher the difficulty ratio of a given dataset, the greater the number of configurations required to achieve 100% priority completeness.

Optimal Number of *Observational Configurations* For a given data set, this metric provides a minimum bound for the number of configurations required to obtain 100% field completeness. We define it as

$$\tau = \frac{t}{n} \tag{5.20}$$

where t denotes the number of targets in the field, while n represents the arms available in the system.

Arm Yield It shows the degree of utilization of the resources in the MXS system. This metric can be applied in two different ways. First, to analyze how many arms are in use for a given observational configuration. In this case, a yield of 100% represents that all arms in service are allocated to a science target. Second, to determine the efficiency in terms of usage of the arms available in a given allocation plan[5]. The allocation plan arm yield is computed by averaging the arm yield of each of the individual configurations.

5.5.2 Benefits of Randomness

Before discussing in depth several synthetic and real fields, we would like to emphasize the benefits of adding randomness in metaheuristics and especially in the GRASP-like approach we have followed in SGPBTA. To illustrate this, we have generated the field in fig. 5.4a with a total number of 14 targets of distinct priorities. Note that one target of maximum priority is located at the center of the FOV. There is nothing preventing any arm physically reaching that point. Therefore, if the patrolling strategy associated with the arms allows it, this target could be

[5] An allocation plan refers to the set of observational configurations required to obtain a 100% field completeness.

reached by every arm in the system. Indeed, the penalty matrix we use in this example, for a point in the center (it belongs to all SoPs), gives to each arm the same patrolling preference

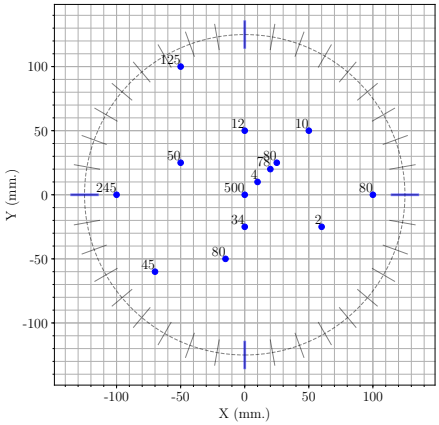
$$\mathbf{C} = \begin{bmatrix} 0 & 10 & 20 & 30 & -1 & -1 & -1 & -1 & -1 & 35 & 25 & 15 \\ 15 & 0 & 10 & 20 & 30 & -1 & -1 & -1 & -1 & -1 & 35 & 25 \\ 25 & 15 & 0 & 10 & 20 & 30 & -1 & -1 & -1 & -1 & -1 & 35 \\ 35 & 25 & 15 & 0 & 10 & 20 & 30 & -1 & -1 & -1 & -1 & -1 \\ -1 & 35 & 25 & 15 & 0 & 10 & 20 & 30 & -1 & -1 & -1 & -1 \\ -1 & -1 & 35 & 25 & 15 & 0 & 10 & 20 & 30 & -1 & -1 & -1 \\ -1 & -1 & -1 & 35 & 25 & 15 & 0 & 10 & 20 & 30 & -1 & -1 \\ -1 & -1 & -1 & -1 & 35 & 25 & 15 & 0 & 10 & 20 & 30 & -1 \\ -1 & -1 & -1 & -1 & -1 & 35 & 25 & 15 & 0 & 10 & 20 & 30 \\ 30 & -1 & -1 & -1 & -1 & -1 & 35 & 25 & 15 & 0 & 10 & 20 \\ 20 & 30 & -1 & -1 & -1 & -1 & -1 & 35 & 25 & 15 & 0 & 10 \\ 10 & 20 & 30 & -1 & -1 & -1 & -1 & -1 & 35 & 25 & 15 & 0 \end{bmatrix} \tag{5.21}$$

We do three experiments, in all of them we run 30 iterations of the SGPBTA allocator with $\gamma = 0$. Since there are more targets to observe than arms in the system, the number of observational configurations required to obtain 100% field completeness should be greater than one. We will only focus on the first computed configuration.

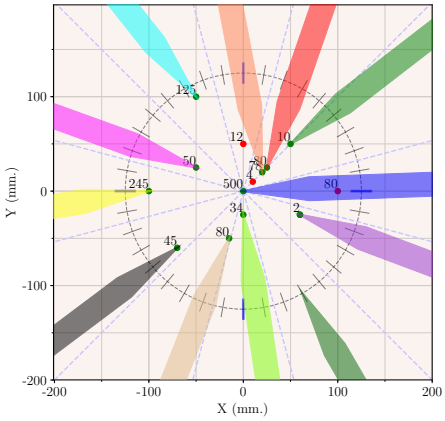
In the initial experiment, we use a pure greedy strategy ($\alpha = 0$). That is, in every step of the solution construction, the best immediate option is selected. Considering that, the target with the highest priority is picked and an unordered list with all arms patrolling it is created. The element at the top is chosen. In this case, the arm 0, establishing this way the assignment pair <500, 0>, where the first component represents the target and the second the arm allocated. Unluckily, this pairing, as can be seen in fig. 5.4b, makes the target 80 unreachable since it is under arm 0. The total aggregated priority of the first obtained configuration is 1249 and all arms but one are assigned to targets (green dots in fig. 5.4b).

We perform a new experiment, but this time, we use $\alpha = 0.05$, which, although being smaller, provides some degree of randomness in the order that targets are picked for allocation. As shown in fig. 5.4c, now the arms 1, 2 and 3 are assigned to different targets, yielding a final aggregated priority of 1251. Like the greedy approach, the arm 10 remains unallocated.

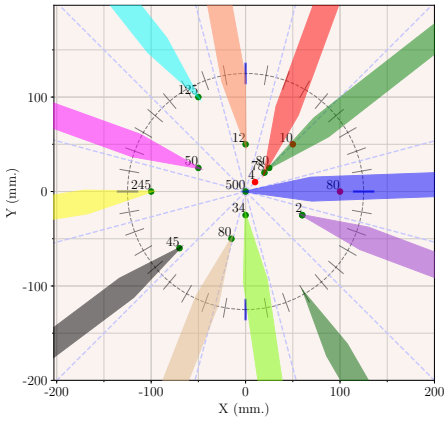
Finally, we maintain $\alpha = 0.05$, and now we enable a stochastic arm selection by employing the PBAS variant. This mechanism is useful to explore alternative solutions when for a given target, more than one arm has the same patrolling preference, which is the case for the target located at the FOV center. As can be appreciated in fig. 5.4d, due to this additional randomness factor, the central target is allocated to arm 11, leaving the target with priority 80 in a reachable state. This target is now assigned to arm 0, consequently increasing in a significant way the aggregated



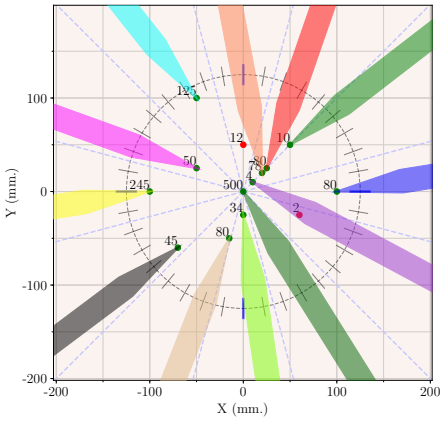
(a) Field with 14 targets where one of them is at the FOV center. The number above the blue dots (targets) indicates the target priority.



(b) Pure greedy SGPBTA, returning an aggregated priority of 1249.



(c) SGPBTA with $\alpha = 0.05$, resulting in an aggregated priority of 1251.



(d) SGPBTA with $\alpha = 0.05$ and arms shuffle, resulting in an aggregated priority of 1331.

Figure 5.4: Allocation for a field considering different greedyness.

priority of this solution. Note that all arms are successfully allocated as well.

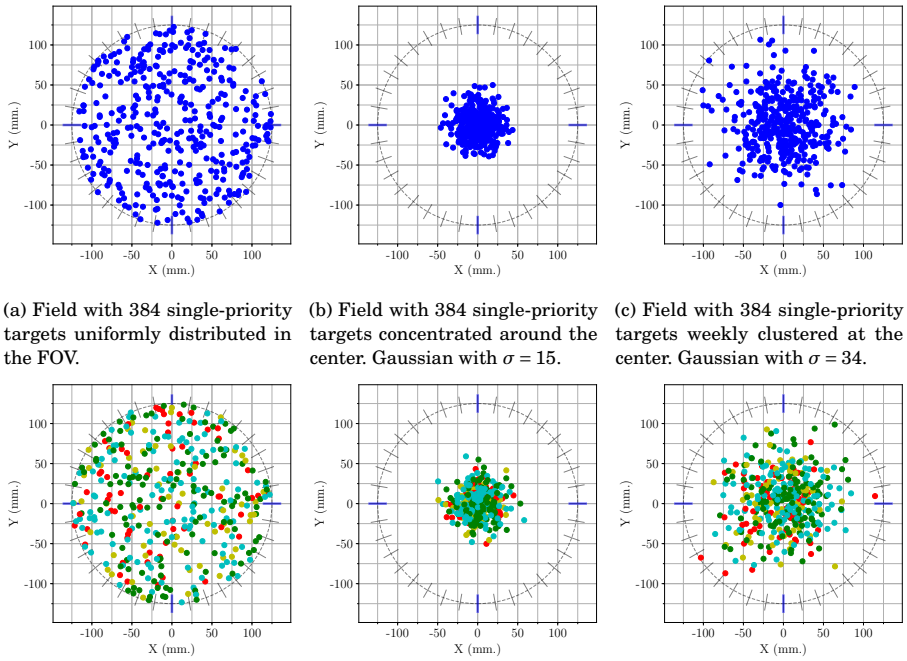
5.5.3 Synthetic Scenarios

As previously mentioned, MIRADAS has been conceived as a common-user instrument for GTC. It should be capable of working with scenarios of very different nature, showing a large variety in the spatial distribution of the targets as well as in the priority schemes. We tested our solutions with mock catalogs where the location of the corresponding targets was randomly generated, applying uniform and Gaussian distributions. For the latter case, two different kinds of fields were created: one where the majority of the population was concentrated at the center of the FOV and other where the targets were weakly clustered at the center. For each family, two different collections of 1400 different datasets were created. Specifically, in each case, we built 200 instances for each of the seven following combinations: 6, 12, 24, 48, 96, 192, and 384 targets. In the first collection, every target has the same priority (1), whereas in the other, priorities were assigned probabilistically. In all multiple-priority fields, the proportion of targets of the same priority was maintained constant. The 10% of the targets were assigned a priority of 500, 40% of the targets a priority of 100, 40% of the targets a priority of 10, and, finally, the rest were assigned a priority of 1. Some examples of the synthetic fields generated are shown in fig. 5.5.

The difficulty associated with two particular sets of experiments can be appreciated in fig. 5.6. The figure contains two different groups of 7 boxplots each. Those situated in the upper part belong to the datasets uniformly distributed, whereas the lower part shows the complexity of the datasets densely clustered around the center of the FOV. The boxplots are arranged according to the number of targets in the dataset, and each has been generated considering the 200 corresponding instances. In each boxplot, the minimum and maximum values are represented by the lowest and highest horizontal lines, the upper and lower ends of the box represent the upper and lower quartiles, the thick line within the box shows the median, and the isolated points are the outliers of the distributions. As expected, the higher the number of targets in a dataset, the lower the difficulty. Moreover, for a given number of targets, the more a dataset is biased toward the center of the FOV, the more the difficulty of a dataset tends to decrease. This is a consequence of the particular workspace of the MIRADAS arm. As seen in chapter 3, the greater proportion of the FOV points reachable by the probe are located in a close neighborhood of the center.

5.5.3.1 SGPBTA

The selection of the values to assign to each of the parameters governing the SGPBTA algorithm is difficult since the variety of scenarios that the allocator should work with is, in theory, large. Moreover, the common sense says that in every particular scenario, totally distinct choices might be needed to find an optimal solution. It is not our intention to focus here on the proper tuning of these parameters but showing the performance of our methods in broad terms. Unless otherwise stated, the number of runs chosen for the experiments was 30, the value for γ is 0.1, the penalty



(a) Field with 384 single-priority targets uniformly distributed in the FOV.

(b) Field with 384 single-priority targets concentrated around the center. Gaussian with $\sigma = 15$.

(c) Field with 384 single-priority targets weekly clustered at the center. Gaussian with $\sigma = 34$.

(d) Field with 384 multiple-priority targets uniformly distributed. Dots in red have priority 500, dots in green 100, dots in cyan 10 and dots in yellow 1.

(e) Field with 384 multiple-priority targets concentrated around the center. Gaussian with $\sigma = 15$. Colors represent the same priorities that those in fig. 5.5d.

(f) Field with 384 multiple-priority targets weakly clustered at the center. Gaussian with $\sigma = 34$. Colors represent the same priorities that those in fig. 5.5d.

Figure 5.5: Synthetic fields used in tests.

matrix was the one seen in eq. (5.21) and, finally, the PBAS method was applied whenever two or more arms present the same patrolling preference. We conduct all experiments with $\alpha = 0.2$ and $\alpha = 0.8$, respectively corresponding to a low degree of randomness and a high one. We will refer to this two configurations of the algorithm as *SGPBTA 0.2* and *SGPBTA 0.8*. We have studied several aspects and otherwise stated, the differences in the results from SGPBTA 0.2 and SGPBTA 0.8 are negligible. Consequently, otherwise mentioned, figures will show only the results obtained with SGPBTA 0.2.

The first point analyzed is the arm yield of the allocation plans corresponding to the three different probabilistic distributions under tests. For each distribution, we have put together the plans returned by the allocator for all single and all multiple priority scenarios. The results are shown by means of histograms, where the horizontal axes represent the allocation plan yield and in the vertical ones is specified the frequency (in percentage) for a given arm yield. All

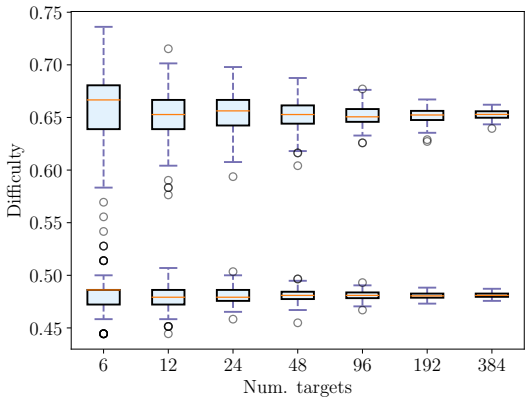
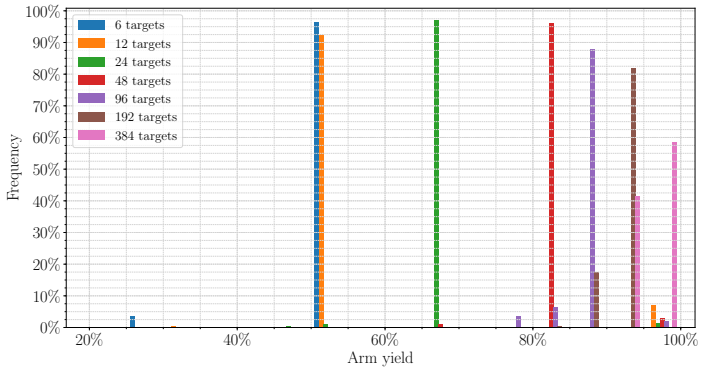


Figure 5.6: Field difficulty for experiments with uniformly distributed single-priority targets (up) and single-priority targets highly concentrated at the center (down)

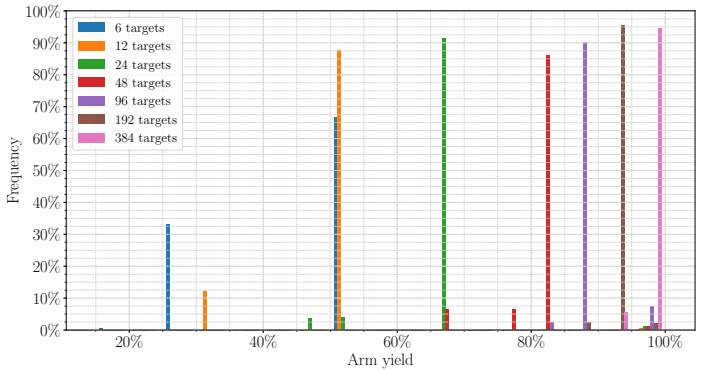
horizontal axes have been divided into a series of intervals, where all bins have the same width -representing an arm yield of 5%. That is, a bar contained, for example, in the horizontal interval 90%-95% represents the number of times the corresponding dataset family has produced an arm yield of that same value.

Confirmed by fig. 5.7, those scenarios where the number of targets to observe is much larger than the arms in the system do not suffer from poor arm yield. That is because of the high excess of targets available for every arm. For every group of datasets, there exist a clear central tendency, being the percentage of plans showing negative deviation very reduced. For instance, all allocation plans that correspond to datasets containing 384 targets deliver, for all distributions, arm yields in the 90%-100% interval. The 80% of the datasets of 192 targets obtain arm yields of 90%-95%. The vast majority of scenarios with 96 targets show an arm yield slightly inferior, in the range of 85%-90%. Regarding the scenarios with only 6 targets, as expected, the yield is mostly around 50%. Note, on the other hand, that a few plans with 12, 24, and 48 targets achieve higher yields than its associated mean.

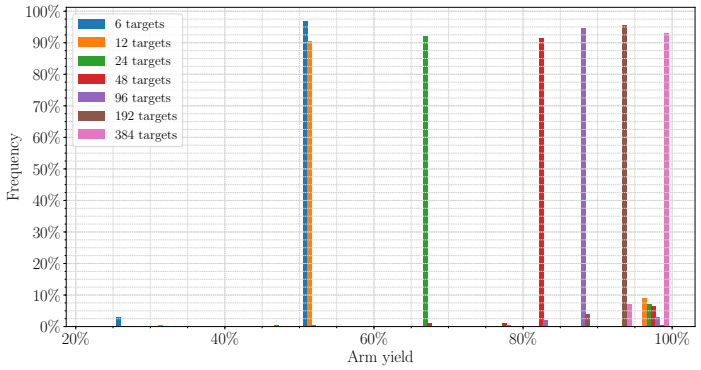
We now consider the arm yield in terms of the individual observational configurations without taking into account the allocation plan they belong to. As in the previous study, for each probabilistic distribution, data coming from single-priority scenarios and multiple-priority ones are grouped together and jointly plotted in the same subfigure of fig. 5.8. The results are also shown in the form of histograms, whose bins share the same characteristics of those in the previous study. Once again, it can be observed a central tendency for most of the groups in the dataset. This tendency is more relevant as the number of targets in the field increases. Indeed, for the sets with 384 and 192 targets more than 80% of the times the allocator delivers configurations



(a) Arm yield for uniform distributions obtained with SGPBTA.



(b) Arm yield for heavily clustered distributions obtained with SGPBTA.



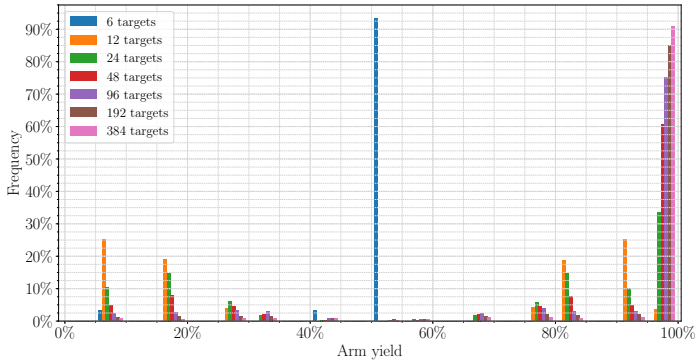
(c) Arm yield for weakly clustered distributions obtained with SGPBTA.

Figure 5.7: Allocation plans arm yield returned by SGPBTA.

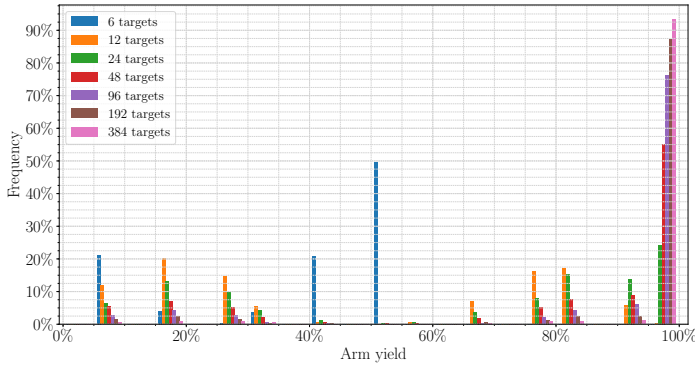
with a very high arm yield, in the interval 95%-100%. Around the 70% of sets in the family of 96 targets are also in that same interval. There are a few instances in the previous families of datasets showing lower yields. These mainly belong to observational configurations delivered in the final steps of the allocation process, where fewer targets are passed to the algorithm and, at the same time, their locations are not homogeneously distributed in the FOV. The predominance of a single arm yield interval is not so clear for the datasets containing 12 targets. These scenarios show the most inferior degree of a priori overall predictability. What is more, the highest achieved arm yield interval has, more or less, the same likelihood that the lowest. Therefore, datasets with only 12 targets should be seen as "double-edged sword" scenarios where, in general, both good and bad results might be obtained. Regarding test benches with only 6 targets, the yield is mostly the expected, around 50%. However, again, due to the bias of some scenarios, in some cases, more than one observational configuration is required to achieve the 100% field completeness. These instances with this second configuration are the ones represented in the bars showing lower yield. In general, the datasets where the number of targets is smaller or closer to the number of arms are critical since the results are susceptible to the locations of the targets in the FOV.

Now, we turn our attention to the number of observational configurations required to achieve a particular value of field completeness. As confirmed by the results of the experiments, the three different distributions show similar behavior. Hence, we will include here, as an example, only one plot, the corresponding to the uniform distribution. As in the previous graphical examples, the results from single and multiple priority targets scenarios are appropriately combined together. The average number of configurations, as appreciated in fig. 5.9, linearly grows in all cases following a step pattern, which is more marked as the number of targets increases. The reason behind this pattern is related to the way this metric is computed. It basically determines the minimum number of observational configurations to obtain a given field completeness. But, as it is obvious, a given number of configurations can contain targets responding to various field completeness percentages. It is worth noting how this stepped behavior disappears when completeness is almost 100%. This is due to the low arm yield obtained in the last iterations of the allocation process, where observational configurations tend to contain every time fewer targets assigned.

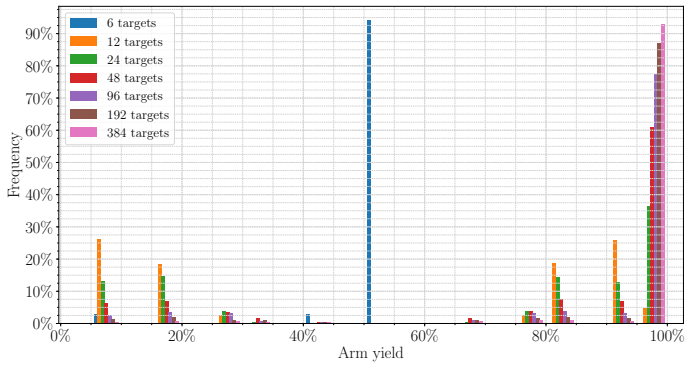
Finally, like field completeness, as confirmed by the experiments, the three different distributions show equivalent behavior in terms of the average number of observational configurations versus priority completeness. Here, we include only one plot, the one obtained from the data coming from the uniform distribution. As seen in fig. 5.10, initially priority increases following a concrete step pattern. More specifically, this pattern is repeated from 0 to about 70% priority completeness. Then, growth abruptly becomes steeper, and finally, around 95% priority completeness, the average number of configurations increases more or less exponentially. This step pattern behavior again is connected with how priority completeness is defined, whereas the different rates of increment relate to the particular priority distribution used in the experiments. If there



(a) Arm yield for uniform distributions obtained with SGPBTA.



(b) Arm yield for heavily clustered distributions obtained with SGPBTA.



(c) Arm yield for weakly clustered distributions obtained with SGPBTA.

Figure 5.8: Observational configuration arm yield in SGPBTA.

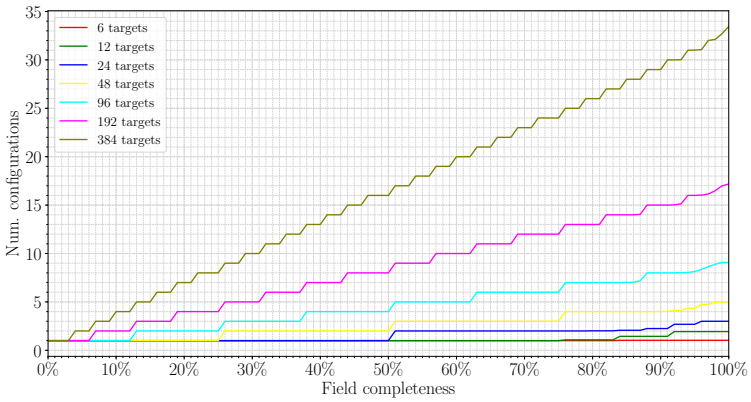


Figure 5.9: Average number of observational configurations versus field completeness for SGPBTA.

is nothing preventing it, the algorithm will give preference to the allocation of those targets possessing the highest associated priority. In the beginning, the targets with higher priorities are still available, but then, once the 500 and 100 priority targets are consumed, only targets with relatively much more lower priority (10 and 1) are pending. Therefore, comparatively much more configurations are required to achieve similar growth in priority completeness. That is why the slope becomes steeper. Finally, the targets left for the last steps of the process are mainly those showing a priority value equal to 1, which are the 10% of the total. Although some of them might have been used to complete some previous observational configurations, many remain unallocated until the last stages.

5.5.3.2 IPBA

In this section, we perform the same experiments than the ones carried out in section 5.5.3.1, but now employing the Integer Programming Based Allocator. This time, the patrolling strategy adopted for each arm in the system corresponds to each arm's workspace. In addition, relying this allocator upon an exact mathematical optimization method, with only a single execution of the algorithm for each problem instance is enough. The allocator always determines the best -or optimal- plan according to the model defined.

We start by considering the arm yield of the allocation plans delivered by the algorithm for the three different random distributions. Once again, for each distribution, we have combined the plans obtained for all single and all multiple priority scenarios, and the results are represented employing histograms. As shown in fig. 5.11, the algorithm produces plans having high arm yield

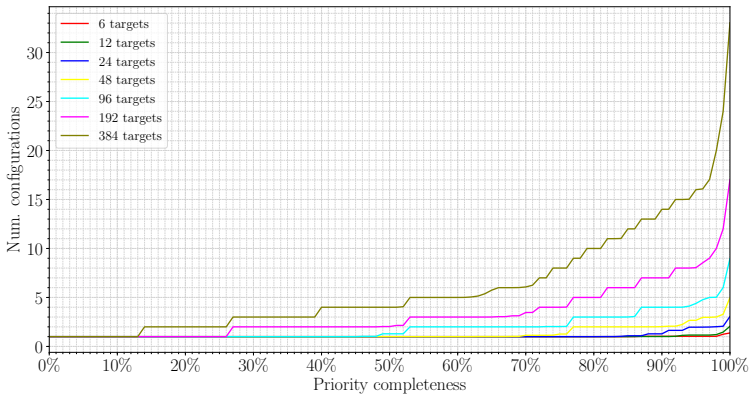
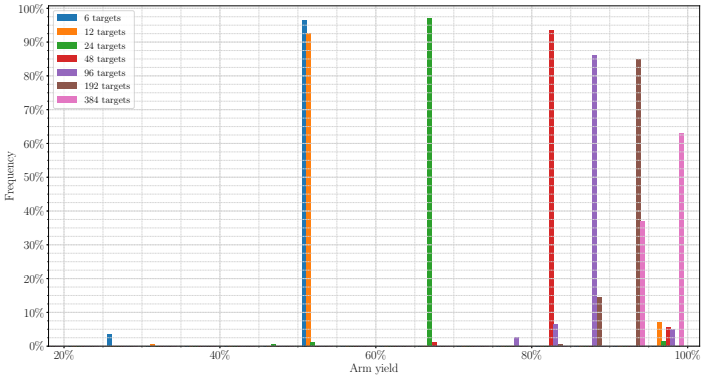


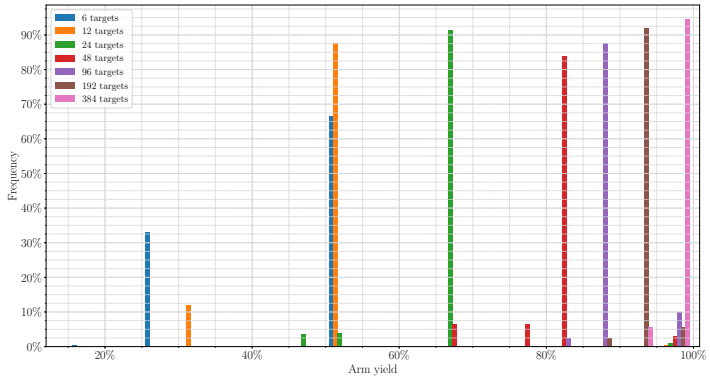
Figure 5.10: Average number of observational configurations versus priority completeness for SGPBTA.

for all cases where the number of sky targets is large. Certainly, all problem instances containing 96, 192 and 384 targets deliver yields ranging from 80% to 100%. Furthermore, when compared with the results obtained by the other allocator proposed, now, for scenarios with 384 targets, the frequency of instances residing in the 95%-100% bin has risen around 3%. Regarding the instances of 192 and 96 targets, the number of hits has also increased for the interval showing the highest yield possible. These increments are chiefly due to the greater patrolling range of the arms as well as the nature of the optimization method utilized. As stated before, this allocator delivers the best global plan in terms of the aggregated priority of each observational configuration. Therefore, the solver of the model prefers assigning arms to science targets rather than to blank sky since all the latter locations have priority 0. It can also be observed that there is no significant improvement in those scenarios with few targets. In these cases, the location of an object is more determinant than its priority value. The arm yield associated with every individual observational configuration regardless of the allocation plan where they were generated is illustrated in fig. 5.12.

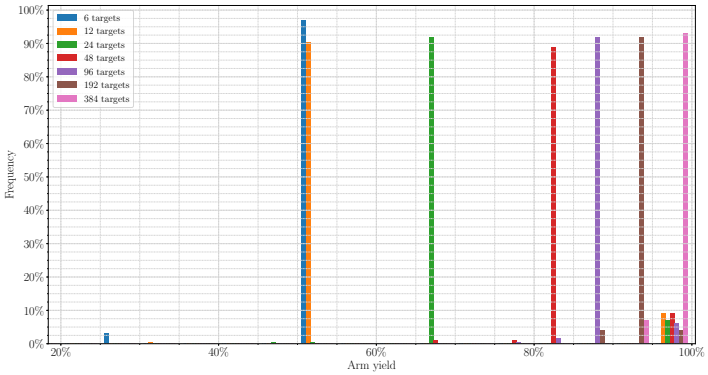
As confirmed by the results of experiments, the number of observational configurations required to achieve a particular field completeness value increases following the same trend as in the SGPBTA solution. Also, priority completeness behaves similarly to SGPBTA. Plots can be seen in fig. 5.9 and fig. 5.10 respectively. The major differences between SGPBTA and IPBA arise in the latter stages of completeness, especially in scenarios with 96, 192, and 384 targets. In table 5.1 and in table 5.2, comparisons are provided. There, we present the average, minimum, maximum, and theoretical number of configurations for four representative percentages of completeness.



(a) Arm yield for uniform distributions obtained with IPBA.

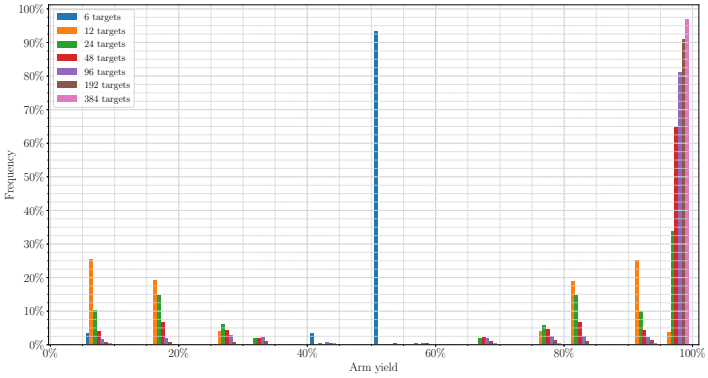


(b) Arm yield for heavily clustered distributions obtained with IPBA.

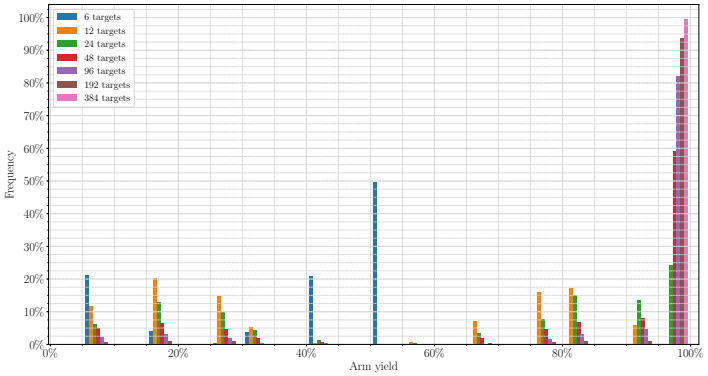


(c) Arm yield for weakly clustered distributions obtained with IPBA.

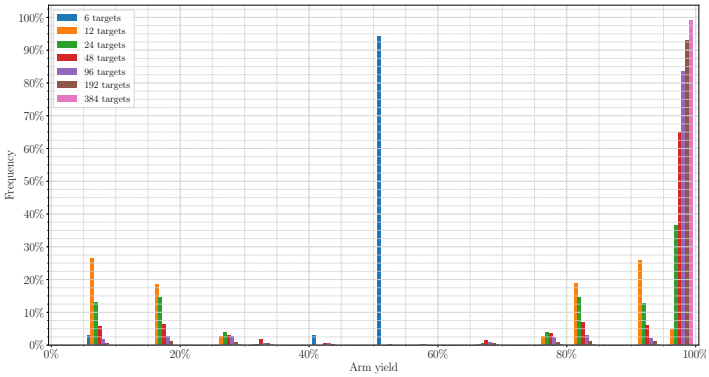
Figure 5.11: Allocation plan arm yield returned by IPBA.



(a) Arm yield for uniform distributions obtained with IPBA.



(b) Arm yield for heavily clustered distributions obtained with IPBA.



(c) Arm yield for weakly clustered distributions obtained with IPBA.

Figure 5.12: Observational configuration arm yield in IPBA.

Table 5.1: Detailed field completeness comparison between SGPBTA and IPBA. The differences between both allocators are highlighted in bold type.

		96 targets				192 targets				384 targets			
	Field Complet.	theo. #conf	avg. #conf	min. #conf	max. #conf	theo. #conf	avg. #conf	min. #conf	max. #conf	theo. #conf	avg. #conf	min. #conf	max. #conf
SGPBTA uniform	25%	2	2	2	2	4	4	4	4	8	8	8	8
	50%	4	4	4	4	8	8	8	8	16	16	16	16
	75%	6	**6.005**	6	**7**	12	12	12	12	24	24	24	24
	100%	8	**9.08**	8	**10**	16	**17.185**	**17**	**19**	32	**33.455**	**33**	35
IPBA uniform	25%	2	2	2	2	4	4	4	4	8	8	8	8
	50%	4	4	4	4	8	8	8	8	16	16	16	16
	75%	6	**6**	6	**6**	12	12	12	12	24	24	24	24
	100%	8	**8.582**	8	**9**	16	**16.893**	**16**	**18**	32	**32.511**	**32**	35
SGPBTA heavy.	25%	2	2	2	2	4	4	4	4	8	8	8	8
	50%	4	4	4	4	8	8	8	8	16	16	16	16
	75%	6	6	6	6	12	12	12	12	24	24	24	24
	100%	8	**9.05**	8	**10**	16	**17.055**	16	**18**	32	**33.13**	32	**34**
IPBA heavy.	25%	2	2	2	2	4	4	4	4	8	8	8	8
	50%	4	4	4	4	8	8	8	8	16	16	16	16
	75%	6	6	6	6	12	12	12	12	24	24	24	24
	100%	8	**8.471**	8	**9**	16	**16.896**	16	**17**	32	**32.732**	32	**33**
SGPBTA weak.	25%	2	2	2	2	4	4	4	4	8	8	8	8
	50%	4	4	4	4	8	8	8	8	16	16	16	16
	75%	6	6	6	6	12	12	12	12	24	24	24	24
	100%	8	**8.995**	8	**10**	16	**17.035**	16	**18**	32	**33.06**	32	**34**
IPBA weak.	25%	2	2	2	2	4	4	4	4	8	8	8	8
	50%	4	4	4	4	8	8	8	8	16	16	16	16
	75%	6	6	6	6	12	12	12	12	24	24	24	24
	100%	8	**8.594**	8	**9**	16	**16.322**	16	**17**	32	**32.487**	32	**33**

5.5.4 Real Scenarios

In this section, we demonstrate the performance of the proposed solutions with real targets. Specifically, we test the algorithms with four scenarios representative of the MIRADAS use cases:

1. LBV 1806-20, a candidate luminous blue variable (LBV) located about 28000 light-years (8700 pc) from the Sun, toward the center of the Milky Way.

2. NGC 869, also known as *h Persei*, is an open cluster at a distance of over 7000 light-years, located in the constellation of Perseus.

3. M13, also known as Messier 13, is a prominent globular in Hercules composed of several hundred thousand stars.

4. Sgr A, or Sagittarius A, is an astronomical feature at the center of the Milky Way composed by a supernova, a spiral structure and a very bright compact radio source at the center of the spiral.

The different targets selected from each of the four previously enumerated astronomical objects were converted from equatorial coordinates to standard coordinates employing the MI-

Table 5.2: Detailed priority completeness comparison between SGPBTA and IPBA. The differences between both allocators are highlighted in bold type.

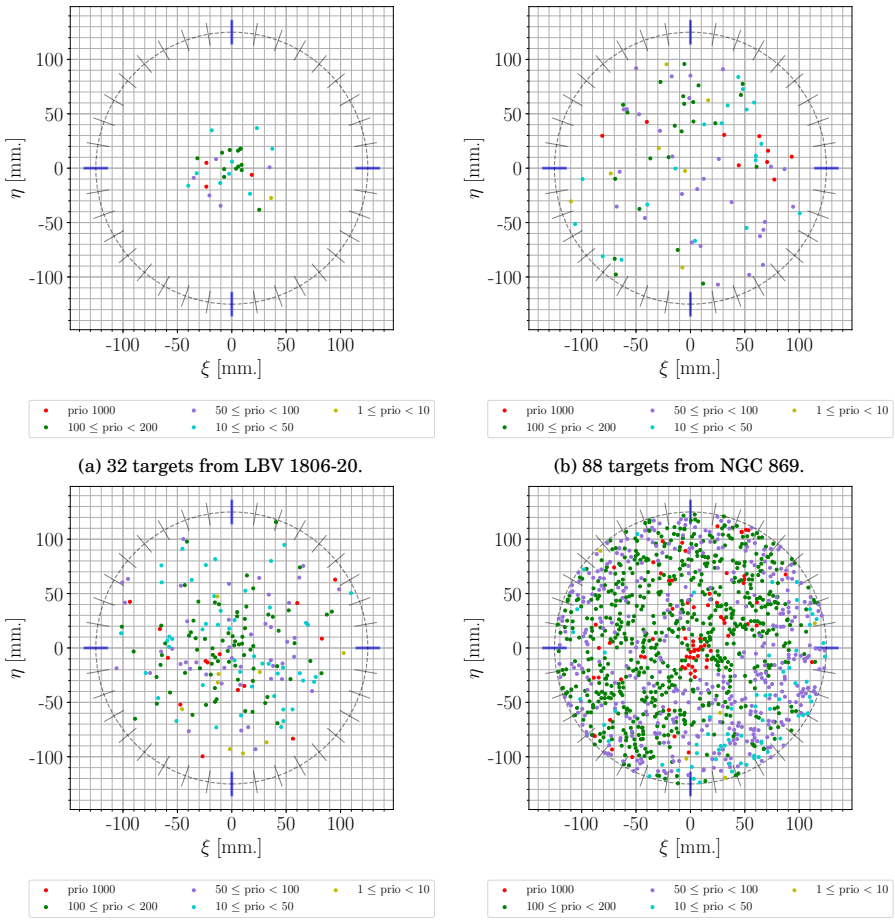
		96 targets				192 targets				384 targets			
	Priority Complet.	theo. #conf	avg. #conf	min. #conf	max. #conf	theo. #conf	avg. #conf	min. #conf	max. #conf	theo. #conf	avg. #conf	min. #conf	max. #conf
SGPBTA uniform	25%	1	1	1	1	1	1	1	1	2	2	2	2
	50%	1	**1.29**	1	**2**	2	**2.03**	2	**3**	4	4	4	4
	75%	2	**2.03**	2	**3**	4	4	4	4	8	8	8	8
	100%	8	**8.985**	8	**10**	16	**17.035**	16	**18**	32	**33.07**	32	**34**
IPBA uniform	25%	1	1	1	1	1	1	1	1	2	2	2	2
	50%	1	**1**	1	**1**	2	**2**	2	**2**	4	4	4	4
	75%	2	**2**	2	**2**	4	4	4	4	8	8	8	8
	100%	8	**8.325**	8	**9**	16	**16.886**	16	**17**	32	**32.974**	32	**33**
SGPBTA heavy.	25%	1	1	1	1	1	1	1	1	2	2	2	2
	50%	1	**1.32**	1	**2**	2	**2.135**	2	**3**	4	4	4	4
	75%	2	**2.065**	2	**3**	4	**4.01**	4	**5**	8	8	8	8
	100%	8	**8.95**	8	**10**	16	**17.005**	16	**18**	32	**33.045**	32	**34**
IPBA heavy.	25%	1	1	1	1	1	1	1	1	2	2	2	2
	50%	1	**1**	1	**1**	2	**2**	2	**2**	4	4	4	4
	75%	2	**2**	2	**2**	4	**4**	4	**4**	8	8	8	8
	100%	8	**8.715**	8	**9**	16	**16.904**	16	**17**	32	**32.643**	32	**33**
SGPBTA weak.	25%	1	1	1	1	1	1	1	1	2	2	2	2
	50%	1	**1.2**	1	**2**	2	**2.055**	2	**3**	4	4	4	4
	75%	2	**2.04**	2	**3**	4	4	4	4	8	8	8	8
	100%	8	**8.97**	8	**10**	16	**16.96**	16	17	32	**32.91**	32	**34**
IPBA weak.	25%	1	1	1	1	1	1	1	1	2	2	2	2
	50%	1	**1**	1	**1**	2	**2**	2	**2**	4	4	4	4
	75%	2	**2**	2	**2**	4	4	4	4	8	8	8	8
	100%	8	**8.15**	8	**9**	16	**16.12**	16	17	32	**32.03**	32	**33**

RADAS plate scale. The location of the targets and their priorities can be seen in fig. 5.13.

5.5.4.1 SGPBTA

As with previous synthetic fields, we also conduct all experiments with $\alpha = 0.2$ and $\alpha = 0.8$ and utilize the same configuration of the algorithm detailed in section 5.5.3.1. For each of the scenarios in fig. 5.13, the SGPBTA algorithm was run 20 times with different random seeds in every execution.

Contrarily to the experiments with probabilistic targets, the results concerning the arm yield of the allocation plans for *SGPBTA 0.2* and *SGPBTA 0.8* are this time significantly different. The results, as seen in fig. 5.14a, are represented in a histogram whose bins correspond to intervals of 5% arm yield. For the LBV 1806-20 field, the resulting arm yield in both *SGPBTA 0.2* and *SGPBTA 0.8* are the same. The only difference is the frequency, being greater in *SGPBTA 0.8* for the interval representing the highest yield delivered in both cases. In those scenarios comprising the largest number of targets, Sgr A and M13, all plans obtained by *SGPBTA 0.8* are in a bin immediately superior to the one in *SGPBTA 0.2*. Exactly, with *SGPBTA 0.8* all instances belong to the interval 95%-100%. However, the performance of *SGPBTA 0.8* for NGC 869 is poorer than the one returned by *SGPBTA 0.2*.



(a) 32 targets from LBV 1806-20.

(b) 88 targets from NGC 869.

(c) 195 targets from M13.

(d) 1219 targets from Sgr A.

Figure 5.13: Real science scenarios.

Table 5.3: Arm yield for 100% field completeness comparison.

	LBV 1806-20			NGC 869			M13			Sgr A		
	avg.	min.	max.	avg.	min.	max.	avg.	min.	max.	avg.	min.	max.
SGPBTA 0.2	0.606	0.533	0.666	0.886	0.815	0.916	0.908	0.903	0.956	0.939	0.931	0.949
SGPBTA 0.8	0.646	0.533	0.666	0.861	0.815	0.916	0.956	0.956	0.956	0.972	0.967	0.986
IPBA	0.888	0.888	0.888	0.916	0.916	0.916	0.956	0.956	0.956	0.986	0.986	0.986

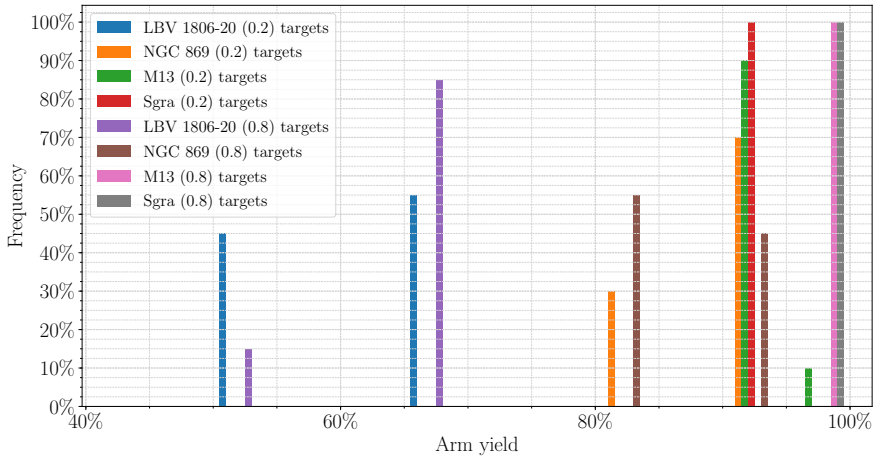
In fig. 5.14b, the performance of the algorithms considering the arm yield per observational configuration can be appreciated. In all cases but NGC 869, the results obtained by *SGPBTA 0.8* are better than those delivered by *SGPBTA 0.2*. Results for field completeness, as shown in fig. 5.15, follow the typical incremental step pattern, appreciated as well in synthetic fields. However, the performance for a 100% field completeness is better in *SGPBTA 0.8* since in every case the number of configurations required is smaller; see fig. 5.15. For instance, for the Sgr A field, with *SGPBTA 0.2* a total of three additional observational configurations were needed to achieve full completeness. These results were partly anticipated by the preliminary study in section 5.5.2. There, we saw how a high degree of randomness enables the exploration of the search space in a much broader way, consequently, producing solutions that would not have been found if following a pure greedy approach.

A visible pattern can be also appreciated in priority completeness, especially for the Sgr A scenario; see fig. 5.16. In the growth of priority completeness, regardless of the value of α utilized, three different stages are distinguished. Each of these stages corresponds to one of the three quasi-linear segments presenting different slope. The nature of every segment comes determined by the group of targets the algorithm is allocating. In the first, most targets selected belong to the set of sky locations having a priority equal to 1000. In the second, the solution allocates those targets whose priority ranges from 200 to around 50, whereas in the last segment, assignments are provided for the remaining ones.

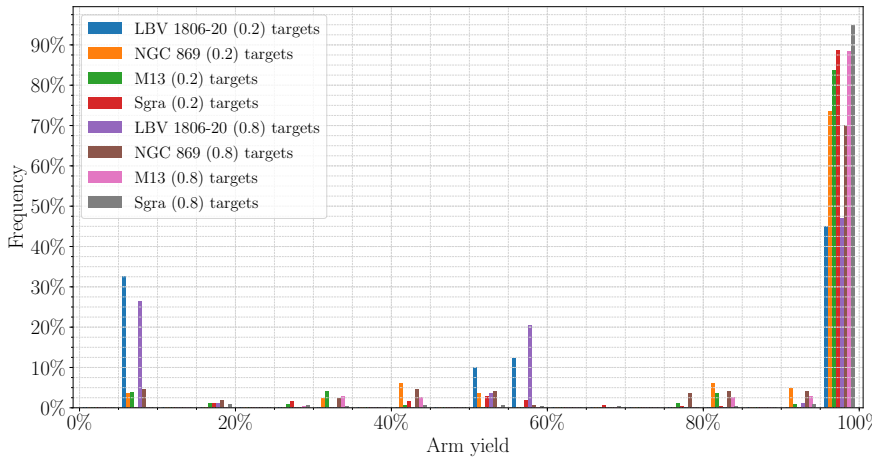
5.5.4.2 IPBA

IPBA provides a deterministic way to calculate observational configurations. In table 5.3, we compare the arm yield for 100% field completeness for *SGPBTA 0.2*, *SGPBTA 0.8* and IPBA. The data for both SGPBTA versions are obtained considering 20 different experiments for each scenario, using for each instance a different seed. As it is shown, IPBA is the allocator returning, in general, higher yields and the gap between IPBA and the other methods is smaller as greater is the number of targets in the field.

In table 5.4, a comparison between the two variants of SGPBTA and IPBA is shown for several values of field completeness. As it can be shown, the efficiency of IPBA is higher than the SGPBTA methods for those scenarios with a relatively small number of targets (LBV 1806-20 and NGC 869) and differences decrement as the number of targets increment. For instance, for

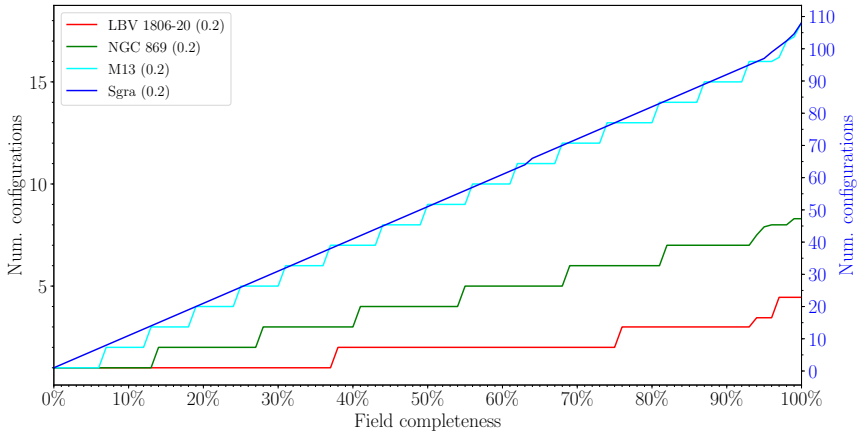


(a) Allocation arm yield using *SGPBTA 0.2* and *SGPBTA 0.8*.

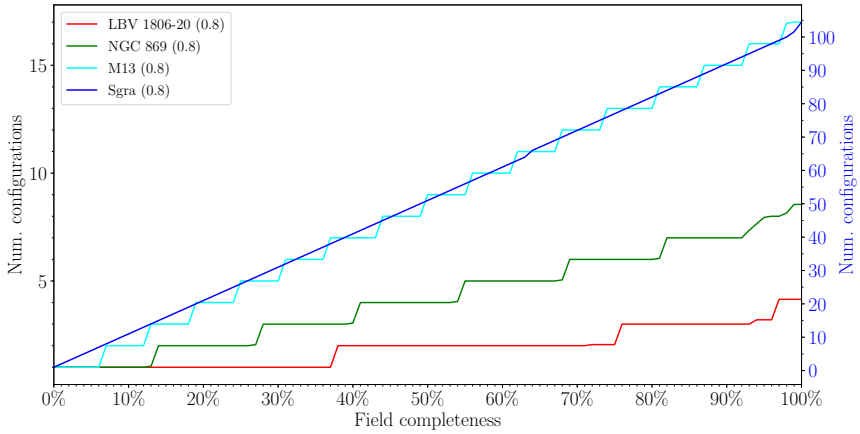


(b) Arm yield per configuration using *SGPBTA 0.2* and *SGPBTA 0.8*.

Figure 5.14: Allocation plan arm yield and yield per configuration for real scenarios using SGPBTA.

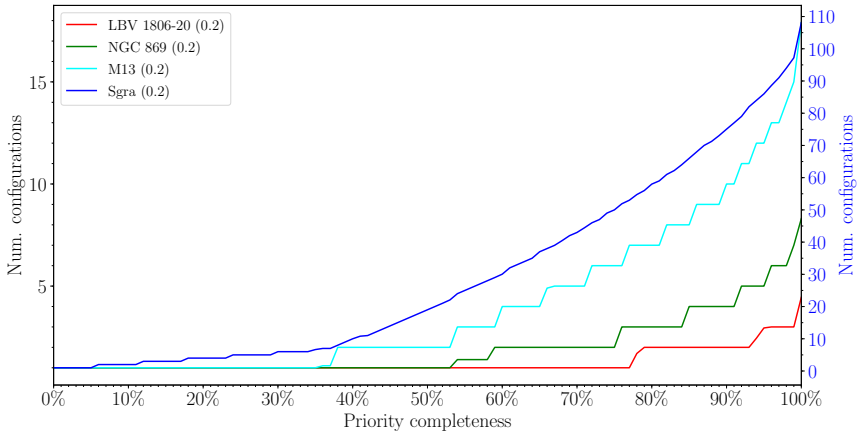


(a) Field completeness for *SGPBTA 0.2*. The number of configurations in Sgr A are expressed according to the scale displayed at the right vertical axis (blue).

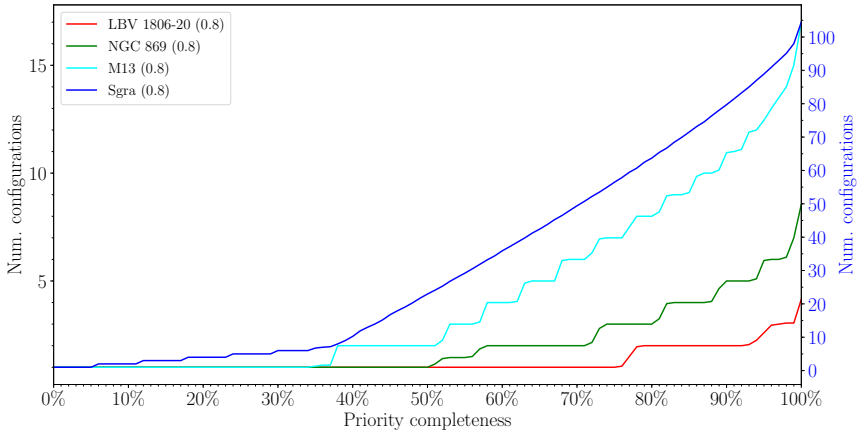


(b) Field completeness for *SGPBTA 0.8*. The number of configurations in Sgr A are expressed according to the scale displayed at the right vertical axis (blue)

Figure 5.15: Average number of observational configurations versus field completeness for SGPBTA.



(a) Priority completeness for *SGPBTA 0.2*. The number of configurations in Sgr A are expressed according to the scale displayed at the right vertical axis (blue).



(b) Priority completeness for *SGPBTA 0.8*. The number of configurations in Sgr A are expressed according to the scale displayed at the right vertical axis (blue).

Figure 5.16: Average number of observational configurations versus priority completeness for SGPBTA.

Table 5.4: Detailed field completeness comparison between the allocators proposed. The differences are highlighted in bold type.

		LBV 1806-20				NGC 869				M13				Sgr A			
	Field Complet.	theo. #conf	avg. #conf	min. #conf	max. #conf	theo. #conf	avg. #conf	min. #conf	max. #conf	theo. #conf	avg. #conf	min. #conf	max. #conf	theo. #conf	avg. #conf	min. #conf	max. #conf
SGPBTA 0.2	25%	1	1	1	1	2	2	2	2	5	5	5	5	26	26	26	26
	50%	2	2	2	2	4	4	4	4	9	9	9	9	51	51	51	51
	75%	2	**2**	2	**2**	6	6	6	6	13	13	13	13	77	77	77	77
	100%	3	**4.45**	**4**	**5**	8	**8.3**	8	**9**	17	**17.9**	17	**18**	102	**108.1**	**107**	**109**
SGPBTA 0.8	25%	1	1	1	1	2	2	2	2	5	5	5	5	26	26	26	26
	50%	2	2	2	2	4	4	4	4	9	9	9	9	51	51	51	51
	75%	2	**2.05**	2	**3**	6	6	6	6	13	13	13	13	77	77	77	77
	100%	3	**4.15**	**4**	**5**	8	**8.55**	8	**9**	17	**17**	17	**17**	102	**104.45**	**103**	**105**
IPBA	25%	1	1	1	1	2	2	2	2	5	5	5	5	26	26	26	26
	50%	2	2	2	2	4	4	4	4	9	9	9	9	51	51	51	51
	75%	2	**2**	2	**2**	6	6	6	6	13	13	13	13	77	77	77	77
	100%	3	**3**	**3**	**3**	8	**8**	8	**8**	17	**17**	17	**17**	102	**104**	**104**	**104**

Table 5.5: Detailed priority completeness comparison between the allocators proposed. The differences are highlighted in bold type.

		LBV 1806-20				NGC 869				M13				Sgr A			
	Field Complet.	theo. #conf	avg. #conf	min. #conf	max. #conf	theo. #conf	avg. #conf	min. #conf	max. #conf	theo. #conf	avg. #conf	min. #conf	max. #conf	theo. #conf	avg. #conf	min. #conf	max. #conf
SGPBTA 0.2	25%	1	1	1	1	1	1	1	1	1	1	1	1	5	5	5	5
	50%	1	1	1	1	1	1	1	1	2	2	2	2	19	**19**	**19**	**19**
	75%	1	1	1	1	2	**2**	**2**	**2**	6	**6**	**6**	**6**	50	**50**	**50**	**50**
	100%	3	**4.45**	**4**	**5**	8	**8.3**	8	**9**	17	**17.9**	17	**18**	102	**108.1**	**107**	**109**
SGPBTA 0.8	25%	1	1	1	1	1	1	1	1	1	1	1	1	5	5	5	5
	50%	1	1	1	1	1	1	1	1	2	2	2	2	19	**22.95**	**22**	**23**
	75%	1	1	1	1	2	**3**	**3**	**3**	6	**7**	**7**	**7**	50	**56.45**	**56**	**57**
	100%	3	**4.15**	**4**	**5**	8	**8.55**	8	**9**	17	**17**	17	**17**	102	**104.45**	**103**	**105**
IPBA	25%	1	1	1	1	1	1	1	1	1	1	1	1	5	5	5	5
	50%	1	1	1	1	1	1	1	1	2	2	2	2	19	**19**	**19**	**19**
	75%	1	1	1	1	2	**2**	**2**	**2**	6	**6**	**6**	**6**	50	**50**	**50**	**50**
	100%	3	**3**	**3**	**3**	8	**8**	8	**8**	17	**17**	17	**17**	102	**104**	**104**	**104**

the Sgr A field, the best result is obtained by IPBA, and the average of all instances of *SGPBTA 0.8* is close to the IPBA result.

The method that best concentrates the priority for all values of priority completeness as well as for all fields under test is IPBA, as seen in table 5.5. Regarding the SGPBTA variants, the one having a behavior closer to a greedy approach shows better efficiency in all completeness values except for 100%.

A remarkable characteristic of the linear model implemented in IPBA is that it requires two pre-computed matrix to work. The first of them stores all those targets that can be reached by each arm in the system. This data structure is similar to the one needed for SGPBTA as well. However, the second refers to the specific incompatibility matrix, which contains all pairwise assignments producing a collision. In part, the computational complexity of IPBA comes determined by the complexity of determining such a matrix. This complexity is proportional to $\frac{(n\times a)\times[(n-1)\times(a-1)]}{2}$, where n denotes the number of different focal plane locations to consider and a refers to the number of arms in the system. Taking into account that in large fields, the number

of sky locations is much greater than the arms available, then we could say that processing time is quadratic in the number of targets. Specifically, its computation can vary from 0.256 s for 32 targets up to 102.761 s for about the 1200 targets in the largest scenario tested. Moreover, another aspect impacting the run time of this solution is the number of variables x_{jk} -see eq. (5.17)- to consider by the solver of the model. The complexity of the problem to solve is proportional to the number of targets awaiting assignment. Hence, run time, initially, comes determined by the total number of targets in the field and incrementally decrements as targets are allocated. By way of illustration, the time to compute an observational configuration ranges from a few milliseconds in the case of LBV 1806-20 (32 targets) to almost minutes in Sgr A (more than 1200 targets). For the Sgr A scenario, more than 100 observational configurations are required to achieve 100% field completeness, consequently demanding the equal number of executions of the allocator and requiring a total processing time of more than 6 min. The processing time of this allocator is good for LBV 1806-20 (0.75 s), NGC 869 (2 s) and M13 (6 s) fields, however, for datasets containing 300 or more targets, the time might be excessive. Therefore, making this approach impractical. In these cases, any of the SGPBTA variants is recommended since it returns an allocation plan in less than 1 min.

5.6 Conclusion

Assignment problems are everywhere. They arise in a range of fields such as transportation, healthcare, education, and sports. In fact, this is a well-studied topic under optimization or operations research branches. Concretely, in MIRADAS, the assignment problem is related to how to pair user-defined targets with probe arms of the MXS system. Traditionally, in order to address questions of this kind, several diverse approaches such as exact methods, heuristics, and metaheuristics have been proposed.

In this chapter, we presented two different solutions for the MIRADAS target allocation problem. Both share a common objective function: maximizing the total aggregated priority of the targets assigned. We first proposed SGPBTA, a metaheuristic inspired in the popular GRASP approach. Specifically, although it incorporated all the fundamental stages found in classical GRASP literature, we divided the construction block into two different phases. The first delivered solutions, which might be incomplete, where the arms involved are assigned to sky targets. Then, if this stage failed to allocate all arms available, in a second phase, we attempt to assign the pending arms to not conflicting blank sky positions.

Relying on mathematical optimization, the second method computes, unlike the metaheuristic approach, the optimal solution. In particular, we solved the problem utilizing a constrained integer program. Our contribution extended the model previously proposed in KMOS. Our method is able to consider: (i) a set of targets with cardinality different from that of the set of arms and (ii) the possible collisions between the default locations of the arms (park) and all other assignable

locations. To accomplish this, the proposed model uses two matrices. One of them specifies the reachability of the targets, while the other defines incompatibilities between each pair of potential *target-arm* assignments due to collisions. Both matrices need to be computed before solving the linear program, taking the latter a significant amount of time if the number of elements in the field is large. However, for scenarios with a few hundreds of targets, its computation, as well as the time required to solve the system, is irrelevant. In these cases, this method offers better results than the metaheuristic approach, and it should be used as a first option. Unfortunately, with more massive data sets the exact method is impractical, and the SGPBTA algorithm is preferred, delivering reasonable solutions at moderate computing costs. Finally, although the model currently only tries to maximize the aggregated priority of the targets allocated, it could be easily extended. For instance, further geometric constraints could be added to return assignments enabling the motion planning algorithm to compute better trajectories.

CHAPTER 6

MOTION PLANNING

One of the main tasks of the MIRADAS MXS control software is to provide an algorithmic procedure capable of translating high-level specifications into low-level descriptions. In our case, these high-level requirements refer to observation jobs, whereas the low-level is concerned with how to move the instrument manipulators. Commonly, in robotics and also in MOS instruments, this conversion task is conveniently divided into two distinct steps sequentially executed. The first of them, as we have seen in chapter 5, is responsible for assigning a job to each of the agents in the system. In astronomical instrumentation, it is generally termed as target allocation. However, sometimes, scientists specifically prefer to use the name of fiber allocation, if a MOS relies on fibers to collect light or arm allocation if the instrument is based on robotic arms. Then, once a concrete allocation plan has been found, the sequence of motions to be performed for each opto-mechanical device need to be carefully determined so that collisions between each other are avoided. This process in robotics is broadly referred to as *motion planning* or *trajectory planning*.

A classic example of motion planning is the popularly known as the Piano Mover's Problem (Schwartz and Sharir, 1983a,b,c). Given a precise model of the environment[1], the algorithm solving this problem must compute the sequence of actions to move the piano from one room to another without hitting anything. Since all obstacles to avoid are static, this collection of motions what specifies is a geometric path. However, if the obstacles are dynamic or more agents are moving in a shared environment, motion planning needs to be approached from a time-varying perspective. In this case, the planning algorithm has to determine a trajectory, which comprises a geometric path connecting an initial position and a destination as well as an associated time profile. In other words, a trajectory defines a path as a function of time -for instance, in terms of

[1] A house in this case.

the mechanism velocity and acceleration. When more than one mechanism or robot is involved in the task, then, the term *cooperative motion planning* or *multi-robot motion planning* (MRMP)[2] is more suitable. The question attempting to address now is the computation of trajectories for a group of robots so that all of them reach their goal positions starting from the initial ones. Additionally, physical limitations such as robots must not collide with each other, and they must avoid obstacles in the environment during their motions must be respected.

Unfortunately, there is very little information regarding cooperative motion planning in MOS since this is an entirely new research area in astronomical instrumentation. Most of the publications found by the authors are related to fiber-based instruments, which are conceptually different from MIRADAS. However, we include them here for completion. On the other hand, although some observatories have fiber and multi-slit based MOS instruments, at the moment, there is only one instrument similar to MIRADAS. As we have already mentioned before, it is KMOS, an integral field spectrograph for ESO's VLT that achieved first light in November 2012 (Sharples et al., 2013).

Parts of this chapter have previously been published as: "Roadmap search based motion planning for MIRADAS probe arms", Journal of Astronomical Telescopes, Instruments, and Systems 4(3), 034001, 2018. (Sabater et al., 2018a)

6.1 Related Work

As stated previously, the study of coordination techniques for multiple robots within the field of astronomical instrumentation is quite recent. With the advent of more and more sophisticated MOS instruments, this area of knowledge is gaining more attention. However, to the best of our knowledge, there are still very few works published. The more recent are those from the teams working on the following three different instruments: DESI (Edelstein et al., 2018), MOONS (Cirasuolo et al., 2014) and IRIS (Larkin et al., 2016). The first is a spectroscopic instrument employing 5000 fiber positioners to measure the effects of dark energy that will be installed in the Mayall 4m telescope at Kitt Peak National Observatory (Arizona, USA). The second, MOONS, is a MOS containing around 1000 fibers conceived for the 8.2m ESO's VLT at Paranal Observatory (Chile). Finally, IRIS is an infrared (0.84 - 2.4 micron) integral field spectrograph with three pick-off arms for the Thirty Meter Telescope (Hawaii, USA). All of them adopted the same obstacle avoidance traditional approach based on *artificial potential fields* (APF). They propose a decentralized method where each agent individually determines the path to follow by using its own APF as a navigation function.

The APF technique is often employed in the local path planning stages of some autonomous vehicles, being the real motions determined by a *gradient descent* optimization algorithm (Dolgov et al., 2008). It was initially developed to relocate some of the slow, high-level planning tasks to

[2] In literature, it is also referred to as motion planning for multi-robot systems or multi-robot path planning.

the lower, faster level of control in robotics (Hwang and Ahuja, 1992; Khatib, 1985). This was done by directly coupling sensing to an actuation algorithm, leading this way to a fast and instinctive response to immediate hazards. In fact, being this a reactive approach focusing on the short term, it has been traditionally complemented by a global path planning algorithm (Arnay et al., 2016; Morales et al., 2016). This upper-level planner is in charge of finding a path connecting an origin and a destination, whereas the local one attempts to follow it introducing small modifications, if necessary, at each instant to accommodate the motion to the changing environment.

APF combines attractive forces leading the mechanism through the space toward its goal with repulsive ones pushing it away from obstacles. In the context of a multi-robot system, these obstacles are other robots moving in the same environment. Although the idea behind is very alluring due to its simplicity, as it is well-known, the APF approach presents a few drawbacks (Koren and Borenstein, 1991; Snyman, 2005). It suffers from getting trapped in local minima. This problem can arise due to the particular workspace or to the incorrect selection of the weight coefficients for the distinct potential fields (Bounini et al., 2017). Furthermore, the method can produce oscillatory motion plans in the presence of multiple obstacles or narrow passages. In these scenarios, the attractive forces cancel the repulsive ones when they have a similar magnitude and, in addition, they are collinear but in the opposite direction (Amiryan and Jamzad, 2015).

The occurrence of local minimum is highly dependent on the geometry of the problem at hand. Therefore, no general solution exists. For instance, in DESI the APF guides a *decentralized navigation function* (DNF) in such a way that a single global minimum is guaranteed (Makarem et al., 2014a). However, such a solution is not valid for the positioners of MOONS. In this fiber-based MOS deadlocks might arise if two different positioners meet in the middle of their paths (Makarem et al., 2016; Tao et al., 2018). If such is the case, the corresponding robot repulsive forces will be activated, and consequently blocking the positioners. At that point, each positioner is waiting for the other to get away so that it can continue its motion toward its destination. The authors in Tao et al. (2018) attempt to minimize some of the deadlocks by employing a particular state machine with heterogeneous zones of avoidance. In practice, this approach is equivalent to give to the positioners different priorities. As they assert, this technique avoids some deadlocks but not prevent all of them. On the other hand, for the polar coordinate probe arms in IRIS, the general guidelines widely used to effectively apply DNF were not suitable due to the mechanism particular geometry (Chapin et al., 2016). However, local minima are skipped by temporarily modifying the vector field of the arms. Each time a conflicting situation is encountered, a new component scaling the gradient of the repulsive potential by a given factor is applied but orthogonally.

Traditionally, researchers involved in the study of motion planning have only concentrated on the geometrical nature of the robot movement, without carefully considering dynamics. Therefore, in the vast literature related to robotics, the authors determine trajectories where the robots are either allowed to change their velocities instantaneously or are performing an

inconveniently large number of motions. Those are well-known assumptions in the community since the first works in this field (LaValle and Hutchinson, 1998). Nevertheless, there is a growing body of research dedicated to *kinodynamic* planning, which determines trajectories considering both kinematic and dynamic constraints, including continuous velocities profiles. Indeed, the kinodynamic planning problem is formulated in a state-space similar to the one we will see in section 6.2.1.1 but it contains first-order differential constraints as well (LaValle and Kuffner, 2001). The reader is referred to Akella and Peng (2004), Peng and Akella (2005), Hsu et al. (2002), and Phillips et al. (2004) for further details of this kind of planning. More specifically, in the APF-based planner employed in DESI, during the planning stage, dynamic constraints where neglected. Hence, an additional stage is utilized later to decrease the number of velocity profiles[3] required to execute the trajectories (Makarem et al., 2014b).

In recent times, as indicated in section 5.2, a new concept of fiber positioners has been developed (Gilbert et al., 2012; Staszak et al., 2016). Starbugs consist of a pair of concentric piezo-ceramic actuators freely moving throughout an instrument FOV. With their particular ability to rotate and translate, these mini robots currently form part of TAIPAN, a 150-fiber MOS for the UK Schmidt Telescope, and are expected to be in the MANIFEST instrument for GMT as well. The TAIPAN motion planner -the authors call it router- attempts to find a collision-free trajectory for each starbug from their current positions to their next targets (Goodwin et al., 2014; Lorente et al., 2016). Besides, it finds two additional sets of trajectories: one moving the positioners from their current positions to their park positions and another to reach their associated targets from their park positions. Although they are individually controlled, all starbugs in the system move synchronously. A path is considered to be composed of a series of *ticks* -a predefined duration. Due to constraints in the electronics, during a tick, the entire group of starbugs must perform the same motion primitive. They will either rotate on their central axes, translate, or wait until the end of the tick. Since the system is synchronous, the motions associated with the tick n do not start until the tick $n-1$ has finished. Therefore, the mini-robots always start the execution of every respective motion primitive from a stopped position.

The TAIPAN planning tool provides three different motion algorithms of increasing complexity. If the most straightforward algorithm cannot determine a trajectory, then the more computationally demanding ones are sequentially tried until either a complete motion plan is found or a particular target is marked as unreachable. The first of these series of planners is named *Simple Vector* and delivers a configuration time similar to a sequential positioning when succeeds. In this planner, wait states are not considered. Initially, simple paths[4] connecting the starting and final positions for each starbug are calculated, and those not crossing are moved first to their destinations. Next, every line-of-sight conflicting pairs are attempted, with one of the starbugs in the pair moved first and then the other. Finally, the path for any remaining bug is tried and those for which a valid path could not be found proceed to the next planner known as

[3] In this context, a velocity profile can be considered to be similar to a motor controller command.
[4] Mainly, straight line paths.

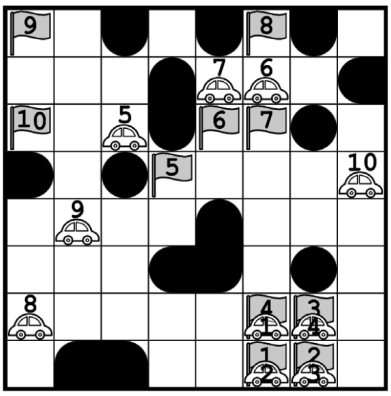


Figure 6.1: Grid-based representation of the environment. Cars represent initial positions, whereas flags denote destinations. Obstacles are in black. Note that the complete footprint of every agent fits in a cell of the grid. *Image Credit:* Standley and Korf (2011).

Traffic Lights. This second planner introduces no-motion primitives (wait states), so any robot can wait somewhere along its simple path for one or more ticks to avoid collisions. However, unavoidable collisions arise when either the start or end positions of one bug are too close to the path of another bug. The final planner is based on the Cooperative A^* (CA^*) technique and its associated spatio-temporal reservation table.

When only a single moving robot is present, the pathfinding problem can be effectively solved using the A^* algorithm (Hart et al., 1968). However, when multiple agents move in a shared environment, extra care must be taken to avoid that two agents occupy the same space at the same time. That problem is often known in literature as Cooperative Path Finding Problem (CPFP). The world in this problem is often modeled as a grid, and a general assumption is that each agent occupies a single cell of the world (Standley and Korf, 2011); see fig. 6.1. This fundamental premise tends to be valid in mobile robotics, where: (i) the geometry of the robot can be effectively abstracted by its corresponding footprint and (ii) this footprint is much more smaller than the environment where the robot moves. In that case, then the CA^* algorithm provides a decoupled approach to CPFP (Silver, 2005).

In CA^*, the Cartesian space where the agents move is discretized and an additional time dimension is added to that map. Then, the motion planning task is decoupled into a series of single-agent searches. The algorithm sequentially calculates paths for each agent individually. Once a path for the current agent is found, that path is marked into the corresponding global <*time-step, location*> pairs of the reservation table. Subsequent agents cannot conflict with the reserved paths; see fig. 6.2. The concept of using such a reservation table has also been employed to enable multiple vehicles to move on a shared crossroad (Dresner and Stone, 2008).

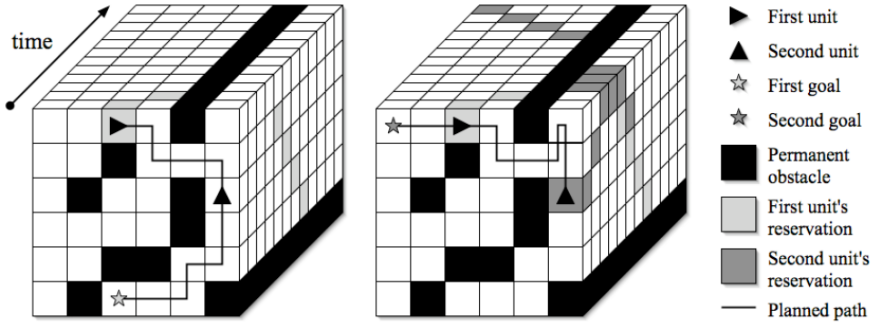


Figure 6.2: Sequential path finding for two different agents. (a) The first unit searches for a path and marks it into the reservation table. (b) The second unit searches for a path, taking into account the used slots, and also marks it into the reservation table. Note that this approach implicitly assumes that both units move at the same speed. *Image Credit: Silver (2005).*

As all decoupled approaches, CA^* is fast, but, unfortunately, incomplete[5]. Therefore, as stated in Lorente et al. (2016), although this strategy has been found successful in a large number of TAIPAN use-cases, some test configurations remain unsolved.

Finally, we will turn our attention to KMOS. As commented before, this is the instrument in operation more similar to MIRADAS. Both instruments employ deployable IFUs in the form of probe arms distributed around a circular bench to observe several user-defined targets simultaneously. However, technically, they are different. First, the mechanical design of KMOS probe arms makes the task of positioning the arm in a given focal plane location more intuitive than in MIRADAS. The KMOS mechanism presents a polar (r, ϕ) motion approach, while in MIRADAS, as we have seen in chapter 3, the structure of the arm is more complex, resulting in more elaborated motions. Second, the target allocation and motion planning seem to be naturally coupled in KMOS. The constraint of collision-free trajectories is embedded in the assignment phase. Specifically, the objective function to minimize is the sum over the distances from the pivot axis of each arm to its assigned target for all possible assignment pairs. This approach results from the fact that when this sum is minimal, then the arms do not cross. But, that condition is only satisfied if the arms are dimensionless. In consequence, assignments are checked afterward for collision, considering now the real arm dimensions. If any incompatible assignment is detected, then it is discarded and the target treated in a subsequent iteration of the allocator. Hence, in theory, the allocator delivers feasible plans in terms of collisions.

[5] An algorithm is considered to be incomplete if it is not able to find a correct solution even though one exists.

6.2 Motion Planning in Robotics

For completeness, we also include a general overview of the topic related to general robotics. In the 1960s, Nilsson wrote one of the first documented studies in the field (Nilsson, 1969). Since then, due to the increasing proliferation of robotic elements in everyday life, the scientific community has produced an enormous number of papers. It is difficult to provide comprehensive coverage of all motion planning methods, especially for those related to multi-robot systems. In LaValle (2006) and Choset et al. (2005), two books devoted to the particular case of single robot motion planning, an introduction to the multi-robot problem is given. For a review of MRMP, see Yasuda (2011) and the references of Wagner and Choset (2015).

Although a few of them have been briefly introduced in chapter 3, in the following subsections, we discuss more thoroughly some motion planning key concepts. Next, an overview of the more representative single robot techniques is given, and finally, we review some particular methods specific for multi-robot systems. It is important to remark that we focus only on methods addressing the MRMP in a centralized fashion. In MIRADAS, there is no way to solve the problem in a distributed manner since each arm cannot be considered as an autonomous agent capable of sensing the environment and communicating with the others. Contrarily, the individual robots merely execute the centrally determined plan.

6.2.1 The State Space

A robot $\mathscr{A}$ is a rigid body moving in the real world, a Euclidean space denoted by $\mathscr{W}$ that is a subset of $\mathbb{R}^2$ or $\mathbb{R}^3$. In a few scenarios such as assembly lines, the robot tasks are commonly specified in terms of the environment $\mathscr{W}$ where the robot resides. For instance, the robot might need to pick up a few objects placed in predetermined points of the environment, or the end-effector might need to follow a linear path. However, if there are no real-world constraints, motion planning can be effectively carried out in the *state space*. Hence, a proper, implicit, and finite representation of the robot and the environment obstacles in the *state space* is indispensable.

6.2.1.1 The Configuration Space

A configuration q is the minimal set of parameters which univocally determine the *placement* (position and orientation) of each element of $\mathscr{A}$ in $\mathscr{W}$. The space of all possible configurations of $\mathscr{A}$ defines the configuration-space $\mathscr{C}$. In an articulated mechanism, the dimensionality of q is identical to the degrees-of-freedom of the mechanism. Thus, $\mathscr{C}$ can be expressed by the Cartesian product of the n distinct p_i parameters contained in a *n-dimensional* configuration q:

$$\mathscr{C} = p_0 \times p_1 \times \ldots \times p_n \tag{6.1}$$

where $\mathscr{C}$ is bounded by upper and lower limits on each of the composing p_i. The points of a particular configuration space can be grouped according to their safety. The set of those

configurations making the mechanism collide with an obstacle is known as *C-space obstacle* and denoted by $\mathscr{C}_{obs}$. The set of all safe configuration is represented by $\mathscr{C}_{free}$.

Using the state space defined in eq. (6.1), the motion problem of an object living in $\mathscr{W}$ can be effectively reduced to finding a continuous path connecting two points in the configuration-space. Besides, if an implicit representation of $\mathscr{C}_{obs}$ is obtained, which is not a trivial task, there is no need to check for collisions while planning.

6.2.2 The Canonical Problem for a Single Robot

Let $\mathscr{A}$ be a robot moving in an environment $\mathscr{W}$ containing an area $\mathscr{O} = \{\mathscr{O}_1, \mathscr{O}_2 \ldots \mathscr{O}_n\}$ formed by n static and closed obstacles. Then, the canonical problem can be formulated as follows:

- Given a starting and a final position of $\mathscr{A}$ in $\mathscr{W}$, find if exists a path τ between them while avoiding collisions of $\mathscr{A}$ with $\mathscr{O}$, or, otherwise, report a failure. This path τ will contain a continuous sequence of positions of $\mathscr{A}$ in $\mathscr{W}$.

Using the concept of *configuration-space*, the position of every point of the robot $\mathscr{A}$ in $\mathscr{W}$ can be advantageously represented by a particular $q \in \mathscr{C}$. Let $\mathscr{A}(q)$ denote the subset of $\mathscr{W}$ occupied by $\mathscr{A}$ at a particular configuration $q \in \mathscr{C}$. Then, the representation of the obstacle area $\mathscr{O} \subseteq \mathscr{W}$ in the configuration-space can be expressed as:

$$\mathscr{C}_{obs} = \{q \in \mathscr{C} \mid \mathscr{A}(q) \cap \mathscr{O} \neq \emptyset\} \tag{6.2}$$

while the set of configurations which avoid collision, $\mathscr{C}_{free}$, can be defined as:

$$\mathscr{C}_{free} = \mathscr{C} \setminus \mathscr{C}_{obs} \tag{6.3}$$

Once defined $\mathscr{C}_{free}$, the canonical problem can be conveniently expressed in terms of the configuration-space:

- Given a starting configuration q_s and a final configuration q_f, both belonging to the $\mathscr{C}_{free}$ of a robot $\mathscr{A}$, find if exists a continuous path τ in $\mathscr{C}_{free}$ between them, or report failure if no such path exists.

Conceptually, the problem of computing a continuous path τ in configuration-space from q_s to q_f can be formulated as follows:

$$\tau : r \in [0,1] \rightarrow \tau(r) \in \mathscr{C}_{free} \tag{6.4}$$

with $\tau(0) = q_s$ and $\tau(1) = q_f$.

In practice, the continuous function τ is transformed into a discrete one (τ_d) which is defined by a finite sequence of configurations $q_i \in \mathscr{C}_{free}$:

$$\tau_d = \{q_s, q_1, q_2, \ldots, q_f\} \tag{6.5}$$

6.2.2.1 Complexity

In 1979, it was shown that the complexity of the Piano's Mover Problem was PSPACE-hard[6] in the cardinality of $\mathscr{C}$ (Reif, 1979). Some researchers, inspired by the idea of *C-space*, proposed a series of solutions, all of them requiring the explicit representation of $\mathscr{C}_{free}$ in a convenient form. Schwartz and Sharir (1983b) proposed an algorithm called the *cell decomposition* that recursively partitions the $\mathscr{C}$ into a finite number of cylindrical cells so that $\mathscr{C}_{free}$ is a subset of those cells. Although complete[7], the running time of their approach is doubly exponential on the problem dimensionality. Later, Canny (1988) presented a single exponential-time algorithm based on roadmaps, a subspace capturing the connectivity of $\mathscr{C}_{free}$, and established the problem is PSPACE-complete. However, this approach is impractical as a direct computation of $\mathscr{C}_{free}$ and $\mathscr{C}_{obs}$ can be extremely challenging in some particular scenarios.

6.2.2.2 Building $\mathscr{C}_{obs}$

Exact motion planning algorithms rely on an explicit representation of $\mathscr{C}_{obs}$; however, its construction is not straightforward when working with real manipulators. In the simple case of mobile robots that are only capable of translating, the boundary of $\mathscr{C}_{obs}$ can be obtained by sliding the robot around the obstacles. Let define the *Minkowski sum* of the set $X \in \mathbb{R}^n$ and the set $Y \in \mathbb{R}^n$ as follows:

$$X \oplus Y = \{x + y \in \mathbb{R}^n \mid x \in X, y \in Y\} \tag{6.6}$$

Then, with the help of the above expression, $\mathscr{C}_{obs}$ can be constructed by convoluting the two following sets

$$\mathscr{C}_{obs} = \mathscr{O} \oplus (-\mathscr{A}(0)) \tag{6.7}$$

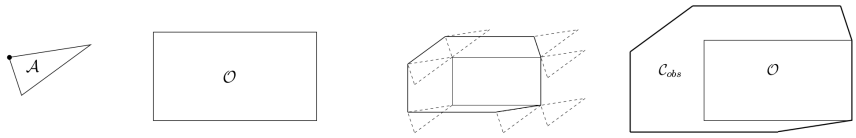
where $-\mathscr{A}(0)$ is the robot $\mathscr{A}$ reflected at its origin [8]. Figure 6.3 illustrates how this convolution determines $\mathscr{C}_{obs}$. In those scenarios where $\mathscr{W} \subseteq \mathbb{R}^2$ and $\mathscr{O}$ and $\mathscr{A}$ are convex polygonal objects, the simple *star algorithm* can be also employed (LaValle, 2006; Lozano-Pérez, 1983). However, when mobile robots are permitted to rotate, the representation of $\mathscr{C}_{obs}$ becomes much more complicated due to the non-linearity constraints imposed by the objects (Erdmann and Lozano-Pérez, 1986). Furthermore, if the robot consists of several rotating chains such as many commercial manipulators, a process based on semi-algebraic models exists, which is partially documented in LaValle (2006). But, given the complexity of such task and its associated computational cost, determining an exact mapping of $\mathscr{C}_{obs}$ is, in many cases, impractical.

In the 1990s, a few approximated approaches emerged (Wise and Bowyer, 2000). Among those, one which gained popularity is the grid of cells or *bitmaps*. There, $\mathscr{C}$ is discretized into a number of cells and they are properly labeled as *safe* or *forbidden*. Indeed, bitmaps are often

[6] PSPACE-hard problems require polynomial space to solve.

[7] An algorithm is considered to be complete if it determines a feasible solution, if one exists, in a finite time, otherwise, it returns a failure.

[8] The set $-\mathscr{A}(0)$ can be obtained by negating each value in $\mathscr{A}(0)$



(a) A triangular robot with the ability to translate and a rectangular obstacle.

(b) Sliding the robot around the obstacle while keeping them both in contact and the resulting $\mathscr{C}_{obs}$.

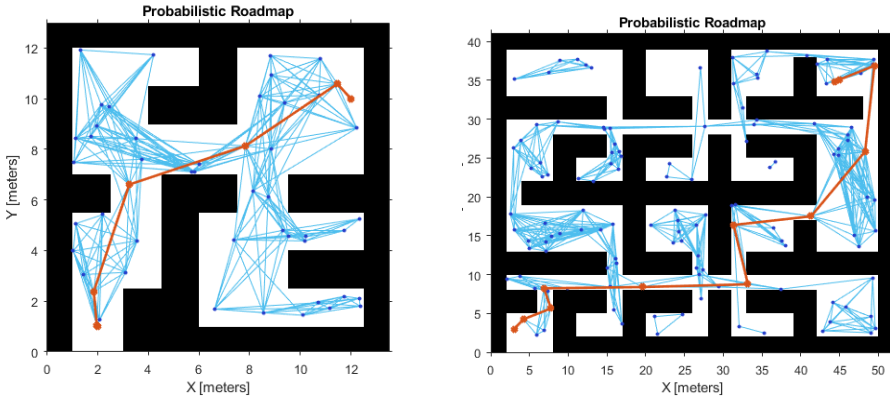
Figure 6.3: Construction of $\mathscr{C}_{obs}$. *Image Credit: LaValle (2006)*.

used in literature to represent dynamic environments. The great challenge here, however, is how to effectively reconstruct the bitmap every time a change in the environment is detected. Kavraki (1995), which addresses the problem for a 2-dimensional mobile robot able to translate and rotate, rebuilds it with the help of fast Fourier transforms to reduce the cost of the convolution. More recently, the authors in Lau et al. (2013) consider the problem for a 3D mobile robot whose shape is invariant. Sadly, the shape of a real manipulator is variable and depends on the configuration it is at.

6.2.3 Algorithms for Single Robot Motion Planning

The different algorithms developed in the field can be broadly categorized into *combinatorial* and *sampling-based*. While the former contains mainly theoretical approaches, offering efficient solutions to a reduced set of problems, the latter consists of practical approximations motivated by computational costs issues in real environments. We include a third family, those based on the potential-field technique, which does not fit into any of the two previous categories.

Potential fields This method is based on an obstacle avoidance approach (Hwang and Ahuja, 1992; Khatib, 1985) and achieves significant performance in complex environments (Ge and Cui, 2002). This heuristic method relies on a differentiable real-valued cost function that determines the direction of motion of the robot. That function consists of two different components. One of them is an attractive term, a metric estimating the distance from any configuration in $\mathscr{C}$ to the goal. The other is a repulsive term, penalizing those configurations leading to collisions with the environment; see Latombe (1991) for a detailed discussion of different functions. A path can be computed applying a gradient descend like optimization algorithm. Despite being practical, this approach is only able to find local minima (Koren and Borenstein, 1991). Therefore, some extensions such as navigation functions (Rimon and Koditschek, 1992) and random walks (Barraquand and Latombe, 1991) have been proposed to overcome that limitation. For an in-depth discussion of navigation functions refer to Dimarogonas and Kyriakopoulos (2007).



(a) PRM roadmap with a path (in red) connecting two points.

(b) A larger PRM roadmap with a path (in red) connecting a starting and a final point. Intermediate dots represent way points.

Figure 6.4: Roadmap constructed by the PRM algorithm and a path found by a graph search method connecting to given points of the world. *Image Credit: MathWorks*.

Combinatorial These methods are said to be exact as they construct a mapping of $\mathscr{C}_{free}$. They were popular in the 1980s, being the visibility graph the first known (Lozano-Pérez and Wesley, 1979). Since then, many appeared, having all in common the building of a roadmap accurately representing the connectivity of $\mathscr{C}_{free}$ (Choset et al., 2005; LaValle, 2006). Inspired by a country highway system, roadmaps model the topology of a complex environment in a simple graph that contains a reduced set of the initial data. They are built once and posteriorly used many times to solve multiple queries to connect different starting and goal configurations. Paths are commonly found, if they exist, with the help of any known graph search algorithm like Dijkstra (Dijkstra, 1959).

Probabilistic or Sampling-based Due to the high computational costs of thoroughly characterizing $\mathscr{C}_{free}$, this family of algorithms, which avoid the exact representation of that space, has received considerable attention lately. They only need to know if a particular robot configuration is safe. Therefore, they incrementally sample the *C-space* with the help of a collision-detection routine to construct a data structure storing 1D *C-space* curves representing collision-free paths. Then, this connectivity map is systematically explored to find a solution. An example can be seen in fig. 6.4. These algorithms are said to be *probabilistic complete* because if a path exists, the probability that the planner solves the problem approaches to one as the running time goes to infinity. In other words, by increasing the number of samples, the probability of finding a collision-free path, if one exists, asymptotically approaches to one (Hsu et al., 2006). Planners such as the Probabilistic Roadmap Method (PRM) and the Rapidly exploring Random Tree (RRT)

are becoming very popular since they are computationally efficient in finding solutions in systems presenting many degrees-of-freedom (Kavraki et al., 1996; LaValle et al., 2000). Specifically, they are more efficient than classical occupancy grids (LaValle et al., 2004) and some variants have been successfully ported to multi-robot systems (Sánchez and Latombe, 2002b). The graph-based PRM precomputes the connectivity of an invariant environment. Thus, it is more suitable when paths have to be found for multiple different queries. However, the tree-based RRT determines trajectories directly without the need for any pre-computation, so it is best suited to work online (Bruce and Veloso, 2002). Many variants can be found, including one based on lazy evaluation (Sánchez and Latombe, 2001); see LaValle (2006) and Geraerts and Overmars (2004) for a comparative study and additional details. Finally, note that this kind of algorithms present some issues (Lindemann and LaValle, 2005) and, in the last decade, some aspects have been extensively researched (Hsu et al., 2006; Nieuwenhuisen and Overmars, 2004; Yershova and LaValle, 2007).

6.2.4 The Canonical Problem for Multiple Robots

Using the notation found in LaValle (2006), the composite state space $\mathcal{X}$ for n robots denoted by $\mathcal{A}^1, \mathcal{A}^2, \ldots, \mathcal{A}^n$ can be generated by the Cartesian product of the configuration-spaces $\mathcal{C}^i$ of the individual robots:

$$\mathcal{X} = \mathcal{C}^1 \times \mathcal{C}^2 \times \ldots \times \mathcal{C}^n \tag{6.8}$$

where the dimensionality of $\mathcal{X}$ is equal to the sum of the dimensions of each individual $\mathcal{C}^i$. Thus, if q^i denotes a particular configuration of a robot $\mathcal{A}^i$, any state x of $\mathcal{X}$ is properly specified by its individual components:

$$x \in \mathcal{X} = (q^1, q^2, \ldots, q^n) \tag{6.9}$$

Since the system is formed by n robots, there are n different composite state space $\mathcal{X}_{obs}^i$ (where $i \in [1, n]$) representing the collision of any part of a robot $\mathcal{A}^i$ with the static obstacle area $\mathcal{O}$:

$$\mathcal{X}_{obs}^i = \{x \in \mathcal{X} \mid \mathcal{A}^i(q^i) \cap \mathcal{O} \neq \emptyset\} \tag{6.10}$$

In environments where multiple robots are working close to each other, the motions of all of them must be carefully planned. First, to avoid collision between robots and obstacles. Second, to skip collisions between each pair of robots. Therefore, $\mathcal{X}_{obs}^{ij} \subseteq \mathcal{X}$, which models collision between a pair of robots $\mathcal{A}^i$ and $\mathcal{A}^j$, is:

$$\mathcal{X}_{obs}^{ij} = \{x \in \mathcal{X} \mid \mathcal{A}^i(q^i) \cap \mathcal{A}^j(q^j) \neq \emptyset\} \tag{6.11}$$

Finally, the total obstacle region $\mathcal{X}_{obs}$ of the multi-robot system can be obtained by combining eq. (6.10) and eq. (6.11):

$$\mathcal{X}_{obs} = \Big(\bigcup_{i=1}^{n} \mathcal{X}_{obs}^i\Big) \bigcup \Big(\bigcup_{ij, i \neq j} \mathcal{X}_{obs}^{ij}\Big) \tag{6.12}$$

where the first term expresses the obstacle region consisting in collisions of each robot with the static obstacles and the second one considers the collision between each pair of robots of the system. Then, the region of the composite space belonging to $\mathscr{X}_{free}$:

$$\mathscr{X}_{free} = \mathscr{X} \setminus \mathscr{X}_{obs} \tag{6.13}$$

The traditional approaches followed in literature to solve this problem are classified into two different categories: *coupled* and *decoupled*. The use of one approach over the other depends on the characteristics of each scenario, and it is generally determined by the trade-off between computational cost and the degree of completeness required.

6.2.5 Coupled approach to MRMP

A coupled planner models the multi-robot system as a single robot system, constructing the combined configuration-space $\mathscr{X}$ in eq. (6.8) and then determining a path τ in it using a single-robot motion planning technique. Let $q_s^1, q_s^2, \ldots, q_s^n$ denote the starting configurations of n robots and $q_f^1, q_f^2, \ldots, q_f^n$ the final configurations. Additionally, let $x_s = (q_s^1, q_s^2, \ldots, q_s^n)$ be the starting composite configuration state and $x_f = (q_f^1, q_f^2, \ldots, q_f^n)$ the final composite configuration state. Then, the problem of finding a path τ between x_s and x_f in $\mathscr{X}_{free}$ for all the individual robots can be formulated as if it was a path finding problem for a single robot. Using eq. (6.4), the path τ can be expressed as:

$$\tau : r \in [0,1] \rightarrow \tau(r) \in \mathscr{X}_{free} \tag{6.14}$$

where $\tau(0) = x_s$ and $\tau(1) = x_f$.

As every point r_i of the path $\tau(r)$ defined in eq. (6.14) describes the individual configurations $q_i^1, q_i^2, \ldots, q_i^n$ for each of the n robots, this approach simultaneously specifies the paths of the particular robots and their trajectories, which determine how paths must be coordinated.

This approach presents two major advantages. First, it solves the multi-robot motion planning problem by using any general single motion planning algorithm applied to the composite configuration-space $\mathscr{X}$. If the search space has a finite and discrete representation, A^* (Hart et al., 1968) or any other graph specific algorithm can be applied to determine the paths for all robots simultaneously. Second, if the original algorithm is able to find a path in a single robot environment, then completeness will be retained in a multi-robot system as well. Moreover, if A^* is employed, optimality is also guaranteed. However, in general, the high dimensionality of the state space $\mathscr{X}$, which increases linearly with respect to the number of robots, usually makes this approach impractical. In such cases where the optimality criterion can be relaxed, suboptimal anytime variants of A^* might be of great help to reduce execution time (Likhachev et al., 2005, 2003). On the other hand, Iterative Deepening A^* (IDA*) decreases the memory consumption of the original A^* by using depth-first search (Korf, 1985). Unfortunately, it can only be effectively employed when robots are packed densely enough that only a few of them can move at any time. Due to their suitability for high-dimensional spaces, some authors have proposed probabilistic

planners (see section 6.2.3) for exploring the joint configuration space (Carpin and Pagello, 2002; Ferguson et al., 2006; Sánchez and Latombe, 2002a,b). However, the structure and size of the joint configuration space limits such approaches to relatively small numbers of robots (Ferguson et al., 2006; Čáp et al., 2013).

This complexity has inspired several approaches that attempt to prune the search space while maintaining completeness. They fundamentally decompose the shared environment into a number of regions such that robots in different regions do not interact with one another. Then, simple rules are established to govern how robots can safely move from distinct regions. One of the first works to exploit this idea consists in creating a *super-graph* for the composite robot by properly combining simple roadmaps independently constructed for each robot in the system (Švestka and Overmars, 1998). A single vertex of this multi-level graph represents an entire region of the workspace where the robots can be safely moved. Motions then are coordinated as soon as, in order to achieve its goal, one robot needs to transition from one super-graph vertex to another. The whole process begins by creating a flat-graph considering all vertexes of the individual roadmaps. The vertexes of this new graph are formed by one node of each of the individual roadmaps so that when the robots are configured at the corresponding configurations do not collide against each other. Then, two nodes are connected if just one robot is moving at a time, and the path it follows is not *blocked* by a stationary robot. This represents the first level of the graph. The subdivision process continues by similarly grouping the level k vertexes into smaller connected subgraphs of level $k+1$. Iterations end when the subgraphs obtained are formed by only one node and no edges. Although intuitively, after testing the proposed algorithms with mobile robots, the authors concluded that up to 5 robots, their approach is practical. However, if more robots need to be correctly coordinated, they state that decoupled planning should be used.

More recent methods go one step further by removing the complexity of reasoning about the geometry. They model the environment as a graph, and robots are considered to be placed in its vertexes, being at most one robot in each vertex. Two robots in different vertexes are also considered to be collision-free. Moreover, these works assume that robots can move between adjacent positions instantaneously. That is, they move from one vertex to a neighbor in a single time step. This particular graph is called *collision-free unit-distance graph* (CUG), where every edge is of unit length and robots travel at unit speed through the edges (Yu and LaValle, 2013). More specifically, the work of Auletta et al. (1999), the Tree-Based Agent Swapping Strategy algorithm (Khorshid et al., 2011), Push and Swap (Krontiris et al., 2013; Luna and Bekris, 2011b), and Push and Rotate (de Wilde et al., 2013) basically try to find a solution by exchanging the positions of two robots in the graph, which can be a natural decision in mobile robots, but not in manipulators. Figure 6.5 shows a complete example of Push and Swap. Similar approaches shift robots into locations where they will not interfere with the motion of the others, enabling then planning for a small number of robots at a time (Peasgood et al., 2008; Wang et al., 2011). In

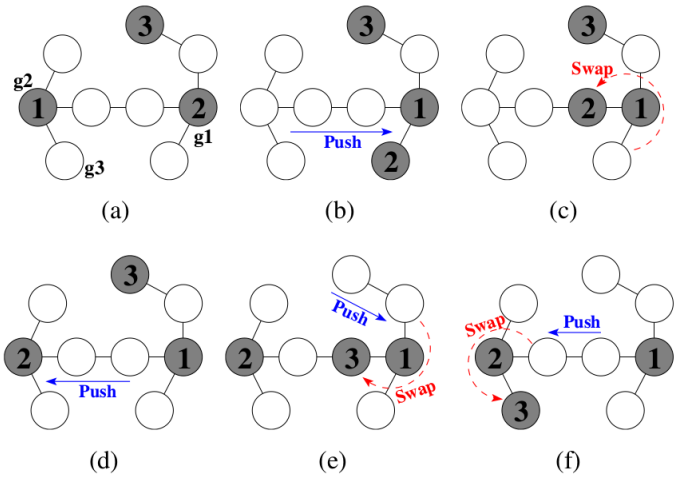


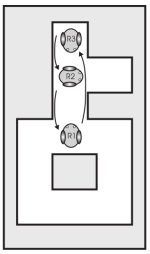
Figure 6.5: A complete example of Push and Swap. (a) Start and goal configurations for three different robots. (b) Robot 1 pushes toward its goal, moving away robot 2. (c) As robot 2 is blocked by 1, then they swap positions. (d) Robot 2 pushes toward its goal. (e) Robot 3 pushes but it is blocked by 1. Then 3 exchanges position with 1. (f) Robot 3 swaps position with 2, and reaches its goal *Image Credit:* *Luna and Bekris (2011a)*.

fig. 6.6, a planning example solved by the multiphase method relying on spanning-trees proposed by Peasgood et al. is given.

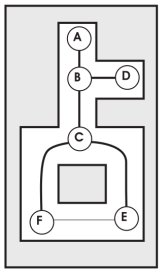
All previous works relying on CUGs are guaranteed to find a solution in polynomial time, but the quality of the path is typically low. In particular, all the above methods only allow a single robot to move at any given time, returning this way sequential paths yielding very long execution times. Parallel Push and Swap is a variant of Push and Swap allowing simultaneous motion of multiple robots, significantly improving paths (Sajid et al., 2012). Unluckily, all these approaches present a few practical limitations for no free-flying robots, for which a common topological roadmap in the control space cannot be easily computed. Furthermore, if the control space is significantly different from the environment, a composite state-space might be required to properly guarantee that two robots in different nodes do not collide.

6.2.6 Decoupled approach to MRMP

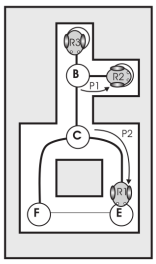
The approaches belonging to this category tend to decrease the dimensionality of the problem state space by planning the motion of each robot independently and then coordinating all of them to avoid collisions. The popularity of these planners has recently increased, mainly because they are faster than the centralized ones. However, they are generally incomplete as in some cases



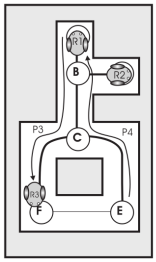
(a) The planning problem to solve.



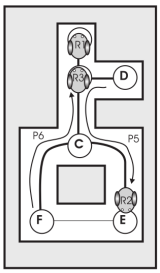
(b) A topological graph-based map of the environment. Nodes represent valid positions.



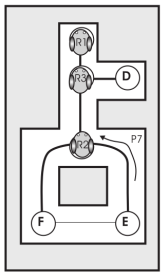
(c) **Phase 1**: all robots are moved to unoccupied leaf nodes of the spanning tree.



(d) **Phase 2**: all robots with goal positions located on leaf nodes fill their goals.



(e) **Phase 3**: all robots with goals on interior nodes are moved into positions where they can reach those goals without obstructing another robot.



(f) **Phase 4**: fill the remaining goals on interior nodes.

Figure 6.6: Multiphase planning in tunnels with spanning-trees. *Image Credit:* *Peasgood et al. (2008)*.

they may not find a solution even though it exists. Definitely, these planers trade off efficiency and completeness for computational simplicity.

Although there are multiple ways of decoupling the motion of multiple robots, they can be classified into two broad families: *prioritized planning* and *fixed-path coordination*. The former assigns different priorities to each of the robots, and then motions are orderly planned one at a time according to the assigned priority. Whereas, the latter, first, independently determines the paths for each robot and finally coordinates the motion of the robots along those paths by adjusting the robots velocities.

6.2.6.1 Prioritized Planning

In this type of approach, a different priority is assigned to any of the different robots, and their motions are successively determined in order of priority using a single robot motion algorithm. By planning one robot at a time, this approach transforms the initial high-dimension combined state space into a limited sequence of low-dimension spaces. Prioritized planning reduces the combined problem to a series of motion planning problems for single robots moving in a previously known dynamic environment.

Formally speaking, in a multi-robot system consisting in n robots, each of which having a given priority, the path of the robot $\mathcal{A}^i$ is calculated considering the higher-priority robots $\mathcal{A}^1, \mathcal{A}^2, \ldots, \mathcal{A}^{i-1}$ as if they were dynamic obstacles moving along their previously determined paths. Therefore, as the positions of the robots $\mathcal{A}^1, \mathcal{A}^2, \ldots, \mathcal{A}^{i-1}$ from the point of view of $\mathcal{A}^i$ are not constant, its configuration-space obstacle region $\mathcal{C}^i_{obs}(t)$ is a time-varying set. This set $\mathcal{C}^i_{obs}(t)$ models the constraints of $\mathcal{A}^i$ at any particular time:

$$\mathcal{C}^i_{obs}(t) = \{q \in \mathcal{C}^i \mid \mathcal{A}^i(q) \cap \mathcal{O}^i(t) \neq \emptyset\} \tag{6.15}$$

where $\mathcal{O}^i(t)$ are the locations in the world of the previously planned robots $\mathcal{A}^1, \mathcal{A}^2, \ldots, \mathcal{A}^{i-1}$ at a given point in time. And, $\mathcal{C}^i_{free}(t)$, according to eq. (6.3), is the complement of the set $\mathcal{C}^i_{obs}(t)$.

The seminal work about prioritized planning was conducted in the mid-1980s (Erdmann and Lozano-Pérez, 1986). There, the authors address the time-varying nature of $\mathcal{C}^i_{obs}(t)$ extending the basic configuration-space to account time. The resulting diagram is called the *configuration space-time*, which is a discrete sequence of ordinary *configuration-space slices*. Each of those slices defines a concrete $\mathcal{C}^i_{obs}(t)$, as expressed in eq. (6.15). Then, the problem of planning a path for the robot $\mathcal{A}^i$ is solved by searching a discrete path in its associated $\mathcal{C}^i_{free}(t)$ that connects the starting and final configurations. The beauty of determining a path in this diagram is that it finds a trajectory, the path for a robot and also a time schedule. However, special attention must be paid to two essential points. First, the paths found by the search algorithm should be monotonically increasing in the time dimension to prevent unrealistic situations (a robot cannot go backward in time). Finally, the real velocities of the robots are tightly bounded to some maximum values; therefore, this imposes additional constraints on the slopes of the valid paths.

The configuration space-time diagram has been extensively used in posterior prioritized planners. This abstraction can be exploited in conjunction with any traditional single robot motion planning method. By way of illustration, in Warren (1990) each robot is assigned a fixed priority, and then a technique based on potential fields is used to sequentially determine in decreasing priority order the paths of each robot in the configuration space-time. First, the authors, starting with the highest-priority robot, build a straight-line path formed by n segments in the configuration space-time connecting the initial and final configurations of the current robot. Finally, the location of the end-points of each of the n segments are adjusted to avoid the dynamic obstacles according to an optimization step based on the potential field applied to the robot's

configuration space-time diagram. In Ferrari et al. (1998), a fixed priority scheme is also used and then stochastic detours are chosen for lower-priority mobile robots.

Since the order in which the paths are planned has a profound impact on the solution found, scientists have studied the performance of a few priority profiles. Buckley (1989) introduced a priority graph to determine a profile that maximizes the number of mobile robots[9] moving along straight lines. This approach consists of two different stages. First, linear paths in the Euclidean space are determined for every robot. Then, priorities are properly assigned to avoid the two following types of conflicts: (i) *starting* conflicts[10] and (ii) *goal* conflicts[11]. However, there is no need to prioritize those paths presenting *en route* conflicts as they can be avoided by properly tuning the velocity of the lower-priority robot. The adjustment is made according to the *area of collision*, determined by geometrically intersecting the area swept by both robots and the time schedule of the higher-priority robot. Unhappily, in some situations, any of the previous enumerated methods cannot correctly resolve some particular conflicts. For instance, when a robot must move out of the way of another one to avoid a collision. In these cases, the authors compute paths containing intermediate points. Even though it works well in sparse environments, this approach fails in crowded environments.

As the choice of priorities affects the length of a robot's path, Bennewitz et al. (2002) and Bennewitz et al. (2001) employed a hill-climbing based approach to determine the priority scheme delivering the overall shortest paths. In the first iteration, they assign a priority to each robot and then a cost-optimal collision-free path is individually computed in descending order. These paths are determined by an A^* algorithm exploring a discrete grid representing the corresponding *configuration space-time* diagram of each robot[12]. Later, two robots are randomly selected, their priorities exchanged, and paths recalculated according to the new priority scheme. This step is repeated until no further improvement in path lengths is found. Furthermore, to avoid getting stuck in local minima, a typical problem with hill-climbing, several initial runs with different robot priorities are computed. Additionally, the authors also provide an alternative approach to the random interchange of priorities, which speeds up the whole process considerably. They compute independent optimal paths for each robot according to the A^* algorithm and then prioritize the interchange among those robots whose paths present cyclic dependencies[13]. As expected, the constrained approach leads the authors to find safety paths faster. Unfortunately, if the intention is to find the shortest-distance paths, then, the search needs to be expanded again to the unconstrained space.

Alternative priority based approaches are also proposed in Van Den Berg and Overmars

[9] Each robot is only able to perform a translating motion.

[10] The linear path of robot $\mathscr{A}_i$ from q_{s_i} to q_{f_i} intersects with the starting point q_{s_j} of another robot $\mathscr{A}_j$

[11] The linear path of robot $\mathscr{A}_i$ from q_{s_i} to q_{f_i} intersects with the target point q_{f_j} of another robot $\mathscr{A}_j$

[12] The heuristic guiding the search is based on the occupancy cost (Moravec and Elfes, 1985), which corresponds to the likelihood that a particular point of the grid is occupied.

[13] Two paths present a dependency if the goal position of one robot lies in the optimal path of the other one or vice-versa.

(2005a), Clark et al. (2002), Geramifard et al. (2006), Regele and Levi (2006), Desaraju and How (2012), and Čáp et al. (2015). In particular, although the solution in Clark et al. (2002) is aimed to determine online motion plans for mobile robots equipped with sensing sensors, it introduces a dynamic priority system that can be conveniently exploited in offline approaches. The robots independently construct their trajectories according to their own configuration-space roadmaps, and they follow them until one or more robots are detected in their neighborhoods. When two or more robots are close enough, the robots priorities are recomputed. Higher priorities are assigned to those robots whose close environment is more crowded. That way, the robots with more room to maneuver are forced to replan their trajectories taking into account the motions (especially the milestones of the roadmaps) of the highest-priority robots.

Finally, a more robust solution is to dynamically vary the priority of each robot so that every one of them has the highest priority for a short period. This can be achieved by $WHCA^*$ (Silver, 2005), where robots perform windowed planning from start to goal in a grid-based world with the help of a reservation table. Searches with A^* related methods can be costly if inaccurate heuristics are employed. Therefore, $WHCA^*$ begins by initially carrying out a backward A^* search for each agent, and the results are used as heuristic guidance. Then, the cooperative search is limited to a fixed depth specified by the current window, reverting to a single robot computation after the window is exceeded. By interleaving searches, the order of planning rotates. Consequently, every unit has the opportunity to be the one with the highest priority during a time-step, being this way the first one marking the reservation table for this concrete time-step.

6.2.6.2 Fixed-Path Coordination

This decoupled approach consist of two stages. In the first of them, paths for each robot are independently determined considering only the obstacles and ignoring the other robots of the system. Thus, an algorithm suitable for a single robot system (see section 6.2.3) can be used to generate them. Once the paths are known, the motions of the robots along these *fixed-paths* are properly coordinated to avoid collisions. There are many approaches to this problem (Akella and Peng, 2004; Cui et al., 2012; LaValle and Hutchinson, 1998; Leroy et al., 1999; Peng and Akella, 2005; Simeon et al., 2002), some of them based on Mixed Inter Linear Programming (MILP). However, fundamentally, the most intuitive coordination among robots can be achieved by stopping them at a particular instant and moving them forward or backward along their independently determined paths.

The path-velocity decomposition approach was first presented in Kant and Zucker (1986). In this pioneering study, the authors plan the motions of a point-object mobile robot. First, a path π which avoids collisions with static obstacles is determined and expressed as a function of the arc length of the path, denoted by s. Then, the 2D *path-time* space ($s \times t$) is built and the dynamic obstacles represented there. A moving obstacle interfering a path can be adequately represented in the robot's $s \times t$ space by the area swept out by this object in that space. Finally, once the

moving obstacles are geometrically present as polygons in $s \times t$, a collision-free path in this space is found with the help of a visibility graph (Lozano-Pérez and Wesley, 1979) properly modified. This visibility graph must be a directed graph, as any path in $s \times t$ must be time-monotonic (the variable time must always increase), and the slopes of the edges are constrained to the maximum velocity of the robot. Lee and Lee (1987) used a similar approach but this time to coordinate two PUMA manipulators (Billingsley, 2006). The authors introduced what they call a *collision map*, which is a 2D representation of a robot's trajectory.

The problem of coordinating the paths of two manipulators was formally addressed in O'Donnell and Lozano-Pérez (1989). The major contribution of this work was the introduction of the *task completion* diagram, which is a *coordination space* still widely used in the field. The authors assume that a precomputed path is divided into several segments. Then, they build a 2D grid, the task completion diagram, with the paths segments belonging to two distinct manipulators, named A and B. A cell C_{ij} of the grid is considered safe if the volume swept out by robot A while performing its i^{th} path segment does not overlap with the volume generated by robot B while performing its j^{th} path segment. Otherwise, the cell is forbidden. In this study, the first stage of the decomposition was called *path generation* and the second one *trajectory generation*. Nowadays, the latter is commonly known by the name of *velocity tunning* or *velocity scheduling*. This denomination is more appropriate because in the second step of the process and using the coordination-space abstraction, the relative velocities of each robot along their precomputed paths are carefully adjusted to avoid conflicts between robots.

If there are more than two robots in the environment, the coordination of their motions can be done in a *global coordination-space* or in a *pair-wise coordination space*. While in the former, all fixed-paths are coordinated at the same time, coordinating paths using the latter requires multiple iterations since only two different paths are considered in each step of the process. In both approaches, the time dimension is implicit. As we will see, each state of any of these coordination-spaces determines the configurations of all robots at a particular instant. Thus, the robots must appropriately adapt their velocities to reach that state at the same moment. Once this discrete coordination space is built, a velocity profile is obtained by exploring the space using any well-established graph traversal algorithm.

The Global Coordination Space Let $\tau^1, \tau^2, \ldots, \tau^n$ denote the independently computed paths for robots $\mathscr{A}^1, \mathscr{A}^2, \ldots, \mathscr{A}^n$, each of which defined as:

$$\tau^i : r \in [0,1] \rightarrow \tau^i(r) \in \mathscr{C}^i_{free} \tag{6.16}$$

where $\tau^i(0) = q^i_s$, $\tau^i(1) = q^i_f$ and, q^i_s and q^i_f are respectively the starting and final configurations of robot $\mathscr{A}^i$. Then, its *coordination-space* $\mathscr{X}$ is a n-dimensional state space:

$$\mathscr{X} = S^1 \times S^2 \times \ldots \times S^n \tag{6.17}$$

where each $S^i \in [0,1]$ represents the domain of the fixed-path of $\mathscr{A}^i$; see eq. (6.16). Specifically, a given point $s \in S^i$ denotes the configuration $q \in \mathscr{C}^i_{free} = \tau^i(s)$. Thus, the states $x_s \in \mathscr{X} = (0,0,\ldots,0)$ and $x_f \in \mathscr{X} = (1,1,\ldots,1)$ models that all robots are at their starting configuration and at their final ones respectively.

But, in order to determine a collision-free path for all robots using the coordination-space abstraction, the set $\mathscr{X}_{free} \subseteq \mathscr{X}$ must be computed. This set contains all those points of $\mathscr{X}$ where the robots do not collide. For each pair, $\mathscr{A}^i$ and $\mathscr{A}^j$ of robots, the set $\mathscr{X}^{ij}_{obs} \subseteq \mathscr{X}$ is:

$$\mathscr{X}^{ij}_{obs} = \{(s^1, s^2, \ldots, s^n) \in \mathscr{X} \mid \mathscr{A}^i(\tau^i(s^i)) \cap \mathscr{A}^j(\tau^j(s^j)) \neq \varnothing\} \tag{6.18}$$

and, then, $\mathscr{X}_{obs}$ denotes the prohibited points of $\mathscr{X}$ that must be avoided in path coordination:

$$\mathscr{X}_{obs} = \bigcup_{ij,\, i \neq j} {\mathscr{X}_{obs}}^{ij} \tag{6.19}$$

Once defined $\mathscr{X}_{free} = \mathscr{X} \setminus \mathscr{X}_{obs}$, motion planning consist in searching a path ϕ that connects $\phi(0)$ and $\phi(1)$:

$$\phi : s \in [0,1] \rightarrow \phi(s) \in \mathscr{X}_{free} \tag{6.20}$$

where for a given s_i, the image of the function $\phi(s_i) = (s^1_i, s^2_i, \ldots, s^n_i)$ specifies a placement of the n robots at their configurations $\tau^1(s^1_i)$, $\tau^2(s^2_i)$, …, $\tau^n(s^n_i)$. Finally, the state $\phi(0) = (0,0,\ldots,0)$ represents the initial configuration of each robot and $\phi(1) = (1,1,\ldots,1)$ the final configurations.

The Pair-Wise Coordination Space This approach was conceived to reduce the dimensionality of the state space associated with global coordination-space. Given n previously calculated paths for n distinct robots, the approach consist in successively determine $n-1$ paths in $n-1$ two-dimensional coordination-spaces. Let $\mathscr{X}^2, \mathscr{X}^3, \ldots, \mathscr{X}^n$ denote each of the $n-1$ state spaces. Then, the state space $\mathscr{X}^i$ is defined by:

$$\mathscr{X}^i = \begin{cases} S^i \times S^{i-1}, & \text{if } i \text{ is } 2 \\ S^i \times S^{1..(i-1)}, & \text{if } i \in [3,n] \end{cases} \tag{6.21}$$

where S^i denotes the points belonging to the fixed-path i and $S^{1..j}$ the domain of the coordinated path $\tau^{1..j}$ obtained by coordinating all fixed-paths $p \in [1,j]$:

$$\tau^{1..j} : s \in [0,1] \rightarrow \tau^{1..j}(s) \in \mathscr{X}^j_{free} \tag{6.22}$$

where $\mathscr{X}^j_{free} \subseteq \mathscr{X}^j$ contains those states of $\mathscr{X}^j$ which are collision-free. For example, the two coordinate axes of $\mathscr{X}^2$ encode the domains of the paths 1 and 2, formally denoted by S^1 and S^2. Then, the coordinated path $\tau^{1,2}$ defined by eq. (6.22) can be determined by properly searching the state space $\mathscr{X}^2 = S^1 \times S^2$. Once determined $\tau^{1,2}$, it can be similarly coordinated with the following fixed-path. This time one axis of $\mathscr{X}^3$ will encode the previously determined points of the path $\tau^{1,2}$, denoted by $\mathscr{S}^{1,2}$, and the other axis the domain of the fixed-path 3, denoted by $\mathscr{S}^3$. By iteratively applying this process, all fixed-paths will be coordinated as long as such pair-wise coordination exists.

6.3 Roadmap Search Based Motion Planning for MIRADAS

Once finished the review of the related literature, in this section, we discuss the motion planning solution proposed for MIRADAS. First of all, we mathematically formulate the problem to solve and introduce some basic notation employed through this section.

6.3.1 Problem Description

A description of our problem follows. We refer to the m independent parameters that specify the position of an arm by the name of *configuration*. A configuration is denoted by q. The m-dimensional space containing all possible configurations is called configuration-space. We consider a system composed of any given integer number n of articulated arms $\mathscr{A} = \{\mathscr{A}_1, \mathscr{A}_2 \dots \mathscr{A}_n\}$ moving in a common three-dimensional environment represented by $\mathscr{W}$. Each arm $\mathscr{A}_i$ patrols a reduced area of this environment and can move at a maximum velocity v_{max_i}. Each arm $\mathscr{A}_i$ has its own configuration-space represented by $\mathscr{C}_i$. In addition, for each arm $\mathscr{A}_i$ there is an area in $\mathscr{C}_i$ prohibited due to mechanical constraints that we will denote by $\mathscr{C}_i^{ZoA}$. The subset of $\mathscr{C}_i$ containing all the mechanically possible configurations is represented by $\mathscr{C}_i^{feas}$. We use the notation $\mathscr{A}_i(q)$ to refer to the volume in the common environment occupied by the arm $\mathscr{A}_i$ at configuration $q \in \mathscr{C}_i^{feas}$. Each robot has been given a starting and final configuration, q_{s_i} and q_{f_i} respectively, both belonging to $\mathscr{C}_i^{feas}$. For convenience, we reduce for each arm $\mathscr{A}_i$ its available $\mathscr{C}_i^{feas}$ into a smaller set known as a *roadmap*, represented by $\mathscr{R}_i$, that also contains q_{s_i} and q_{f_i}; see section 6.3.4.1 for an in-depth discussion about how a roadmap is constructed.

We define a path from a given q_{s_i} to a q_{f_i} as being a continuous sequence of configurations entirely belonging to $\mathscr{R}_i$ and a trajectory as a path parametrized by time. A trajectory for $\mathscr{A}_i$ is formally expressed as $\pi_i : t \in [0, T_i] \rightarrow \pi_i(t) \in \mathscr{R}_i$ such that $\pi_i(0) = q_{s_i}$ and $\pi_i(T_i) = q_{f_i}$. The variable t denotes time and T_i the *travel time*, the instant when $\mathscr{A}_i$ reaches its destination and remains there. For convenience, these trajectories will be parametrically expressed in terms of a time sample $k \in \mathbb{N} \cup \{0\}$. The time step between two consecutive samples is denoted by Δt. Then, the continuous function $\pi_i(t)$ becomes the discrete function $\pi_i(k) = \{q_{i_0}, q_{i_1} \dots q_{i_n}\}$, where $q_{i_j} \in \mathscr{R}_i$ is the configuration of $\mathscr{A}_i$ when $k = j$. This way, $\pi_i(k)$ is defined by several waypoints for which $\mathscr{A}_i$ has to pass through at a particular time $k * \Delta t$. Two trajectories π_i and π_j are said to be *safe* if the arms following them do not collide and neither of them moves faster than its corresponding maximum velocity.

By *configuration time*, we understand the time required for all arms to move from their initial positions to their goals. This time, frequently known as *makespan*, is defined as the travel time of the last arm reaching its destination. For a given trajectory set $\mathscr{T} = \{\pi_1, \dots, \pi_n\}$, it can be computed as $T_{\mathscr{T}} = \max_{\forall i \in \mathscr{A}} k_i^{tr}$, where k_i^{tr} denotes the discrete time step in which the arm $\mathscr{A}_i$ reaches its destination.

Finally, we assume that initial and final positions for each arm come from two different

feasible assignment plans delivered by the target allocator. We also assume that target allocator has effectively performed its task; refer to chapter 5 for details. Therefore, it is guaranteed that there is no collision between two any initial and two any final positions and that also the minimum distance between targets are met.

Problem 1 (Motion Planning Along Roadmaps) Taking into account all previous definitions and assumptions, the problem we want to solve is the following: given a set of n arms $\mathscr{A}_i$, where n can be any integer number, each controlled by a device showing the behavior discussed in section 3.5.1.2 and each with a roadmap $\mathscr{R}_i$ and a task $\langle q_{s_i}, q_{f_i} \rangle$ to be accomplished, find a set $\mathscr{T} = \{\pi_1, \ldots, \pi_n\}$ fulfilling:

1. No two trajectories π_i, π_j are unsafe.
2. The number of *start-stop* cycles (refer to section 3.5.1.2) in each π_i is not greater than a given $nStop$.
3. The configuration time is equal to or less than a given value T_p, set to 120 s.

Points (2) and (3) are not related to science but to the efficiency of the instrument. While point (3) originally was a strict requirement for the MXS motion planner (Eikenberry et al., 2014), we are now treating this as a goal.

6.3.2 Sensing the Real World

To provide a high degree of flexibility and portability, the motion planning algorithm will only retrieve information about the n arms it can work with through a number of boolean functions. We can distinguish here functions of two kinds:

1. Those checking if a given configuration $q \in \mathscr{C}_i$ is contained in the $\mathscr{C}_i^{ZoA}$ of a given arm $\mathscr{A}_i$: `IsInZoA(q)`.
2. Those reporting if a given arm $\mathscr{A}_i$ configured at $q \in \mathscr{C}_i^{feas}$ collides with the others at time t: `Collision(q,t)`.

The algorithm will need as many functions of each type as arms have to be planned. That is, it will require n functions `IsInZoA(q)`, each providing particular information about a distinct arm, as well as n functions `Collision(q,t)`, one for each arm. Each function `IsInZoA(q)` considers the real ZoA of each arm as computed by a calibration procedure. All functions `Collision(q,t)` work with a safety tolerance. It considers a small area around each arm to prevent issues caused by small timing divergences during trajectory execution as well as differences between arms due to manufacturing tolerances.

For the motion planner, these $n \times 2$ functions are black boxes hiding the particular kinematic characteristics of each arm in the system. Every time the planner requires information from a particular arm, it will call the appropriate function.

6.3.3 Prioritized Approach

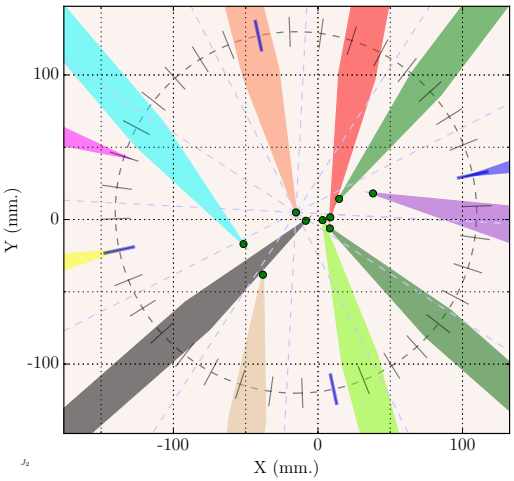
The trajectories determination problem defined in section 6.3.1 belongs to the field of motion planning in a multi-robot system. As discussed in section 6.2, there are two methods to solve this sort of problems: coupled and decoupled. Considering the high dimensionality of the composite state-space as well as the constraints imposed to the algorithm, here we propose a decoupled approach based on priorities. In section 6.3.3.2, high-level pseudocode for the prioritized planning approach is presented. However, before discussing the algorithm, a few concepts and notation need to be introduced.

6.3.3.1 Notation

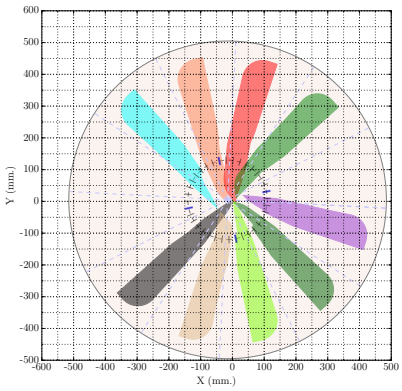
We will work with the concept of *space-time* domain, which is a four-dimension space combining the 3D environment where all arms move ($\mathcal{W}$) with time. It is denoted by $\mathcal{Y}$ and formally defined as $\mathcal{Y} = \mathcal{W} \times t$, where $t \in [0, \infty)$. When an arm follows a given trajectory, then it can be thought of as a dynamic object occupying a particular region in $\mathcal{Y}$. In addition, the *obstacle-time* subset, denoted by $\mathcal{O}^{\Delta}$, will contain the *space-time* regions occupied by all previously planner arms. In fig. 6.7, an example is provided. Finally, we will employ the notation $\mathcal{A}_i^{\Delta}(\pi_i)$ for the transformation determining the *space-time* region swept by the arm $\mathcal{A}_i$ when moving along the trajectory π_i.

6.3.3.2 Algorithms

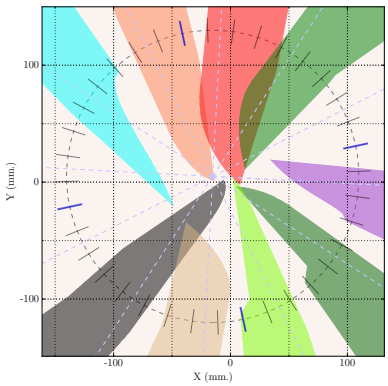
Two elements are required to build a prioritized solution: (i) a strategy to prioritize the arms and (ii) a method to compute the motions of a single arm in a known dynamic environment. The first point is one of the most important choices to take in an approach of this kind since the ordering of the agents will impact the performance of the solution. In practice, any method for assigning priorities can be employed. However, among all the possible combinations and considering the requirements of the problem at hand, it makes sense to utilize one appropriate for the purpose of minimizing the system makespan. Then, once selected the priority scheme, the arms are arranged in priority descending order. As can be seen in algorithm 6, this sequence is employed by the function `Prioritized-Planning`, which iteratively plans trajectories for each of arm according to its priority. The trajectory of each arm is determined by `Arm-Motion-Plan` (line 5), a single arm roadmap-based motion planner described in section 6.3.4. This method tries to find a trajectory in the arm's roadmap $\mathcal{R}_i$ such that avoids the space-time regions occupied by the previously planned arms (*obstacle-time* set $\mathcal{O}^{\Delta}$). If `Arm-Motion-Plan` fails to find a safe trajectory for the current arm, then `Prioritized-Planning` terminates with failure (line 7). Otherwise, it tries to determine a motion plan for the next arm.



(a) A close look of the MXS bench with the arms at their destinations. Nine out of twelve probes have been assigned to targets (dots). The circle represents the MIRADAS focal plane.



(b) The 9 different zones in the MXS bench show the areas swept by the assigned arms when moving from their initial positions to their final ones.



(c) Detail of the swept areas. Although a few of these areas intersect, the arms will not collide if a proper time profile is found.

Figure 6.7: The space-time set $\mathcal{O}^{\Delta}$ containing 9 trajectories projected in the 2D shared environment.

Algorithm 6 Arm-prioritized Planning

Input: $\mathcal{A}$: A set of n arms ordered by priority, $\mathcal{R}$: a set of n roadmaps
Output: $\mathcal{T}$: the set of trajectories for $\mathcal{A}$ if it exists. Otherwise failure ($\varnothing$)

```
1: function PRIORITIZED-PLANNING(A, R)
2:     T ← ∅
3:     O^Δ ← ∅                ▷ No trajectories planned yet, so empty obstacle-time set
4:     for i ← 1 to n do
5:         π_i ← ARM-MOTION-PLAN(A_i, R_i, O^Δ)   ▷ Plan trajectory with current obstacle-time set
6:         if π_i = ∅ then    ▷ Has a trajectory been found for the current arm?
7:             return ∅                                              ▷ Failure
8:         end if
9:         T ← T ∪ {π_i}                          ▷ Trajectory found, so update the set
10:        O^Δ ← O^Δ ∪ A_i^Δ(π_i)   ▷ Update obstacle-time set with the new trajectory π_i
11:    end for
12:    return T                                  ▷ Success: trajectories found for all arms!
13: end function
```

6.3.4 Planning an Arm Trajectory

In this section, we present a roadmap-based motion planner for a single arm moving in a dynamic environment known beforehand. Algorithms relying on sampled-based roadmaps such as PRM (Kavraki et al., 1996), RRT (LaValle et al., 2000), and their multiple variants have been successfully demonstrated on different robotic platforms. They construct roadmaps by sampling C-space and checking their validity with the help of a function sensing the environment. By limiting the motions of the robots to the paths present in the roadmap, the search space to be explored is conveniently reduced. Once the roadmap is built, a standard graph-search technique can be employed to find the best path connecting two points of the roadmap according to giving criteria.

As stated before, sampling algorithms are said to be probabilistically complete as its probability of failure exponentially tends to zero when the number of samples in the roadmap approaches to infinity. Furthermore, the quality of the trajectories computed highly depends on the quality of the roadmaps (Lindemann and LaValle, 2005). If roadmaps accurately model the real connectivity of the mechanism C-space, this kind of planners offers a good compromise between being practically usable in complex scenarios and completeness. Indeed, some authors assert that motion planning in an unconstrained space is not useful (LaValle and Hutchinson, 1998; Van Den Berg and Overmars, 2005a). Evidence shows that the path quality of a robot constrained to a roadmap improves as roadmap more densely covers the free configuration-space. However, after some threshold, adding more states hardly improves the solution found (Van Den Berg and Overmars, 2005a).

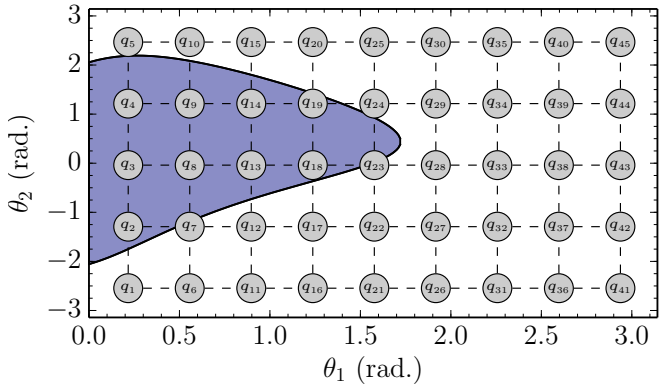
6.3.4.1 The Roadmap

As its name indicates, a roadmap is a smaller set of the original arm feasible C-space containing a series of alternative paths connecting a starting and a goal configuration. Every path comes determined by two elements: its *vertices* and its *edges*. The former of these represents waypoints in the path. They can be the initial, the final or any intermediate point in the path. The edges model the segments of the path that connect every two consecutive waypoints. In principle, these segments can have any shape, which will depend on the interpolation method utilized to connect the two vertices involved. Here, any polynomial function producing smooth transitions or an ad-hoc heuristic search could be effectively applied (Geraerts and Overmars, 2005; Isto, 2002). However, for their simplicity, straight-lines have been frequently used in theoretical works. In our case, due to the problem constraint presented in section 3.5.1.2, all edges will also be straight-lines.

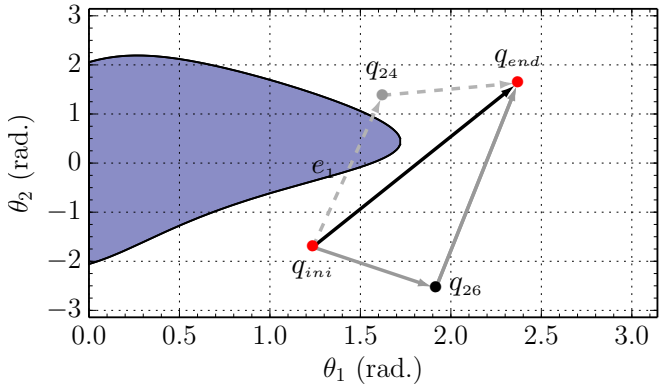
A roadmap is expressed in terms of a graph $\mathscr{R} = (V, E)$ formed by the set V containing the vertexes and the set E containing the edges. By sampling the arm C-space with the appropriate `IsInZoA(q)` function, introduced in section 6.3.2, paths are incrementally constructed following a process involving three steps. The two first of them do not take into account the initial and final configuration of the arm. In the first step, the set of all vertexes in the roadmap is determined. There exist in literature a spectrum of approaches to computing this set, ranging from pseudo-random to grids (LaValle et al., 2004). We use a grid similar to the one in fig. 6.8a and check each configuration belonging to a junction for feasibility. Those configurations in the arm's ZoA are automatically discarded, while the rest are included in the set V.

Then, in the second step, we determine which pairs of vertices of V will be connected, resulting, this way, in a different number of edges. A predefined number of pairs (q_i, q_j) from V are stochastically chosen. If the straight-line connecting a given q_i and a given q_j do not traverse ZoA, the edge $q_i \rightarrow q_j$ is added to the roadmap edge set E. When selecting these pairs, we impose the constraint that a given vertex $q_i \in V$ cannot be present in more than one resulting edge.

Finally, in the third step, the initial q_{ini} and final q_{end} configurations of the arm are considered to form several paths between them involving a different number of edges. First, q_{ini} and q_{end} are properly connected to every edge in the previously computed set E, delivering paths with three edges $(q_{ini} \rightarrow q_i \rightarrow q_j \rightarrow q_{end})$. If the corresponding edges $q_{ini} \rightarrow q_i$ and $q_j \rightarrow q_{end}$ do not traverse ZoA, then the path is included in the roadmap. Furthermore, the q_{ini} and q_{end} points are connected to every vertex q_i of the set V, resulting multiple paths with two edges $(q_{ini} \rightarrow q_i \rightarrow q_{end})$. If the edges of these paths are mechanically feasible, the paths are added to the roadmap. Finally, a direct path linking together q_{ini} and q_{end} is also included if it is feasible. In fig. 6.8b, three different paths are represented in C-space.



(a) The arm C-space is sampled using a grid. Here, as an example, we show a particular 9×5 grid. All those configurations q_i not in the ZoA will be included as vertexes in the arm roadmap. Additionally, a random number of pair of vertexes (q_i, q_j) are connected and included in the roadmap's edge set.



(b) Three paths connecting the initial (q_{ini}) and final (q_{end}) position of an arm in *C-space*. The feasibility of each edge is checked. If any passes through ZoA, then the path containing it is automatically discarded. In this case, the path $q_{ini} \rightarrow q_{24} \rightarrow q_{end}$ is discarded due to the infeasibility of e_1 ($q_{ini} \rightarrow q_{24}$).

Figure 6.8: Sampling C-space and roadmap paths.

6.3.4.2 The Planner

Our planner uses a two-level approach to compute a collision-free trajectory. In the first of them, the most promising unexplored path connecting the origin and destination of the arm is selected from the arm's sampled-based roadmap. Then, on the local level, the motions of the arm along that path are appropriately coordinated with those of the previously planned arms. As we have seen, a path might be composed of several edges, each of which will be adequately discretized

and systematically explored. Finally, our solution incrementally generates the search space as it directly depends on the previous motions performed by the arm.

The pseudocode of the proposed motion planner is given in algorithm 7. It iterates over the paths in the arm's roadmap (line 2), from the most promising in terms of estimated travel time to the least one. If two or more paths present the same estimation, one of them is arbitrarily selected. Then, the edges in a path are sequentially explored (line 6) by the method `Find-Edge-Traj`, discussed in the following pages. If it founds a safe trajectory for the current edge in the path, then, the next one is visited. Otherwise (line 8), the remaining edges are skipped, and the next path in the roadmap is checked. The search ends when collision-free trajectories have been found for every edge in a path or the entire roadmap has been unsuccessfully visited.

6.3.4.3 Estimating Path Travel Time

The travel time of a given roadmap path is computed as the sum of the estimated travel times of each of its parts. A lower-bound estimate for the travel time of a roadmap edge e_{ij} connecting arm configuration $q_i = (\theta_1^i, \theta_2^i)$ and $q_j = (\theta_1^j, \theta_2^j)$ can be determined as follows:

$$t_{e_{ij}} = \max_{\forall k \in [1,2]} \frac{|\theta_k^i - \theta_k^j|}{v_{max}^k} \tag{6.23}$$

Algorithm 7 Finding a trajectory for one arm

Input: $\mathcal{A}_i$: a given probe arm , $\mathcal{R}_i$: arm's roadmap , $\mathcal{O}^{\Delta}$: *obstacle-time* set
Output: π_i: a trajectory for $\mathcal{A}_i$ if it exists. Otherwise, failure ($\varnothing$)

```
1:  function ARM-MOTION-PLAN(A_i, R_i, O^Δ)
2:      for all p ∈ R_i do   ▷ Explore paths in Roadmap: most promising first
3:          t ← 0                                                ▷ Travel time
4:          π_i ← ∅
5:          for all e ∈ p do                          ▷ Explore all edges in a path
6:              ⟨π_e, t_e⟩ ← FIND-EDGE-TRAJ(e, A_i, O^Δ, t)
7:              if π_e = ∅ then          ▷ Check if traj. found for the current edge
8:                  Exit loop                    ▷ Skip remaining edges in cur. path
9:              end if
10:             π_i ← π_i ∪ {π_e}      ▷ Update traj. with local traj. along the edge
11:             t ← t + t_e ▷ Update travel time with the time spent in the edge
12:         end for
13:         if trajectories found for all edges in a path then
14:             return π_i                                    ▷ Trajectory found!
15:         end if
16:     end for
17:     return ∅                    ▷ Failure: no trajectory found for this arm
18: end function
```

where θ_k is the k component (or *degree-of-freedom*) of an arm configuration and v^k_{max} is its maximum velocity.

6.3.4.4 Exploring an Edge of the Roadmap

In this section, we discuss how a trajectory along a given edge of the roadmap is explored to determine if it can be safely followed by the arm. The method we employ here is partially inspired by a grid-based planner for robotics application (Van Den Berg and Overmars, 2005b). We fundamentally represent the state-time search space in a sort of grid that is dynamically grown. Here, the given edge is discretized into a mesh of states and connections representing valid transitions between these states. This convenient representation enables addressing motion planning as a graph search problem. Our approach differs from that work in that we do not invariably use a prefixed 3-connected grid. The neighbors of each state are dynamically computed considering the number of *start-stop* cycles accumulated during the previous motions. Furthermore, our approach is also able to limit the maximum number of *start-stop* cycles per trajectory. Finally, we also use different granularities for the motion action time step and collision checking.

The State-Time Space We use the notation $G = (X, C)$ to denote the graph G dynamically built to coordinate the motions of an arm along a given roadmap edge. The set X represents the nodes of the graph, while C is the set of connections between each pair of nodes. Each node $z = (x, t)$ is formed by a compound element x defining the state of the arm and a scalar t specifying a notion of time. The domain containing all feasible pairs (x, t) is known as *state-time space* (Erdmann and Lozano-Pérez, 1987; Fraichard, 1998). Since its inception, this representation has been broadly used for motion planning in dynamic environments.

In our case, all the possible configurations an arm can adopt when moving along a roadmap edge have been conveniently reduced to a variable s, which ranges from 0 to 1. This variable specifies the distance traveled along the edge, where $s = 0$ and $s = 1$, respectively, represent the configurations of the source and destination vertexes of that edge. Then, the state of an arm is defined as a 3-tuple (s, ns, m), where ns denotes the number of *start-stop* cycles the arm has performed and, finally, m the motion action used to reach this state. This way the arm being planned is represented by a point in 4-dimensional *state-time* space and the previously planned arms are treated in each particular instant of time as static objects.

Discretization To perform a discrete-time analysis, we use the approach proposed in LaValle and Hutchinson (1998). We select a global time step Δt. As a result, an edge e_{ij} of length d can be approximated by a finite sequence containing n points. Each of these points corresponds to the arm's configuration at instant $k\Delta t$, where $k \in \{0, 1 \dots n-1\}$. We select then $v_{e_{ij}}$ for e_{ij} as the greatest value smaller than v_{max} so that the edge is divided into an integer number of segments. Consequently, the normalized length of each of those segments is $\Delta s = |v_{e_{ij}}|\Delta t/d$. In addition, for

the purpose of significantly increasing the search space, we have included a parameter $\alpha \in (0,1)$ modulating the value v_{max} of each robot as follows:

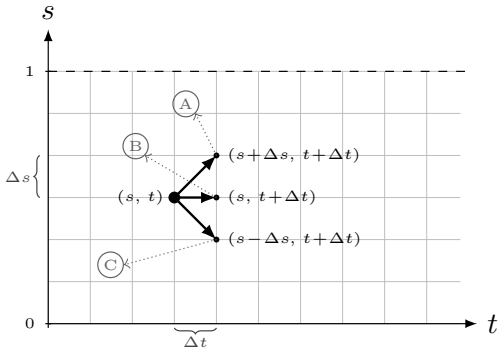
$$v'_{max} = \alpha\, v_{max} \tag{6.24}$$

Indeed, the value of v'_{max} is the one used to appropriately determine the previously commented $v_{e_{ij}}$. This way, we enable the consideration of extra velocity profiles and consequently, extra trajectories for each arm in the system.

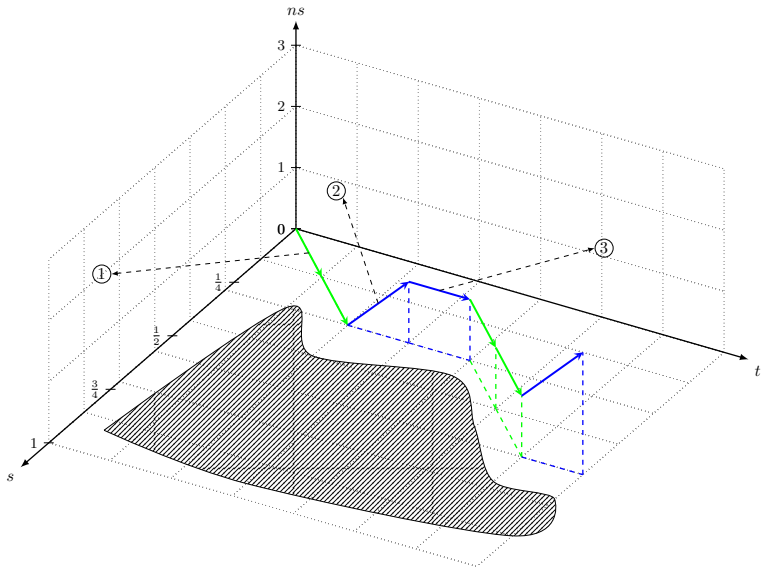
Finally, it is worth mentioning that the $v_{e_{ij}}$ obtained, which is in configuration space units, might be a fraction. This, in terms of control, tends to be unrealistic since stepper motors generally do not allow the execution of fractions of step. Therefore, a normalization stage needs to be carried out to adapt this discretization to the real world. This phase is frequently omitted in academic works unless a kinodynamic approach was followed. Nevertheless, in our case, it can be implemented straightforwardly. Since there is no real need to exactly pass through the intermediate points of a given path defined in any roadmap, the slope of the roadmap edges can be conveniently modified to consider real step velocities for each motor axis.

Motion Actions - Graph Connections It is assumed that the motor controller of an arm can process a command each Δt seconds. The arm, therefore, is only allowed to change velocity at each instant $k\Delta t$ and the values to choose from are constrained to $-v_{e_{ij}}$, 0, and $v_{e_{ij}}$. That is, the arm can decide to move Δs backward along the edge, to remain motionless or to move Δs forward. These motion actions determine the neighbors a graph node can reach. In practice, connections between states must always move forward in time. Thus, if considering only the s component of the arm state, the three potential neighbors of a given state-time node (s,t) are: $(s-\Delta s, t+\Delta t)$, $(s, t+\Delta t)$ and $(s+\Delta s, t+\Delta t)$. As shown in fig. 6.9a, this defines a grid of potential graph connections to consider when trying to find a safe trajectory along the edge. However, not all these connections might be valid.

When the maximum allowed number of start-stop cycles has been reached, the arm is only permitted to move in the current direction. That is the main reason why the nodes of the graph store also information about the motion action m. This variable can only take three possible values $\{-1,0,1\}$, respectively representing backward motion, no motion or forward motion. Although it will be discussed in depth in section 6.3.4.5, we would like to introduce here that, in fact, the planning of motions along a roadmap edge can be seen as a search in a more general 3D space. The additional axis of such search-space, as illustrated in fig. 6.9b, is one specifying the number of start-stop cycles (ns) accumulated. This state variable ns is of considerable significance since it impacts on the geometry of the final path found by the algorithm. Its maximum allowed value can adequately control the smoothness of the solution, which, as stated before, is a fundamental requirement of our design.



(a) The potential neighbors reachable from a given state-time node (s,t), where $s \in [0,1]$ denotes the roadmap edge distance and t a scalar time step (Δt). The neighbors A, B and C respectively represent a forward motion along the roadmap edge, no motion, and a backward motion. Each of those atomic actions has a duration of one time step.



(b) The search space with a third dimension, the *start-stop* cycles counter (ns). If the arm is in motion, this counter is incremented every time a motion action different from the current one happens. The arrows show the different atomic motion actions to be executed by the arm each Δt seconds, while the dashed area defines an *obstacle-time* object to be avoided. The label 1 shows a forward motion. The label 2 represents a stop after a forward motion, therefore, the ns counter is incremented. The label 3 specifies a motionless (or stop) action. The sequence of motion actions shown in the figure are: *forward* (2 time steps), *stop* (2 time steps), *forward* (2 time steps) and *stop* (1 time step).

Figure 6.9: The state-time search space of one roadmap edge.

6.3.4.5 Searching a Trajectory in a Roadmap Edge

As it has been stated in section 6.3.4.4, the problem of finding a safe trajectory along a roadmap edge can be formulated as a graph search problem. The state-time grid has to be systematically explored to find a collision-free path connecting the (s,t) points $(0,t_s)$ and $(1,t_f)$, where 0 and 1 respectively denote the source and destination vertexes of the edge and t_s and t_f ($t_s < t_f$) the initial and arrival time intervals. Since we are interested in obtaining a system makespan lower than a given value, we will try to achieve it by individually minimizing the travel time of each arm. In practice, the A^* algorithm is the de facto choice when an optimal path connecting two nodes of a graph has to be computed. However, in dynamic environments, a *depth-first* graph search (DFS) is guaranteed to be faster than A^* in computing trajectories reaching its destination as soon as possible (Van Den Berg and Overmars, 2005b). Unfortunately, arriving too early might be inconvenient, especially if the scenario presents *goal conflicts*. Conflicts of this kind occur when the path of an arm passes by the goal location of another one. No matter the prioritization scheme, if a lower priority arm arrives at its goal earlier than others having higher priorities, then further actions need to be taken to ensure safety. To illustrate that, let us consider an arm

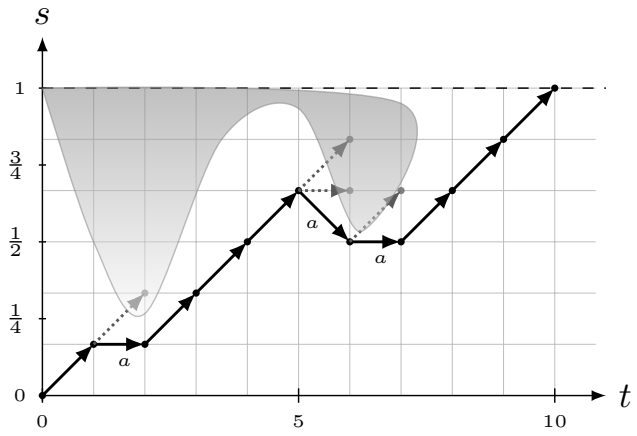


Figure 6.10: A depth-first search of a trajectory along a given roadmap edge considering only (s,t). The gray area represents a dynamic object expressed in *state-time*. The search starts at one edge end ($s = 0$) and completes when the other one ($s = 1$) is reached. The bold arrows show the resulting trajectory, while the dotted ones show those motion actions unsuccessfully explored. The neighbours are always visited in the following order: *forward*, *no motion* and *backward*. Therefore, the DFS tries to get the arm to its final destination in a minimum number of Δt. If there is no constraint in ns, the trajectory found will closely surround the gray area, yielding, in general, a sequence of unsmooth motions. The actions labelled as a are those producing a *start-stop* cycle, consequently incrementing the ns counter.

$\mathcal{A}_i$ with priority p_i and another arm $\mathcal{A}_j$ with priority p_j such that $p_i > p_j$. The collision-free trajectory for $\mathcal{A}_i$ and $\mathcal{A}_j$ are respectively denoted by π_i and π_j, being T_i and T_j the corresponding travel times. Then, if $T_i > T_j$, π_i and π_j are, by construction of algorithm 6, safe during the time interval $[0, T_j]$, but, they might not be during $(T_j, T_i]$. Thus, the feasibility of π_j in the latter interval must be explicitly checked.

Algorithms The method `Find-Edge-Traj` computes the motion actions safely moving the arm along a roadmap edge. The algorithm manages status information and also back-pointers to preceding nodes for each state-time node z generated during the search. The former, labeled as *status* $\in$ {*UNEXPLORED, SAFE, UNSAFE*}, prevents revisiting unpromising nodes, whereas the later, labeled as *parent*, is required to retrieve the sequence of motion actions forming a successful trajectory. Its pseudocode is given in algorithm 8.

`Find-Edge-Traj` first stores (line 2) the source vertex of the edge (z_s) in the sequence of state-time nodes to be explored ($\mathcal{N}$). Then, the algorithm loops over this sequence (line 3). The state-time nodes are consumed (line 4) and visited by the helper method `Explore-Node` (line 5) in LIFO order, which guarantees a *depth-first* search. The search finishes when one collision-free node being at the destination roadmap vertex ($s_i = 1$) is found (line 6), or all state-time nodes have been unsuccessfully explored.

Algorithm 8 Searching a safe trajectory along a roadmap edge

Input:
$z_s = (s_s, ns_s, m_s, t_s)$: a *state-time* node
$nStop$: max. *start-stop* cycles allowed
T_f: max. *time* to explore

Output: true if an edge trajectory exists. It can be recovered using the data stored in the *parent* field of each node z.

```
1:  function FIND-EDGE-TRAJ(z_s, nStop, T_f)
2:      N ← ⟨z_s⟩                          ▷ Sequence of state-time nodes to explore
3:      while N ≠ ∅ do                                  ▷ Are nodes pending?
4:          z_i ← N[−1]                          ▷ Get the last elem. in sequence N
5:          N_i ←EXPLORE-NODE(z_i, nStop, T_f)
6:          if s_i = 1 and z_i.status = SAFE then   ▷ If destination reached and no
    collision there
7:              return true      ▷ Trajectory found. Use backpointers to recover traj.
8:          end if
9:          N ← N − ⟨z_i⟩                  ▷ Remove the current node from the sequence
10:         N ← N + N_i      ▷ Add the neighbours at the end of the remaining nodes
11:     end while
12:     return false
13: end function
```

Algorithm 9 shows the pseudocode for method `Explore-Node`, which employs the appropriate collision checking routine `Collision` discussed in section 6.3.2. Each time it first explores a node

z ($z.status$ = *UNEXPLORED*), this node is checked for safety (line 5) against the environment (the *obstacle-time* set). If it is collision-free, then its neighbors are computed (method `Neighbours` in line 8) only if $t < T_f$. This upper bound of t is introduced to abort the search if a trajectory has not been found after a given number of time steps.

It is assumed that calling `Collision` every Δt, the moments of time each arm is allowed to change its velocity, is enough. However, if the atomic arm motions have a long duration, there is a chance that the arm being planned could jump over thin dynamic obstacles. In those cases that a finer resolution is required, a sub-multiple Δt_{col} of Δt should be used. Then, `Collision` would need to be redefined so that it accordingly interpolates the path connecting the current state-time state and its parent and checks its feasibility every Δt_{col}. Employing different resolutions can be very convenient as the local search can be accelerated (by reducing the branching factor) without compromising safety. Algorithms 10 and 11 determine the neighbors of the node being explored

Algorithm 9 Exploring a *state-time* node

Input:
$z = (s, ns, m, t)$: a *state-time* node
$nStop$: max. *start-stop* cycles allowed
T_f: max. *time* to explore
Output:
$\mathcal{N}$: a sequence with the neighbours of z sorted from the least promising to the most promising.
z.status: node status is updated to *UNSAFE* or *SAFE*

```
1:  function EXPLORE-NODE(z, nStop, T_f)
2:      N ← ∅
3:      if z.status = UNEXPLORED then          ▷ Has the node been explored?
4:          z.status ← UNSAFE
5:          if not COLLISION(s, t) then
6:              z.status ← SAFE                ▷ Node is collision-free, so update status
7:              if t < T_f then ▷ Get neighbours only if max. t has not been reached
8:                  N ← NEIGHBOURS(z, nStop)
9:              end if
10:         end if
11:     end if
12:     return N
13: end function
```

considering the *start-stop* cycles counter (*ns*). If the maximum value (*nStop*) has been reached then the method `Neigh-maxNS` dynamically computes the neighbors. Otherwise, it is done by `Neigh-DF`. As can be appreciated in line 2 of algorithm 11, the method `Neigh-maxNS` always generates a neighboring node containing the same motion action. In addition, if the arm is stopped ($m = 0$), then the neighbor advancing toward the destination edge is also returned (line 6). However, when *nStop* was not reached, the *ns* variable is updated in `Neigh-DF` and all three neighbors are stored in the particular order shown in line 33 of algorithm 11. This way of arranging the nodes along

with the *depth-search* nature of algorithm 8 ensures that the most promising motion action, the one approaching the arm toward the edge destination vertex (see label A in fig. 6.9a), is always explored first. Figure 6.10 illustrates how the method `Find-Edge-Traj` explores the state-time

Algorithm 10 Determining connections of a *state-time* node **(Part 1)**

Input: $z = (s, ns, m, t)$: a *state-time* node , $nStop$: the max. *start-stop* cycles allowed
Output: $\mathcal{N}$: a sequence containing the neighbours of z

```
1: function NEIGHBOURS(z, nStop)
2:     if ns = nStop then
3:         N ← NEIGH-MAXNS(z)        ▷ Returning neighbour in same direction
4:     else
5:         N ← NEIGH-DF(z)    ▷ Returning neighbouring nodes in depth-first order
6:     end if
7:     return N
8: end function
```

space. The effect of the state variable ns in the smoothness of the solution can be appreciated in fig. 6.11, where searches along a roadmap edge are performed with different maximum values for that variable.

6.4 Experimental Results

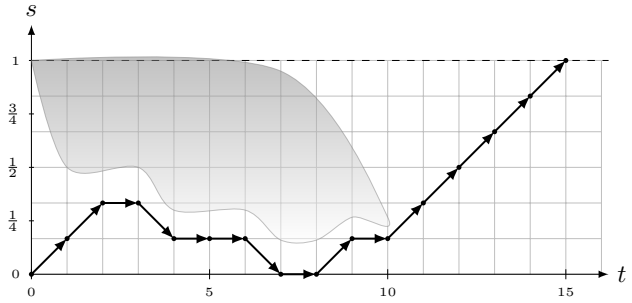
The proposed motion planning algorithm has been implemented in C++ along with the collision detection routines and other functions mentioned in section 6.3.2. We performed experiments in different scenarios with real science targets. But first of all, we will take the reader through one typical scenario of the MXS system in detail. There, during their motions, the arms were exposed to multiple instances of all identified collision conflicts that can occur in MIRADAS.

In all cases, the shapes of the arms have been geometrically modeled by three different polygons, each of them representing the link L_1, L_2 and L_3; see fig. 3.2. These polygons have been constructed with the help of the appropriate methods provided by the *Geometry Engine Open Source* (GEOS) framework, which is available in C++, Java, and Python (The Open Source Geospatial Foundation, 2013). The exact locations and orientations of these polygons are determined by the configuration q of the arm. This geometric engine also contains routines to check intersections between polygons. The corresponding `Collision(q,t)` methods mentioned in section 6.3.2 employ these intersection routines to check pair-wise conflicts between a given arm and the others in the system. Finally, the vertexes in a roadmap and the corresponding paths connecting any pair of them have to be validated. These validations are carry out in the associated `IsInZoA(q)` methods. In order to minimize the cost of intersection checking of any element in the roadmap against ZoA, we define an Oriented Bounding Box (OBB) encapsulating all points of this prohibited area. By using OBBs like the one appreciated in fig. 6.12, it allows us the implementation of fast rejection tests because we only check against real ZoA if the bounding

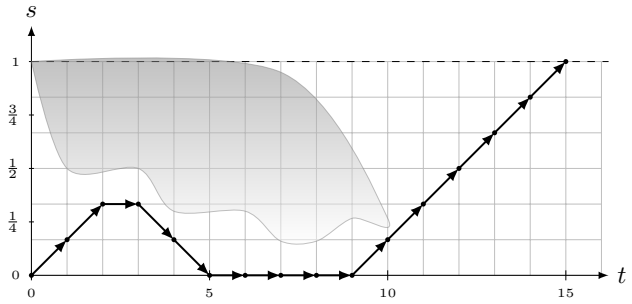
Algorithm 11 Determining connections of a *state-time* node **(Part 2)**

function NEIGH-MAXNS(z)
  $nz \leftarrow (s + m * \Delta s, ns, m, t + \Delta t)$ ▷ Neighbour keeps moving in the same direction
  nz.parent $\leftarrow z$ ▷ Store neighbour's parent state
  $\mathcal{N} \leftarrow \langle nz \rangle$
  if $m = 0$ **then** ▷ If arm was stopped
    $nz \leftarrow (s + \Delta s, ns, m, t + \Delta t)$ ▷ Add also the *forward* neighbour
    nz.parent $\leftarrow z$
    $\mathcal{N} \leftarrow \mathcal{N} + \langle nz \rangle$
  end if
end function

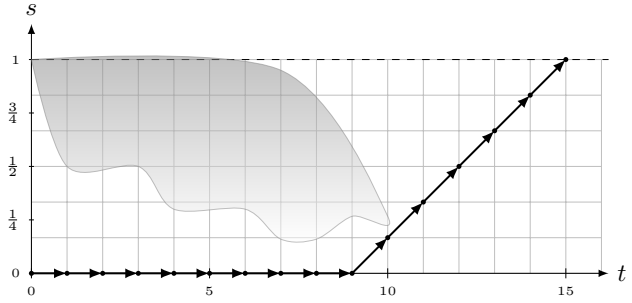
function NEIGH-DF(z)
  if m = -1 **then** ▷ The node being explored has been reached moving *backward* ($m = -1$)
    $ns_{-1} \leftarrow ns$ ▷ Num. *stop-start* cycles performed to reach the *backward* neighbour
    $ns_0 \leftarrow ns + 1$ ▷ Num. *stop-start* cycles performed to reach the *motionless* neighbour
    $ns_{+1} \leftarrow ns + 1$ ▷ Num. *stop-start* cycles performed to reach the *forward* neighbour
  end if
  if $m_i = 0$ **then** ▷ If arm was stopped, do not increment *stop-start* cycles
    $ns_{-1} \leftarrow ns$
    $ns_0 \leftarrow ns$
    $ns_{+1} \leftarrow ns$
  end if
  if $m_i = 1$ **then**
    $ns_{-1} \leftarrow ns + 1$
    $ns_0 \leftarrow ns + 1$
    $ns_{+1} \leftarrow ns$
  end if
  $nz_{-1} \leftarrow (s - \Delta s, ns_{-1}, -1, t + \Delta t)$ ▷ Computes *backward* neighbour
  nz_{-1}.parent $\leftarrow z$ ▷ Store neighbour's parent state
  $nz_0 \leftarrow (s, ns_0, 0, t + \Delta t)$ ▷ Computes *motion-less* neighbour
  nz_0.parent $\leftarrow z$
  $nz_{+1} \leftarrow (s + \Delta s, ns_{+1}, 1, t + \Delta t)$ ▷ Computes *forward* neighbour
  nz_{+1}.parent $\leftarrow z$
  $\mathcal{N} \leftarrow \langle nz_{-1} \rangle + \langle nz_0 \rangle + \langle nz_{+1} \rangle$ ▷ Store at the end of the sequence $\mathcal{N}$ the *forward* neighbour
  return $\mathcal{N}$
end function



(a) The search is guided by the maximum number of start-stop cycles (ns) allowed. In this scenario, there is no constraint in the maximum value the variable can take. Therefore, no smoothness control is considered when exploring the search space, yielding a final motion profile consisting of 9 real motion actions (and 4 start-stop cycles).



(b) Here, the maximum value of ns is constrained to 2, then the solution returned will perform 2 or less start-stop cycles. The real motion actions composing the trajectory are: *forward*, *stop*, *backward*, *stop* and *forward*.



(c) Here, the maximum value of ns is constrained to 0. The resulting arm motion profile is formed by a motionless action during 9 time steps followed by a forward action. This trajectory actually can be executed with a single real arm controller command.

Figure 6.11: How the variable ns impacts on the smoothness of the trajectory found along one edge.

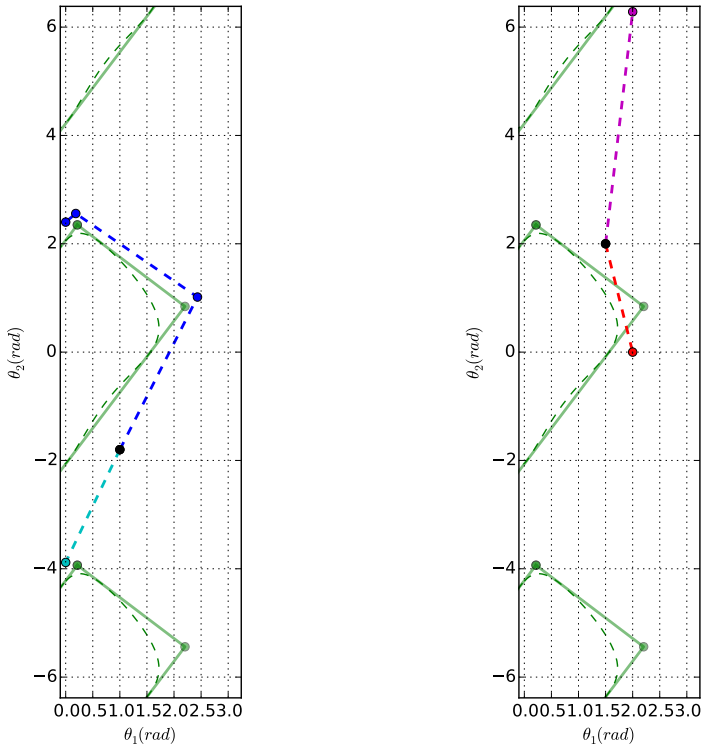
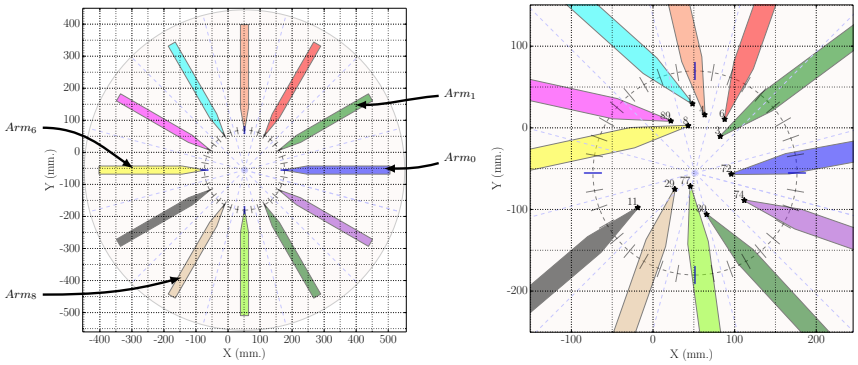


Figure 6.12: Two examples of paths in C-space. The ZoA area, delimited by a green dashed line, is enclosed in an OBB (light green).

area overlaps with the path or any vertex. The smallest-area box enclosing a convex polygon representing ZoA has been computed with the *rotating callipers* technique (Toussaint, 1983). As the real ZoA is a concave polygon, we previously transformed it into a convex one using the convex hull method provided by GEOS.

6.4.1 Preliminary Scenario in Detail

Here, we provide full details about how our algorithm works in a particular scenario. In fig. 6.13, the initial and final positions of the arms are shown. Although this problem example might look simple, during their motions, the arms were exposed to multiple occurrences of all identified collision conflicts that can appear in MIRADAS. Respectively, they are: (i) *goal conflicts* when the path of an arm passes by the goal location of another one and (ii) *en-route conflicts* when two paths intersect with each other, but the common point is not the goal location of any of them.



(a) Top view of the MXS system bench with the 12 arms located at their initial positions (default *park* locations). They are numbered, starting from Arm_0, counterclockwise.

(b) Detail of the MIRADAS system with the arms located at their final placements. The numbered black stars at the center of the figure are targets.

Figure 6.13: Typical scenario presenting *en-route* and *goal* collision conflicts.

Therefore, it can give the reader an idea of the versatility of the proposed approach.

A specific roadmap is built for each arm in the system. Each dimension of the 2D probe arm C-space was divided into equidistant segments, forming an 8×6 grid. Additional connections were set between random pairs of points in the grid, yielding more than 64 potential alternative routes. Then, the initial and final positions in C-space of each arm were connected to each of those points. Those paths passing through any infeasible configuration were automatically discarded. The maximum joint velocity used for each joint was $0.031\,416\,\mathrm{rad\,s^{-1}}$, which is the maximum achievable by the real mechanism as confirmed by laboratory tests. Additionally, we introduce two velocity reduction factors α of 0.5 and 0.25, incorporating to the system two extra velocity profiles to explore. The maximum number of *start-stop* cycles allowed per trajectory was 1. Finally, The collision buffer employed was of 5 mm.

By design, the running time of our solution tightly depends on the collision checking time step (Δt_{col}) and on the motion action time step (Δt). We employed different resolutions for each of them. The value of Δt was chosen taking into account the duration of one of the longest direct motions an arm can execute. That motion takes around 100 s, as empirically determined with real hardware. The value utilized for Δt was 6 s, assuring a good trade-off between arm maneuverability and graph branching factor. Finally, the value of Δt_{col} used in the experiment was 100 ms. That figure was considered small enough, in relation to the real velocities of the arms, to detect collisions between two different bodies effectively. Table 6.1 shows all simulation parameters. Note that T_f refers to the parameter introduced in algorithm 9; see section 6.3.4.5. It is applied as a decision variable to abruptly abort the search in a path segment if a successful

trajectory was not found after that relative number of time steps. Considering the values utilized for this simulation, the planner will discard a path segment if it cannot find a trajectory along it with duration less than 220 s. As stated in section 6.3.1, although the goal is to determine motion plans with a makespan no greater than 120, here, for the sake of studying the algorithm performance, larger values will be accepted.

One of the key points is how to assign the different priorities to the arms. If we have 12 probes in the system, there are 12! possible priority schemes. However, taking into account the requirements of the problem, it makes sense to adopt one that tries to minimize the *makespan* of the whole system. We will attempt it by following the strategy of planing first those arms whose shorter paths (in the roadmap) are the longest. In this way, we expect to give more freedom of action to those arms reaching their associated goals later. Therefore, the heuristic we used to assign priorities is the number of time steps Δt required to reach their goals if the arms follow its shortest path. The greater the number of time steps, the higher the priority.

We executed a single run of our algorithm with the priority policy defined above. A successful motion plan for all twelve arms was computed in 1.394 s on a 3.2GHz 2-Core Intel i5 with 4GB of RAM, as shown in table 6.2, requiring a total of 75668 collision checks. The column *States Explored* indicates the number of states dynamically explored during the graph search. The first state always checked is that corresponding with the starting position of the motion. As expected, the computation time for each arm trajectory depends on the number of collision checks performed, which is tightly coupled with the number of arms previously planned as well as the number of graph states visited. A trajectory was returned almost immediately for arm 6, the one with the highest priority since the dynamic environment was empty, and no collision checks were performed. However, for arm 10 more than 15000 checks were needed, yielding an execution time of 0.292 s. The number of alternative paths present in each arm roadmap was proven to be sufficient for this scenario, accurately representing, therefore, the arms C-space connectivity. Indeed, the first path explored for each of the arms was the one finally accepted.

Details about the trajectories found for each arm are given in table 6.3. The arms priorities, as well as their estimated travel times, are shown in columns *Priority* and *Est. Trav.*, respectively.

Table 6.1: Simulation Parameters for the Preliminary Scenario

Parameter	Value
J_1 max. velocity	$0.031416\,rad/s$
J_2 max. velocity	$0.031416\,rad/s$
Δt	6 s
Δt_{col}	100 ms
max. *start-stop* cycles	1
col. buffer	5 mm
T_f	220 Δt
potential paths in roadmaps	64
vel. reduction factors	0.5 and 0.25

Table 6.2: Running time and collision checks.

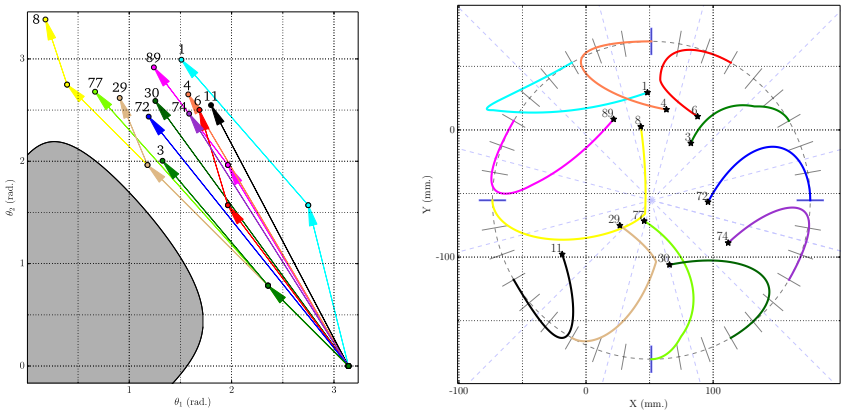
Arm	Priority	Running Time (s)	Paths Explored	States Explored	Col. Checks
6	12	0.005	1	20	0
4	11	0.029	1	19	1,142
5	10	0.082	1	19	2,164
9	9	0.059	1	18	3,186
3	8	0.081	1	16	4,144
8	7	0.138	1	22	6,106
10	6	0.292	1	45	15,722
7	5	0.126	1	15	6,967
2	4	0.117	1	16	7,816
11	3	0.165	1	15	8,649
0	2	0.145	1	14	9,490
1	1	0.158	1	14	10,282
Total	…	1.394	…	234	75,668
Average	…	0.116	…	29.5	6,305

At the right of those columns, we can distinguish three different groups of data. The first of them, named *Accepted Traj*, refers to the collision-free trajectories returned by our method. The second group and the third one respectively labeled *Optim. Traj* and *Roadmap*, show the duration of the optimal trajectory (in Δt units) and information about the paths in each arm's roadmap. The column labeled as *Path* specifies the route followed by each arm.

We want to make some remarks about a few trajectories determined by the method presented. As indicated in table 6.3, the motion of arm 6, the last one expected to arrive at a destination, took 19 simulation time steps Δt (114 s), whereas the optimal trajectory would have taken 18 Δt. For this arm, three different paths with minimal estimated travel time were present in its roadmap. The algorithm randomly selected one of them since that is the normal behavior in such cases as commented in section 6.3.4. The path first explored comprised two linear segments in the arm's C-space. The associated trajectory had only a single *start-stop* cycle, performed in the intermediate point A (see field *Path* in table 6.3), and during the execution of each segment there was not any additional stop (nΔt stop = 0). The C-space path followed by this and all other arms as well as the path followed by the centre of each of the pick-off mirrors can be visualized in

Table 6.3: Trajectories found.

			Accepted Traj.					Optim. Traj.	Roadmap		
Arm	Prio.	Est. Trav. (s)	Path	nΔt	nΔt go	nΔt stop	nStart-Stop	nΔt	Total	Min. nΔt	Avg. nΔt
6	12	107.75	start-A-end	19	19	0	1	18	32	19	21.34
4	11	95.24	start-B-end	17	17	0	1	16	33	16	17.88
5	10	92.86	start-C-end	17	17	0	1	16	33	16	18
9	9	85.29	start-D-end	16	16	0	1	15	33	15	18.50
3	8	84.46	start-end	15	15	0	0	15	33	15	16.85
8	7	83.34	start-E-end	16	15	1	1	14	33	14	17.51
10	6	82.42	start-end	16	14	2	0	14	33	14	16.79
7	5	81.12	start-end	14	0	0	0	14	34	14	16.67
2	4	79.63	start-F-end	14	14	0	1	14	34	14	16.38
11	3	78.45	start-end	14	14	0	0	14	33	14	16.33
0	2	77.55	start-end	13	13	0	0	13	33	13	16.78
1	1	63.82	start-G-end	12	12	0	1	11	33	11	15.82



(a) The C-space paths followed by the arms. The numbered dots are the same targets than the ones found in Figure 6.13. Arrows show the direction motion. All arms start their motions at the same configuration ($\theta_1 = \pi$ and $\theta_2 = 0$) and finish at their corresponding targets.

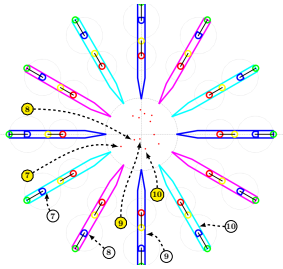
(b) The paths followed by the centres of the pickoff mirrors represented in the arms workspace, the MXS bench. A few arms such as number 8 take detours (discontinuities in their paths) to avoid *goal-conflicts*.

Figure 6.14: Paths in C-space and workspace for the typical scenario.

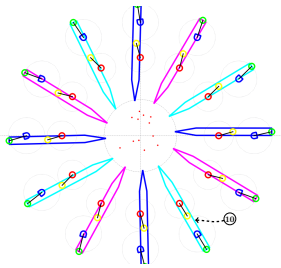
fig. 6.14a and fig. 6.14b respectively.

On the other hand, for arm 10, a safe, direct trajectory linking the initial and final configurations was found. The single motion can be seen in Figure 6.14b, where the pick-off mirror trajectory does not present, in contrast with arm 6, any discontinuity. The arm trajectory took a total of 16 Δt, during 2 of which the arm was stopped. These two Δt were carried out in the two first simulation time steps as no *start-stop* cycle was generated. Consequently, when the rest of arms started their respective motions, arm 10 remained in its initial position during 12 s. Finally, as an example of the versatility of the algorithm, it is worth mentioning the result found for arm 8. For this probe, a path with two segments was computed, yielding a *start-stop* cycle in the connecting point. During one of the 16 time steps that took the trajectory, the arm was stopped. This motion-less action was performed at the beginning of the second segment, remaining the arm at the initial configuration of that segment during 1 Δt.

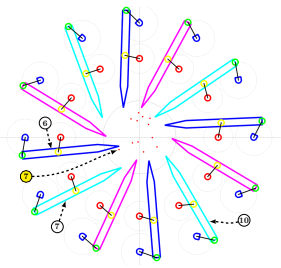
In addition to the previously presented algorithms, we have also developed a cross-platform graphical simulator. Implemented in *Qt*, this application virtually executes the trajectories of the arms in the system as computed by the roadmap-based motion planner. In fig. 6.15, we provide a few selected snapshots showing the sequence of the different motions performed by each arm for this concrete scenario.



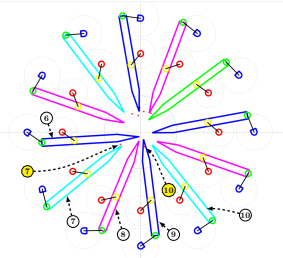
(a) The arms at their start positions. The dots in the center of the figure are the 12 targets. The numbered bold circles point to the target assigned to the corresponding arm.



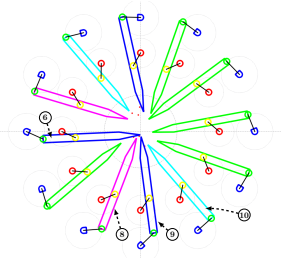
(b) Every probe, except arm 10, starts their motions simultaneously. Arm 10 remains in its initial position.



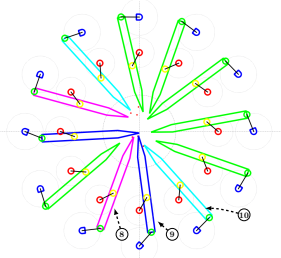
(c) Arm 10 is in motion. Arm 6 is approaching the arm 7 target. As long as arm 6 leaves behind that target before arm 7 reaches it, there will be no goal-conflict.



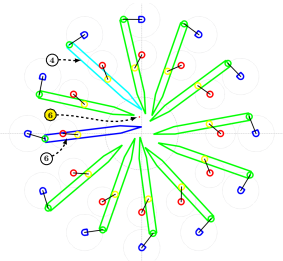
(d) Arm 1 has reached its destination. Arm 6 has left behind the target of arm 7. However, it is invading the arm 8 target and is about to do the same with arm 9 target. Therefore, arm 8 remains motionless. Additionally, arm 9 is over the arm 10 target.



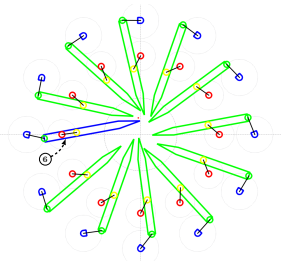
(e) More arms (0, 2, 7, 11) keep arriving at their destinations. Arm 6 is leaving behind the targets of arm 8 and 9. Moreover, Arm 9 is leaving also behind arm 10 target. Therefore, Arm 8, 9 and 10 can continue its motions.



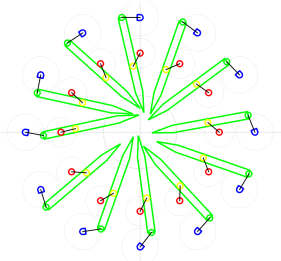
(f) Arms 8, 9 and 10 are approaching their respective targets. No arm is interfering in their paths, so their are able to safely reach their destinations.



(g) All arms, except number 4 and 6, are at their destinations.



(h) Arm 6 is about to reach its assigned target.



(i) All arms at their final positions.

Figure 6.15: The sequence of the motions of the 12 arms in the MXS bench.

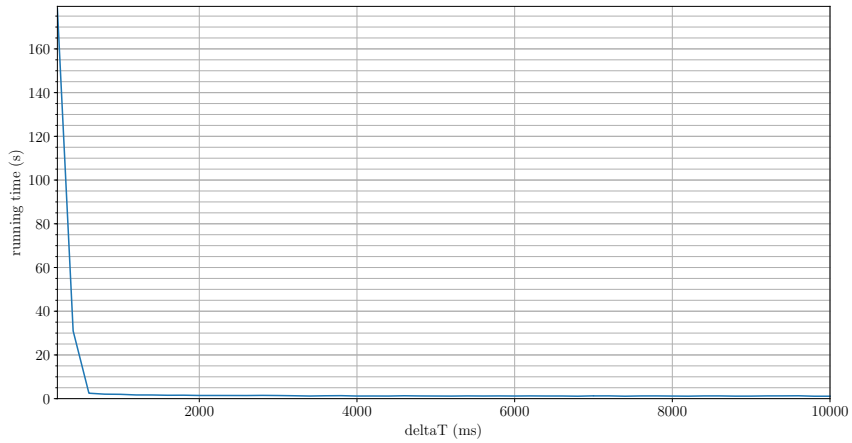
6.4.1.1 Modifying the Quantities

In this section, using the base scenario presented above, we study how the modification of relevant values affect the performance of the method proposed. The parameters that will be altered during these experiments are Δt, the number of start-stop cycles allowed and the number of paths in the roadmaps. If not explicitly mentioned, the reader should consider that only the concept under consideration is varied, using for the other parameters the values in table 6.1.

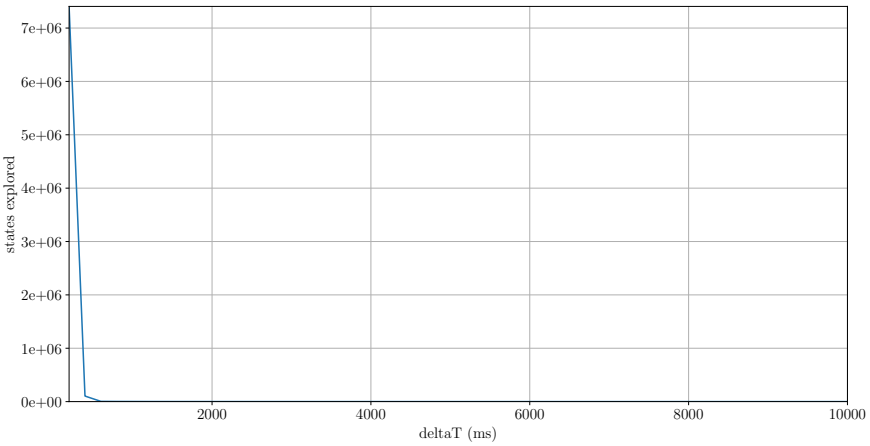
Time Step (Δt) Internally, our approach computes trajectories as a sequence of atomic motion primitives. As we have seen, the different motion primitives to be considered are moving forward, moving backward or remaining in the current location. The time step determines the duration of all possible primitives. Consequently, its value affects the branching factor of the search. The smaller the quantity, the larger the search space. In practice, the minimum value that this parameter can hold comes determined by the motion command reception rate supported by the motor controllers.

By construction, the method expects Δt to be a multiple of Δt_{col}. Hence, we start with a quantity of 200 ms, and we gradually increment it. In every instance of this experiment, the planner was able to find collision-free motions for all arms in the system. The effects of varying the time step on the algorithm running time as well as the number of states of the search space explored can be seen in fig. 6.16 and fig. 6.17 respectively. As expected, the value of Δt directly impacts on both. The smaller the time step, the larger the space search visited and, consequently, the greater the execution time of the method proposed. When the number of states explored is around 2250 or less (for Δt equal to or higher than 1 s), then running time of the solution is below 2 ms. The mean and standard deviation are respectively 1.311 s and 0.172 s for time steps equal or greater than 1 s. The minimum value for this interval is 1.118 s, while the maximum is 1.987 s. For lower values of the time step, running time increases significantly. For instance, execution time is 31.1 s and 178.24 s for a Δt of 400 ms and 200 ms respectively.

We turn our attention to how the parameter at hand impacts on the system makespan. As can be appreciated in fig. 6.18, the minimum value achieved is 108 s, which is relatively close to the estimated travel time for arm 6 (107.75 s) The makespan keeps constant at this value for time steps in the range from 200 ms to 600 ms, whereas a set of different values is obtained for larger Δt. Regarding this set, its mean value is 113.188 s and its standard deviation is 3.951, respectively being 108 s and 124 s the minimum and maximum makespan in this interval. Note that for a time step of 1200 ms, makespan is 108 s as well. Additionally, in a couple of time steps such as 3400 ms and 6800 ms a value close to the minimum is found. Concretely, 108.8 s. These differences in the makespan are primarily due to the acceptance of distinct paths from the respective roadmaps for the different values of the time step parameter. The time step parameter gives a trade-off between the speed of resolution and accuracy of the solution. The lower the value of Δt, the greater the maneuverability of the arms, and, therefore, the makespan computed

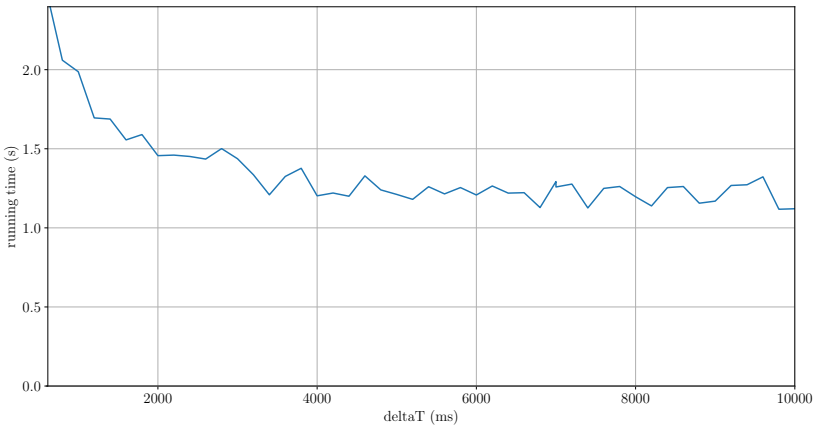


(a) The algorithm running time versus Δt.

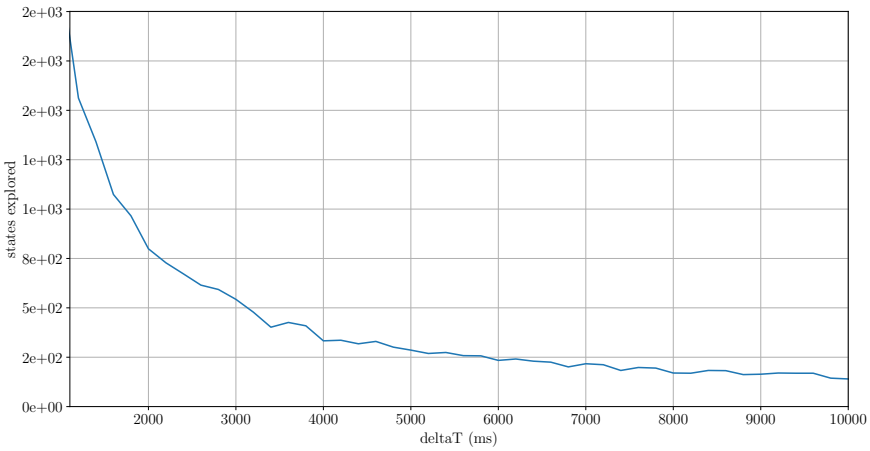


(b) The search space explored by the algorithm versus Δt.

Figure 6.16: Varying Δt parameter and its effects on running time and the number of states explored.



(a) Detail of the algorithm running time versus Δt.



(b) Detail of the search space explored by the algorithm versus Δt.

Figure 6.17: Details of varying Δt and its effects on running time and the states explored.

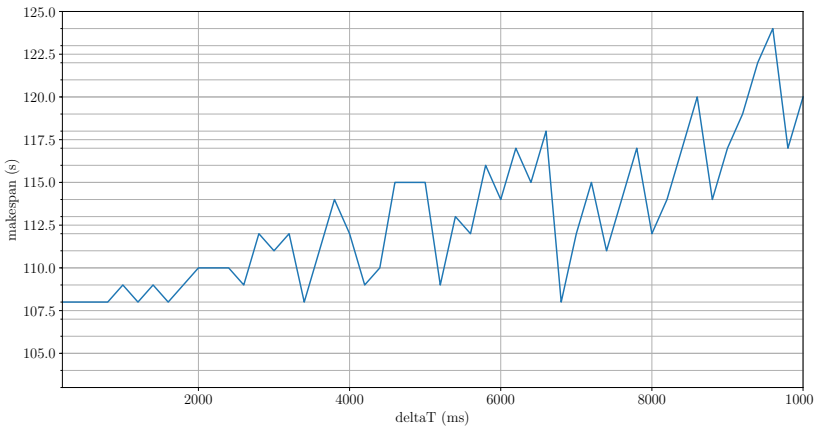
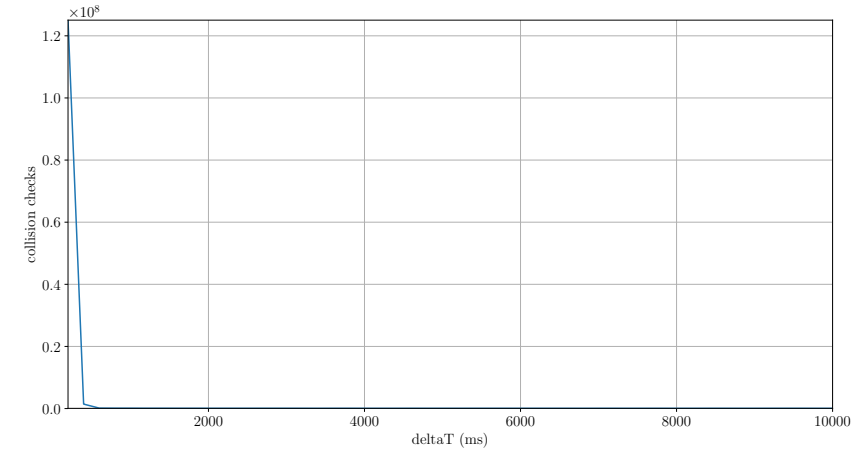


Figure 6.18: The impact of Δt on the system makespan.

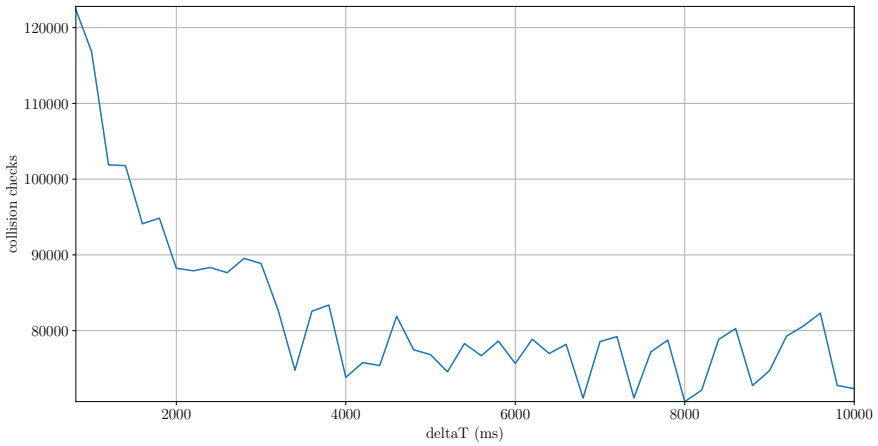
converges to a minimum.

In fig. 6.19, we study the relationship between the number of collision checks performed by the algorithm and the value of the parameter Δt. As can be expected, the collision checks notably increment as it does the time step and, consequently, the number of states visited. In the Δt interval [3500, 10000], the number of collision verifications carried out ranges from 70634 to 82564, being the difference between the limit values relatively small compared to ones for smaller time steps. However, despite this disparity in the checks, no relevant difference in running time is appreciated. On the other hand, the most significant increment detected between two consecutive samples separated by 200 ms occurs in the interval going from 400 ms to 200 ms. In this case the difference is 123607915 checks, being 125045903 the total for 200 ms. This remarkable difference influences the execution time of the algorithm for each of these time steps.

MIRADAS presents a two-step approach for acquisition. The first of them consists in the positioning of the probe arms by means of the execution of their associated trajectories. Then, once, the arms are at their final locations as determined by the motion planner, a second and coarser positioning stage is carried out. With the help of an image processing application iterating on the through-slit images, the differences between the expected arm positions and the ones where they have been left once applied the specified trajectories are calculated. These offsets are then corrected by slightly moving the arms in their close vicinity. Although the ultimate positioning accuracy required for the MXS probes implies a mechanical accuracy of 130 µm, given this two-step approach, the initial accuracy only needs to be around 500 µm (Eikenberry et al., 2014). Here, we conduct an experiment to determine how the simulation time step affects the



(a) Collision checks versus Δt.



(b) Detail of the number of collision checks versus Δt.

Figure 6.19: Varying Δt and its effects on the number of collision checks.

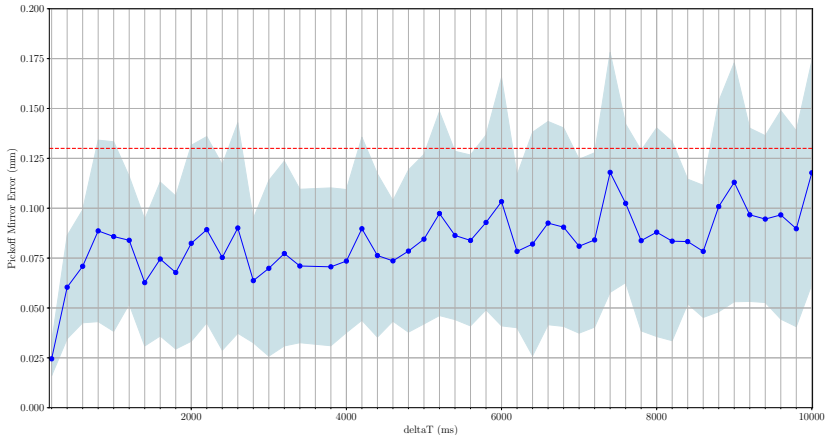


Figure 6.20: The impact of Δt on the pick-off mirror position accuracy. The mean error (in blue) as well as the 90% confidence interval (shadowed) is shown. The dashed line represents the mechanical accuracy required for the mechanism.

positioning of the pick-off mirrors.

In this accuracy test, like in the previous ones, we start with a value for Δt equal to 200 ms, and we gradually increment it by 200 ms. In each iteration of the motion planner, we transform the high-level trajectories determined into low-level ones and collision checks are performed accordingly. The first of these trajectories are expressed in terms of the parameters corresponding to the independent joints of each arm. That is, in function of θ_1 and θ_2. On the other hand, the low-level trajectories are specified in motor steps. The translation between both types has been performed using the degree/step ratio of each joint, an intrinsic parameter of the arm's mechanical design. The resulting values, once in steps, cannot be fractional; consequently, the slopes of the trajectories vary a little and errors due to rounding can also be produced. As a consequence of the multiple conversions, the positioning of the pick-off mirrors in the instrument FOV might experience errors. The positioning errors for this preliminary scenario are shown in fig. 6.20. There, we show the mean as well as the 90% confidence interval. As can be appreciated, for all Δt the error is smaller than 500 µm. Indeed, the mean value is also smaller than the mechanical accuracy -marked by the red dashed line in fig. 6.20- required.

Start-Stop Cycles By construction, the motion planner presented attempts to reach a given destination as soon as possible. As shown in fig. 6.11, if the number of start-stop cycles is unlimited, no smoothness control is applied, and, consequently, the resulting trajectories closely surround the obstacles. That is, the algorithm will naturally perform the minimum number of

Table 6.4: Varying the maximum number of *start-stop* cycles (ns) permitted - Δt = 6000 ms

max. *ns* allowed	*ns* performed	Run Time (s)	States Explored	Col. Checks	Makespan (s)
1	1	1.34738	234	75,668	114
2…20	2	1.06529	205	65,408	114

Table 6.5: Varying the maximum number of *start-stop* cycles (ns) permitted - Δt = 600 ms

max. *ns* allowed	*ns* performed	Run Time (s)	States Explored	Col. Checks	Makespan (s)
1	1	2.62498	4,550	144,327	108
2	2	1.08461	1,820	62,855	108
3	3	1.03767	1,797	62,280	108
4	4	1.03632	1,785	61,878	108
5…20	5	1.02877	1,779	61,740	108

start-stop cycles so that the arms get to their respective goal positions in the shorter amount of time. This behavior can be appreciated in table 6.4, where we show the results obtained by the planner when we vary the allowed number of this kind of cycles. In all cases where the value is greater than 1, the planner explores the same number of states. Indeed, not only the number is the same, but the inspected states as well. Hence, once reached the minimum necessary number of cycles, if we raise the value for this parameter, it will not affect the performance of the algorithm, which will always return the same trajectories. It is worth mentioning that although the makespan is constant, the trajectories determined for $ns = 1$ are different from those obtained for $ns > 1$. Finally, the algorithm execution time for $ns > 1$ is the mean of all instances.

The impact of the maximum number of start-stop cycles permitted can be seen if we reproduce the same experiments but this time with a much smaller value of Δt. For instance, 600 ms. The results obtained are in table 6.5. Here although makespan is constant as well (see fig. 6.11 for an example), the number of states explored for a single start-stop cycle exceeds twice those visited for 5 start-stop cycles. In fact, this higher time of execution is directly linked to the greater degree of backtracking that is often required by the algorithm when the number of start-stop cycles is low.

Paths in the Roadmaps Here we vary the number of paths present in the roadmaps. In each experiment, we add two more vertices in each dimension of the grid used to construct the roadmaps. The number of paths grows more or less proportionally. As seen in table 6.6, this increment does not have an impact on the makespan. Indeed, this is due to the priority policy used, which forces the arm with the longest estimated travel time to be the first planned. Since the scenario employed in all experiments is the same, the arm number 6, the last one arriving at its destination, always is assigned the shortest path available in the roadmap. This path once discretized adequately according to the time step used turns into a trajectory with a travel time of 114 s. This time, as demonstrated by experiments above, can be minimized to 108 s if a smaller value for Δt is chosen. Finally, although makespan keeps invariable in all instances, the

Table 6.6: Varying the roadmap size - Δt = 6000 ms

Vertices	Makespan (s)	Last Arm	Exp. Travel Time (s)	Total States Explored	Col. Checks
8×8	114	6	107.75	234	75668
10×10	114	6	107.75	251	78683
12×12	114	6	107.75	201	62942
14×14	114	6	107.75	210	65306
16×16	114	6	107.75	228	72299
18×18	114	6	107.75	210	65674
20×20	114	6	107.75	214	67165
22×22	114	6	107.75	209	65170
24×24	114	6	107.75	204	63312
26×26	114	6	107.75	229	71954
28×28	114	6	107.75	205	63832
30×30	114	6	107.75	206	64272
32×32	114	6	107.75	213	66745
34×34	114	6	107.75	205	63552

trajectories accepted for arms possessing lower priorities are different in many cases. This can be appreciated in the column of table 6.6 representing the number of states explored during the search, which presents no homogeneous values.

6.4.2 Real Science Objects

We use SGPBTA, the semi-greedy based allocator presented in chapter 5, and compute 200 different assignment plans containing real science objects. Then, we perform two different experiments with the plans. In the first of them, for each assignment plan, the arms start their motions from their corresponding park positions and end at their assigned targets. Before determining the trajectories for each of the plans, the arms are numerically prioritized according to the estimated travel of their corresponding optimal paths. The higher that time, the greater the priority assigned to the arm. We find that in the 97% of the assignment plans the algorithm successfully computes collision free-trajectories with a configuration time inferior to 120 s. The median configuration time is 102.55 s, ranging from a minimum of 96 s to a maximum of 114 s.

In the second experiment, all arms start their trajectories at their park locations and finish them at the targets of the first assignment plan. From there, the arms move to the next assignment plan and so on. As in the previous case, before determining the corresponding trajectories, arms are prioritized according to their estimated travel times. In this case, 93% of all assignments were successfully planed with a median configuration time of 90.6 s and a maximum of 108 s. The median value here is smaller than in the previous case because arms now start the next set of observations from a closer position. However, the success rate is inferior since arms are closer, and consequently, the probability of collision is higher.

Please, note that in all the previous tests, we have only run the motion planning algorithm once, as the purpose was to show the algorithm capability to find satisfactory solutions with the

initial priority scheme. As mentioned in different parts of the document, a frequent strategy to increase the success rate is to run several times the planner, changing the assigned arm priorities on each run, before declaring a given set of targets as failed. Refer to section 2.2.1.1 for a brief discussion about how a higher-level logic should proceed in such situations.

6.5 Conclusion

In this chapter, we surveyed the different approaches and techniques for motion planning in general as well as for multi-robot systems. Although an active area of study in the robotics community, motion planning is still incipient in astronomical instruments. With the advent of more and more sophisticated MOS facilities, this field of knowledge is expected to gain more attention.

MIRADAS possesses a multiplexing system with 12 cryogenic robotic probe arms, each capable of independently selecting a user-defined target in the instrument FOV. The arms are distributed around a circular bench, becoming this a very packed workspace when all of them are in simultaneous operation. Therefore, their motions have to be carefully coordinated. We proposed here an *off-line* motion planning approach for the probe arms of MIRADAS. Our method is based on an iterative solution that individually determines the trajectories of each arm, verifying its feasibility with the help of a collision checking routine. The algorithm uses roadmaps containing several alternative paths, increasing this way, the arms maneuverability. They are constructed and systematically explored with different time profiles. These profiles are achieved employing a discrete *state-time* graph, whose connections are dynamically built while a depth-first technique performs a search. This strategy intrinsically guides the search toward individual trajectories arriving at their destinations as soon as possible. Our approach, besides, considers the constraints imposed by the particular architecture of the probe arms as well as the limitations of the COTS motor controllers used in the mechanical design.

We tested our solution with real science targets and a typical MIRADAS scenario presenting some instances of the two identified collision conflicts that can arise between any pair of probe arms. Experiments showed that the method is versatile enough to compute trajectories fulfilling the requirements. In particular, simulations demonstrated that the algorithm with only one pass was able to find in 95% of the cases collision-free trajectories with a configuration time inferior to 120 s. In the scenarios where the planner failed, it was because the motions of a high priority arm prevented a lower priority one reaching its assigned target. This situation can occur in a prioritized approach since it is incremental in nature. Moreover, since the variety of real fields tends to be high in astronomical MOS instruments, difficulties can arise in some particular scenarios when employing planners incomplete by construction. That is the case also in other instruments using similar solutions, where some test configurations remained unsolved (Lorente et al., 2016). In real operation, our planner will be integrated into a more general strategy, which

will replan trajectories several times before declaring a given set of targets as unsuccessful.

CHAPTER 7

CONCLUSION AND FUTURE WORK

Many of the leading open questions in modern astrophysics can be only answered with the help of moderate to high spectral resolution instruments installed in large collecting area telescopes. As a result, the demand for instruments with MOS capabilities has significantly increased in the last decades. Developed by an international consortium, MIRADAS is a near-infrared multi-object echelle spectrograph for GTC operating at spectral resolution R=20,000. This latest generation astronomical instrument is one of the most powerful of its kind ever constructed. The combination of GTC collecting area and the multi-object spectra provided by MIRADAS make its capabilities unparalleled for addressing a large number of science cases. Among others, MIRADAS will facilitate the study of massive stars, black holes, the chemo-dynamics of Milky Way and blue compact dwarfs, which are crucial elements to understand galaxy evolution and assembly.

One of the most important components of MIRADAS is its multiplexing system, which enables the simultaneous observation of up to 12 user-defined sky locations. This task is accomplished by 12 independent and deployable integral field units based on robotic probe arms with pick-off mirrors. Working in a cryostat, each of these arms has inside a series of optical elements that relay light from the telescope focal plane to the spectrograph. Unlike other devices dedicated to gathering light in similar instruments, this internal optics is specially designed for simplicity. In particular, the optical path length is always maintained constant while the arm is moving between any two FOV locations. Furthermore, the tubes and bars forming the arm are arranged in a way that provides a high level of stability to the mechanical structure when it works upside down. Unfortunately, these optical and mechanical advantages negatively impact on the arm's control, which, due to the complicated arm kinematic behavior, is challenging.

The work in this thesis focuses on the design and implementation of motion techniques specific to MIRADAS. Like most of the existing MOS facilities, MIRADAS is a science-driven

instrument. As such, its performance requirements are defined by a few particular key design reference cases. However, special care has been taken in its design so that MIRADAS could be employed as a GTC common-user instrument for many years to come. Due to this common-user nature, MIRADAS presents a particular observation preparation process with one stage that, to the best of our knowledge, cannot be found in other related instruments. The solutions proposed in this document are tightly connected and influenced by this specific data flow. Roughly, there exist three main processing stages: field segmentation, target allocation, and motion planning. We will discuss each of them in this conclusion and provide some outlook for future work on these topics. Finally, as previously mentioned, what makes MIRADAS different from other similar instruments like KMOS is the structural design of the arms of the multiplexing system. Since the understanding of their behavior is vital to develop motion policies adequately adapted to the instrument, we devoted a chapter to study the probe arm. Hence, here, in this last chapter, we will also provide our final remarks about the MIRADAS opto-mechanical positioner.

However, before turning our attention to the discussion of the several contributions of this work, we would like to remember what economists like to say: “the perfect is the enemy of the good.” There is too much work to be done in each of the areas of this thesis. But not only there. The techniques presented should be seen as a first step, especially as a series of black boxes that should cooperate to successfully achieve a more general task. As it has been reminded throughout this document, the high-level computational logic interconnecting each of the three processing phases was out of the scope of this work. Therefore, for now, and for the future, one of the open questions to address is how to efficiently gluing together the field segmenter, the target allocator and the motion planner.

7.1 MXS Probe Arm

In chapter 3, we introduced the MXS probe arm. We proposed a linkage model representing the arm’s kinematic behavior. Although the mechanism possesses four joints, the model, like the real motion of the arm, is only governed by two of them. More specifically, these two DoFs correspond to the rotation angles of the two rotational joints driven by stepper motors. Also, we showed the arm configuration space, and how, because of the closed kinematic chain nature of the mechanism, a prohibited zone arises there. The workspace of the arm was presented as well as a formulation for the forward kinematics problem and the inverse one. Moreover, we demonstrated that several points of the arm workspace can be reached employing two different configurations. This fact, if conveniently exploited, leads to the design of more robust motion planning solutions. We also showed that due to the considerable range of motion of the arms, several field-of-regard patrolling strategies could be implemented. To be precise, two of them, slice-of-pie and workspace patrolling, were discussed. Finally, although not directly concerning the mechanical design of the probe arm, we added a subsection about the COTS controller used to drive the motorized joints. Particularly,

we compared its functionalities with how in industry and academia are typically managed the low-level motion actions reproducing high-level trajectories. This material is included partly because of his inherent interest and especially to emphasize the hard constraints the selected controller imposes on the design of high-level motion strategies. As commented, due to the limited functionality of this industrial controller, the number of intermediate points in a trajectory should be minimized to avoid excessive wear in the mechanisms.

An aspect that is not considered in this work is how the gravity vector affects the control of each arm in the system. MIRADAS will be attached to GTC's Folded Cassegrain foci through a field rotator device. One of the main functions of this rotator will be to enable the positioning of the MXS bench in distinct vertical orientations. Interesting future work would be to extend the model to include additional parametrization to accommodate gravity information provided by the telescope control system adequately. A further future challenge is to select more advanced COTS motor controllers to overcome the limitation of the current one. That is, to avoid stopping in every intermediate point of a trajectory composed of several way-points. Basically, this alternative controller should be capable of executing parabolic blends through a sequence of points or any other related interpolation method. On the other hand, another interesting point could be the design of a custom electronic board and the corresponding embedded software to control the steppers directly. This way, any sophisticated trajectory could be implemented as well as a more accurate low-level synchronization between all arms in the system might be achieved.

7.2 Field Segmentation

We explored the idea of field segmentation in chapter 4. As stated there, due to MIRADAS inherent GTC common-user characteristic, we have identified this as a remarkable and distinctive area. To the best of our knowledge, this sort of processing is not of interest in most of the related instruments that the authors are studied. In general, astronomical instruments are mainly designed having in mind a very particular set of science cases and are not expected to be functional to address any others not previously considered. Additionally, the instrument tools provided to astronomers to prepare their observations assume that the user will manually gather together the targets so that all of them fit in the instrument FOV. Then, the positioners motions can be computed following the traditional two-step sequential process used in many MOS. First, target allocation is run, and then motion planning determines the trajectories for each assignment plan previously returned by the allocator. However, MIRADAS does not severely constrain the fields it can work with. The idea behind is that targets will be selected by scientists mainly considering their celestial locations, yielding, in many cases, fields where objects are scattered over a wide area of the sky.

In this thesis, we demonstrated how fields of that kind can be partitioned to be properly observed with MIRADAS. Specifically, we proposed a preprocessing step where user-defined

celestial locations are automatically grouped so that all those in the same group fit in the FOV area. This is achieved by employing a hierarchical agglomerative clustering algorithm returning also the center of each cluster, which will be used as telescope's pointing information. This technique successfully partitions the data set of interest as well as creates some overlapping areas that might help to redistribute load among different clusters. However, our solution shows some degree of bias in the distribution of targets, especially in the clusters concentrating those targets located in the edges of the fields. The members of these clusters, unfortunately, are unevenly distributed, which might lead to poor performance of the target allocator.

In a broader perspective, the work presented is only the first step toward a clustering technique more suitable for the particularities and processing data flow of MIRADAS. Future research would include modeling the center of each cluster in a way that the load of each group is balanced more homogeneously. In this direction, we think it could be interesting to explore distinct classical clustering approaches such as Gaussian Mixture Models (see section 4.2.2 for a brief introduction). Using this method in conjunction with a proper high-level computational logic, we could determine clusters where the majority of the targets are concentrated around their centers. Finally, another interesting future work would be to employ the Mixture Model framework but with different probability distributions more adapted to some particular MIRADAS science cases.

7.3 Target Allocation

One of the achievements of this thesis is the proposal and comparison of two different target allocation methods. In fact, as commented in chapter 5, the process of assigning a job (target) to a worker (probe arm) is a fundamental combinatorial optimization problem present in many areas of knowledge. Generally, to solve combinatorial optimization problems, which are typically exponential in terms of time complexity, researchers have relied on either exact or approximate methods. In the first category, integer (or mixed-integer) programming and any variant of the *branch and bound* paradigm (Land and Doig, 1960; Little et al., 1963), among others, are frequently employed, whereas in the second heuristics (including metaheuristics) are more popular. Taking into account the existence of these two broad families, we presented two solutions. First, we proposed a semi-greedy patrol-based target allocator that returns plans that might not be globally optimal. Second, an allocator based on integer programming, which, by construction, determines the best possible plan.

Although an exact technique was previously utilized in KMOS, we consider that the model was not applied in the most natural way. According to the standard formulation of an assignment problem, each job has an associated prize giving an idea of its absolute importance. Consequently, these prizes can be assimilated as the priority of each of the targets in the data set. However, the KMOS allocator used these weights to theoretically (but not practically) avoid collision between the arms assigned. Contrarily, we followed the standard approach and also extended the model to

be able to work with a number of targets different from the number of positioners in the system. Moreover, we introduced a set of constraints that forced the model only to accept pairings not leading to actual conflicts. We demonstrated this model in several synthetic scenarios as well as a few selected MIRADAS science cases. In all of them, the allocator presents a good performance in terms of arm yield, field completeness, and priority completeness. To accomplish its task, the model needs a compatibility matrix that should be precomputed. This matrix efficiently fulfills its mission and for moderately large fields is computed in seconds. However, for scenarios with thousands of sky locations tend to take a few minutes. But, the bottleneck of the exact MIRADAS allocator is in the nature of the model itself. With sets of a few hundreds of targets, the number of decision variables although larger, is still manageable and allocation plans are returned in seconds. However, for huge fields, the processing time might be excessive, making, in some cases, this allocator impractical. In such situations, the semi-greedy algorithm proposed returns an inferior but still good enough solution in a decent amount of time.

A future challenge is to redefine the model so that it naturally takes into account also the *path-aware* criterion mentioned in section 5.1. Concretely, it would imply the implementation of a method for solving a *multicriteria* combinatorial optimization problem modeled as a mixed-integer program. As such, in addition to the already presented objective function maximizing the aggregated priority - see eq. (5.9) - a second one should be optimized as well. To be precise, this should be formed by a single variable, for example, holding the longest trajectory in a solution. Then, this bicriterion problem could be solved by employing a hybrid approach combining the *weighting method* with additional constraints (Riera-Ledesma and Salazar-González, 2005).

An interesting extension to our metaheuristic method is to include additional selection operators such as those typically found in the reproduction phase of Genetic Algorithms (Gendreau and Potvin, 2010). Broadly speaking, selection operators are generally categorized as proportionate or as ordinal-based. In the former family, an individual is picked according to its raw fitness relative to the fitness of the other members of the population, whereas in the latter the selection is based on its rank. For instance, roulette wheel, stochastic universal sampling or tournament (Blickle and Thiele, 1995; Goldberg and Deb, 1991; Pandey, 2016), widely used in artificial intelligence related literature, might be implemented. As in the integer programming allocator, further improvement would also be to apply a more elaborated metaheuristic capable of multiple criteria optimization. A step in this direction would be to implement a Multi-Objective Ant Colony based optimizer, which is an effective problem-solving strategy for a wide range of domains (Angus and Woodward, 2009).

Finally, another aspect to consider is the possible rotation of the FOV. We believe that even rotations of a few degrees might increase the performance of both allocators. This extension could be straightforwardly implemented since it would be as simple as running in parallel several instances of the allocators but with the arms locations slightly shifted from their original positions.

7.4 Motion Planning

In chapter 6, we presented a motion planning technique that given a set of 12 initial and final MXS probe arm positions, computes collision-free trajectories for the arms in the system. Our method is iterative. It assigns a priority to each arm and in descending order tries to find a safe trajectory for each of them independently. A trajectory for an arm is planned avoiding the previously planned arms with higher priority, that are considered as moving obstacles. As any decoupled approach, the method is not complete in terms of the problem formulation. Therefore, scenarios might be found in which successfully trajectories exist but, unfortunately, our method will not be able to determine them. Several research studies show that the results obtained by this approach are very sensitive to the priority policy applied. However, we encountered in practice our solution works fine in many cases with an intuitive priority strategy in which each arm is given a priority according to the length of the optimal path connecting the corresponding starting and final location.

The prioritized approach relies on a single arm motion planner for dynamic environments. This planner samples the arm's control space and constructs a roadmap comprising several alternative paths. Then, this roadmap is systematically explored in a state-time space with the help of a depth-first search strategy, where the number of changes in arm's velocity is strictly monitored and restricted. This last point is a significant contribution of this work, and it has been vital to dynamically adapt the solution search space to the probe arm controller constraints. Besides, the algorithm uses two different parameters, the principal simulation step (Δt) and the collision check step (Δt_{col}), to perform a discrete time analysis. While the former constraints the duration of the atomic motions an arm can perform, the latter, in practice, controls the number of collision checks to carry out per atomic motion. The decoupling of both concepts enables our approach to have an extra level of customization. By adequately tuning both, we can achieve a proper trade-off between branching factor, collision detection, and execution time.

Trajectories for a typical MIRADAS scenario were computed in a few seconds. As demonstrated by experiments, the computation time of a trajectory is tightly bounded to the number of collision checks. The number of checks, in general, increases in every iteration of the algorithm since there are more arms previously planned. Furthermore, the run time of the proposed algorithm is also dependent on the branching factor. The smaller the value of Δt, the higher the degree of maneuverability of the arms. However, an increase in maneuverability comes always at the expense of an increment in running time. On the other hand, the travel time of the trajectories is always expressed in multiples of Δt. Thus, by approaching that step to zero, these times can be compacted more and more. But, as expected, that might also affect negatively on the computation time of the solutions.

Simulations for several real science fields shown that the algorithm presented with only one pass was able to find in 95% of the cases collision-free trajectories. In the scenarios where the planner failed, it was because the motions of a high priority arm prevented a lower priority one

reaching its assigned target. This is a situation that can occur in planners of this kind since it is incremental in nature. A future step would be addressing this fundamental issue by exploring and studying in depth different priority schemes, including priority rotation policies, and replanning motions accordingly. The approaches of Bennewitz et al. (2002), Silver (2005), and Andreychuk and Yakovlev (2018) look promising in this direction. A further future challenge is to make this planner use an incremental coordination space to vary velocities of anteriorly planned robots when conflicts arise. The dimensionality of the coordination space would be higher than that of the space-time space of prioritized planning. But, it would definitely be much smaller than the joint composite space of a coupled approach. Although this extension would be still incomplete, it should fail less (Saha and Isto, 2006). Additionally, in an attempt to keep incrementing the initial success rate of the motion planner, it could also be considered to include in each roadmap a reserved "garage" location specific to each arm. Circulation along such areas would be only allowed to its respective owners. Finally, future research could also include, once an arm has reached its goal, opportunistically moving it away to leave room for other arms to pass. In this way, the occasions where higher priority arms prevent motions of the lower ones will be decremented.

In other perspective, improvements could also be carried out in the single-motion planner algorithm. For instance, applying an anytime heuristic graph search algorithm (Likhachev et al., 2003) where the maximum number of stops could dynamically vary depending on how promising a roadmap path is. In this context, exploring an approach based on the application of motion primitives together with a discrete search-based strategy in the arm's configuration space could be a good first attempt to naturally incorporate the arm's dynamic constraints into the motion planner (Cohen et al., 2010, 2011).

BIBLIOGRAPHY

Aarts, E., Aarts, E. H., and Lenstra, J. K. (2003).
Local search in combinatorial optimization.
Princeton University Press.

Aggarwal, C. C. and Reddy, C. K. (2013).
Data Clustering: Algorithms and Applications.
Chapman & Hall/CRC, 1st edition.

Akella, S. and Peng, J. (2004).
Time-scaled coordination of multiple manipulators.
In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3337–3344. IEEE.

Amiryan, J. and Jamzad, M. (2015).
Adaptive motion planning with artificial potential fields using a prior path.
In *2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM)*, pages 731–736. IEEE.

Anderberg, M. R. (1973).
Cluster analysis for applications.
Technical report, Office of the Assistant for Study Support Kirtland AFB N MEX.

Anderson, J., Mieske, S., and Kaufer, A. (2017).
FORS2 User Manual Issue 101 (VLT-MAN-ESO-13100-1543).

Andreychuk, A. and Yakovlev, K. (2018).
Two Techniques That Enhance the Performance of Multi-robot Prioritized Path Planning.
In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, pages 2177–2179, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.
event-place: Stockholm, Sweden.

Angus, D. and Woodward, C. (2009).
Multiple objective ant colony optimisation.

Swarm intelligence, 3(1):69–85.

Arnay, R., Morales, N., Morell, A., Hernandez-Aceituno, J., Perea, D., Toledo, J. T., Hamilton, A., Sanchez-Medina, J. J., and Acosta, L. (2016).
Safe and Reliable Path Planning for the Autonomous Vehicle Verdino.
IEEE Intelligent Transportation Systems Magazine, 8(2):22–32.

Auletta, V., Monti, A., Parente, M., and Persiano, P. (1999).
A linear-time algorithm for the feasibility of pebble motion on trees.
Algorithmica, 23(3):223–245.

Bacon, R. and Monnet, G. (2017).
Optical 3D-Spectroscopy for Astronomy.
John Wiley & Sons.

Ball, N. M. and Brunner, R. J. (2010).
Data mining and machine learning in astronomy.
International Journal of Modern Physics D, 19(07):1049–1106.

Barden, S. C. and Armandroff, T. (1995).
Performance of the WIYN fiber-fed MOS system: Hydra.
pages 56–67, Orlando, FL.

Barraquand, J. and Latombe, J.-C. (1991).
Robot Motion Planning: A Distributed Representation Approach.
Int. J. Rob. Res., 10(6):628–649.

Bennett, R. J., Davidson, G. H., Rees, P., and Todd, S. P. (2008).
Mechanical design and testing of the cryogenic pick-off arms for the VLT KMOS.
In Atad-Ettedgui, E. and Lemke, D., editors, *Proc. SPIE 7018, Advanced Optical and Mechanical Technologies in Telescopes and Instrumentation, 70182A (July 14, 2008)*, volume 7018, page 70182A, Marseille, France. SPIE.

Bennewitz, M., Burgard, W., and Thrun, S. (2001).
Constraint-based optimization of priority schemes for decoupled path planning techniques.
In *Annual Conference on Artificial Intelligence*, pages 78–93. Springer.

Bennewitz, M., Burgard, W., and Thrun, S. (2002).
Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots.
Robotics and Autonomous Systems, 41(2–3):89 – 99.
Ninth International Symposium on Intelligent Robotic Systems.

Benzécri, J. (1982).
Construction d'une classification ascendante hiérarchique par la recherche en chaîne des voisins réciproques.
Cahiers de l'analyse des données, 7(2):209–218.

Berg, M. d., Cheong, O., Kreveld, M. v., and Overmars, M. (2008).
Computational Geometry: Algorithms and Applications.
Springer-Verlag, Santa Clara, CA, USA, 3rd ed. edition.

Bershady, M. A. (2010).
3d Spectroscopic Instrumentation.
3D Spectroscopy in Astronomy, pages 87–125.

Billingsley, J. (2006).
Essentials of Mechatronics.
John Wiley & Sons.

Blickle, T. and Thiele, L. (1995).
A comparison of selection schemes used in genetic algorithms.
TIK-report.

Bobrow, J. E., Dubowsky, S., and Gibson, J. (1985).
Time-optimal control of robotic manipulators along specified paths.
The international journal of robotics research, 4(3):3–17.

Bode, M. (2012).
The ASTRONET Infrastructure Roadmap: A Strategic Plan for Astronomy in Europe.
Organizations, People and Strategies in Astronomy, 1:39–53.

Borne, K. (2009).
Scientific data mining in astronomy.
arXiv preprint arXiv:0911.0505.

Bounini, F., Gingras, D., Pollart, H., and Gruyer, D. (2017).
Modified artificial potential field method for online path planning applications.
In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 180–185. IEEE.

Bruce, J. and Veloso, M. (2002).
Real-time randomized path planning for robot navigation.
In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2383–2388 vol.3.

Bruynooghe, M. (1977).

Méthodes nouvelles en classification automatique des données taxinomiques nombreuses.
Statistique et Analyse des données, 3:24–42.

Bruzzone, L., Lizzi, L., Marchetti, P., Earl, J., and Milnes, M. (2004).
Recognition and detection of impact craters from EO products.
Proceedings of ESA-EUSC.

Buckley, S. (1989).
Fast motion planning for multiple moving robots.
In *Robotics and Automation, Proceedings, 1989 IEEE International Conference on*, pages 322–326 vol.1.

Buschkamp, P., Seifert, W., Polsterer, K., Hofmann, R., Gemperlein, H., Lederer, R., Lehmitz, M., Naranjo, V., Ageorges, N., Kurk, J., Eisenhauer, F., Rabien, S., Honsberg, M., and Genzel, R. (2012).
LUCI in the sky: performance and lessons learned in the first two years of near-infrared multi-object spectroscopy at the LBT.
page 84465L, Amsterdam, Netherlands.

Calcines, A., Harris, R. J., Haynes, R., and Haynes, D. (2018).
The reformatting advantage: Photonics vs conventional optics!
In *Advances in Optical and Mechanical Technologies for Telescopes and Instrumentation III*, volume 10706, page 107062G. International Society for Optics and Photonics.

Canny, J. (1988).
The complexity of robot motion planning.
MIT press.

Carlin, J. L., Lépine, S., Newberg, H. J., Deng, L.-C., Beers, T. C., Chen, Y.-Q., Christlieb, N., Fu, X.-T., Gao, S., Grillmair, C. J., and others (2012).
An algorithm for preferential selection of spectroscopic targets in LEGUE.
Research in Astronomy and Astrophysics, 12(7):755.

Carpin, S. and Pagello, E. (2002).
On parallel RRTs for multi-robot systems.
In *Proc. 8th Conf. Italian Association for Artificial Intelligence*, pages 834–841.

Chand, S. and Doty, K. L. (1985).
On-line polynomial trajectories for robot manipulators.
The International Journal of Robotics Research, 4(2):38–48.

Chapin, E. L., Dunn, J., Weiss, J., Gillies, K., Hayano, Y., Johnson, C., Larkin, J., Moore, A., Riddle, R. L., Sohn, J. M., Smith, R., Suzuki, R., Walth, G., and Wright, S. (2016).

The Infrared Imaging Spectrograph (IRIS) for TMT: motion planning with collision avoidance for the on-instrument wavefront sensors.
page 99130T.

Chekanov, S. (2006).
A new jet algorithm based on the k-means clustering for the reconstruction of heavy states from jets.
The European Physical Journal C-Particles and Fields, 47(3):611–616.

Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G. A., Burgard, W., Kavraki, L. E., and Thrun, S. (2005).
Principles of Robot Motion: Theory, Algorithms, and Implementations.
MIT Press, Cambridge, MA.

Cirasuolo, M., Afonso, J., Carollo, M., Flores, H., Maiolino, R., Oliva, E., Paltani, S., Vanzi, L., Evans, C., Abreu, M., Atkinson, D., Babusiaux, C., Beard, S., Bauer, F., Bellazzini, M., Bender, R., Best, P., Bezawada, N., Bonifacio, P., Bragaglia, A., Bryson, I., Busher, D., Cabral, A., Caputi, K., Centrone, M., Chemla, F., Cimatti, A., Cioni, M.-R., Clementini, G., Coelho, J., Crnojevic, D., Daddi, E., Dunlop, J., Eales, S., Feltzing, S., Ferguson, A., Fisher, M., Fontana, A., Fynbo, J., Garilli, B., Gilmore, G., Glauser, A., Guinouard, I., Hammer, F., Hastings, P., Hess, A., Ivison, R., Jagourel, P., Jarvis, M., Kaper, L., Kauffman, G., Kitching, A. T., Lawrence, A., Lee, D., Lemasle, B., Licausi, G., Lilly, S., Lorenzetti, D., Lunney, D., Maiolino, R., Mannucci, F., McLure, R., Minniti, D., Montgomery, D., Muschielok, B., Nandra, K., Navarro, R., Norberg, P., Oliver, S., Origlia, L., Padilla, N., Peacock, J., Pedichini, F., Peng, J., Pentericci, L., Pragt, J., Puech, M., Randich, S., Rees, P., Renzini, A., Ryde, N., Rodrigues, M., Roseboom, I., Royer, F., Saglia, R., Sanchez, A., Schiavon, R., Schnetler, H., Sobral, D., Speziali, R., Sun, D., Stuik, R., Taylor, A., Taylor, W., Todd, S., Tolstoy, E., Torres, M., Tosi, M., Vanzella, E., Venema, L., Vitali, F., Wegner, M., Wells, M., Wild, V., Wright, G., Zamorani, G., and Zoccali, M. (2014).
MOONS: the Multi-Object Optical and Near-infrared Spectrograph for the VLT.
page 91470N, Montréal, Quebec, Canada.

Clark, C. M., Bretl, T., and Rock, S. (2002).
Applying kinodynamic randomized motion planning with a dynamic priority system to multi-robot space systems.
In *Aerospace Conference Proceedings, 2002. IEEE*, volume 7, pages 7–3621. IEEE.

Cohen, B. J., Chitta, S., and Likhachev, M. (2010).
Search-based planning for manipulation with motion primitives.
In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2902–2908. IEEE.

Cohen, B. J., Subramania, G., Chitta, S., and Likhachev, M. (2011).
Planning for manipulation with adaptive motion primitives.
In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5478–5485. IEEE.

Connolly, A., Genovese, C., Moore, A., Nichol, R., Schneider, J., and Wasserman, L. (2000).
Fast algorithms and efficient statistics: density estimation in large astronomical datasets.
arXiv preprint astro-ph/0008187.

Constantinescu, D. and Croft, E. A. (2000).
Smooth and time-optimal trajectory planning for industrial manipulators along specified paths.
Journal of robotic systems, 17(5):233–249.

Cook, W. J., Cunningham, W. H., Pulleyblanck, W. R., and Schrijver, A. (1998).
Combinatorial Optimization.
Wiley-interscience series in discrete mathematics. John Wyley & Sons, Inc.

Cormen, T. H., Stein, C., Rivest, R. L., and Leiserson, C. E. (2009).
Introduction to Algorithms.
Computer Science and Intelligent Systems. The MIT Press, 3rd edition.

Coxeter, H. S. M., Coxeter, H. S. M., Coxeter, H. S. M., and Coxeter, H. S. M. (1969).
Introduction to geometry, volume 136.
Wiley New York.

Craig, J. J. (2005).
Introduction to robotics: mechanics and control.
Pearson/Prentice Hall, Upper Saddle River, N.J.

Cui, R., Gao, B., and Guo, J. (2012).
Pareto-optimal coordination of multiple robots with safety guarantees.
Autonomous Robots, 32(3):189–205.

Dalton, G., Trager, S. C., Abrams, D. C., Carter, D., Bonifacio, P., Aguerri, J. A. L., MacIntosh, M., Evans, C., Lewis, I., Navarro, R., Agocs, T., Dee, K., Rousset, S., Tosh, I., Middleton, K., Pragt, J., Terrett, D., Brock, M., Benn, C., Verheijen, M., Cano Infantes, D., Bevil, C., Steele, I., Mottram, C., Bates, S., Gribbin, F. J., Rey, J., Rodriguez, L. F., Delgado, J. M., Guinouard, I., Walton, N., Irwin, M. J., Jagourel, P., Stuik, R., Gerlofsma, G., Roelfsma, R., Skillen, I., Ridings, A., Balcells, M., Daban, J.-B., Gouvret, C., Venema, L., and Girard, P. (2012).
WEAVE: the next generation wide-field spectroscopy facility for the William Herschel Telescope.
page 84460P, Amsterdam, Netherlands.

Davies, R. L., Allington-Smith, J. R., Bettess, P., Chadwick, E., Content, R., Dodsworth, G. N., Haynes, R., Lee, D., Lewis, I. J., Webster, J., Ettedgui-Atad, E., Beard, S. M., Ellis, M. A., Hastings, P. R., Williams, P. R., Bond, T., Crampton, D., Davidge, T. J., Fletcher, J. M., Leckie, B., Morbey, C. L., Murowinski, R. G., Roberts, S. C., Saddlemyer, L. K., Sebesta, J., Stilburn, J. R., and Szeto, K. (1997).
GMOS: the GEMINI Multiple Object Spectrographs.
pages 1099–1106, Landskrona/Hven, Sweden.

de Jong, R. S., Barden, S. C., Bellido-Tirado, O., Brynnel, J. G., Frey, S., Giannone, D., Haynes, R., Johl, D., Phillips, D., Schnurr, O., Walcher, J. C., Winkler, R., Ansorge, W. R., Feltzing, S., McMahon, R. G., Baker, G., Caillier, P., Dwelly, T., Gaessler, W., Iwert, O., Mandel, H. G., Piskunov, N. A., Pragt, J. H., Walton, N. A., Bensby, T., Bergemann, M., Chiappini, C., Christlieb, N., Cioni, M.-R. L., Driver, S., Finoguenov, A., Helmi, A., Irwin, M. J., Kitaura, F.-S., Kneib, J.-P., Liske, J., Merloni, A., Minchev, I., Richard, J., and Starkenburg, E. (2016).
4most: the 4-metre Multi-Object Spectroscopic Telescope project at preliminary design review.
page 99081O, Edinburgh, United Kingdom.

De Rham, C. (1980).
La classification hiérarchique ascendante selon la méthode des voisins réciproques.
Les Cahiers de l'Analyse des Données, 135:144.

de Wilde, B., ter Mors, A. W., and Witteveen, C. (2013).
Push and rotate: cooperative multi-agent path planning.
In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 87–94. International Foundation for Autonomous Agents and Multiagent Systems.

de Zeeuw, P. T. and Molster, F. J. (2007).
A Science Vision for European Astronomy.
ASTRONET.

Defays, D. (1977).
An efficient algorithm for a complete link method.
The Computer Journal, 20(4):364–366.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977).
Maximum likelihood from incomplete data via the EM algorithm.
Journal of the royal statistical society. Series B (methodological), pages 1–38.

Desaraju, V. R. and How, J. P. (2012).
Decentralized path planning for multi-agent teams with complex constraints.
Autonomous Robots, 32(4):385–403.

Deza, M. M. and Deza, E. (2009).
Encyclopedia of distances.
In *Encyclopedia of Distances*, pages 1–583. Springer.

Dijkstra, E. (1959).
A note on two problems in connexion with graphs.
Numerische Mathematik, 1(1):269–271.

Dimarogonas, D. V. and Kyriakopoulos, K. J. (2007).
Decentralized navigation functions for multiple robotic agents with limited sensing capabilities.
Journal of Intelligent and Robotic Systems, 48(3):411–433.

Djorgovski, S., Mahabal, A., Brunner, R., Gal, R., Castro, S., De Carvalho, R., and Odewahn, S. (2000).
Searches for rare and new types of objects.
arXiv preprint astro-ph/0012453.

Dolgov, D., Thrun, S., Montemerlo, M., and Diebel, J. (2008).
Practical search techniques in path planning for autonomous driving.
Ann Arbor, 1001(48105):18–80.

Drake, S. (2003).
Galileo at work: his scientific biography.
Courier Corporation.

Dresner, K. and Stone, P. (2008).
A multiagent approach to autonomous intersection management.
Journal of artificial intelligence research, 31:591–656.

Edelstein, J., Jelinsky, P. N., Levi, M., Tarle, G., and Brooks, D. (2018).
The DESI spectrograph system and production.
In Takami, H., Evans, C. J., and Simard, L., editors, *Ground-based and Airborne Instrumentation for Astronomy VII*, page 272, Austin, United States. SPIE.

Edwards, K. J. and Gaber, M. M. (2013).
Identifying uncertain galaxy morphologies using unsupervised learning.
In *International Conference on Artificial Intelligence and Soft Computing*, pages 146–157. Springer.

Edwards, K. J. and Gaber, M. M. (2014).
Astronomy and Big Data, volume 6 of *Studies in Big Data*.
Springer International Publishing, Cham.

Edwards, M., Eikenberry, S., Marin-Franch, A., Charcos-Llorens, M., Rodgers, M., Julian, J., Raines, N., and Packham, C. (2006).
The Canarias Infrared Camera Experiment (CIRCE): optical and opto-mechanical design and manufacture.
In *Ground-based and Airborne Instrumentation for Astronomy*, volume 6269, page 62694Z. International Society for Optics and Photonics.

Eikenberry, S., Andersen, D., Guzman, R., Bally, J., Cuevas, S., Fletcher, M., Gardhouse, R., Gavel, D., Gonzalez, A., Gruel, N., and others (2006a).
IRMOS: The near-infrared multi-object spectrograph for the TMT.
In *Ground-based and Airborne Instrumentation for Astronomy*, volume 6269, page 62695W. International Society for Optics and Photonics.

Eikenberry, S., Elston, R., Raines, S. N., Julian, J., Hanna, K., Hon, D., Julian, R., Bandyopadhyay, R., Bennett, J. G., Bessoff, A., Branch, M., Corley, R., Eriksen, J.-D., Frommeyer, S., Gonzalez, A., Herlevich, M., Marin-Franch, A., Marti, J., Murphey, C., Rashkin, D., Warner, C., Leckie, B., Gardhouse, W. R., Fletcher, M., Dunn, J., Wooff, R., and Hardy, T. (2006b).
FLAMINGOS-2: the facility near-infrared wide-field imager and multi-object spectrograph for Gemini.
page 626917, Orlando, Florida , USA.

Eikenberry, S., Hinkle, K., Joyce, D., Liang, M., Muller, G., Heileman, E., French, J., Ge, J., Packham, C., Julian, R., Gaughan, N., and Sprayberry, D. (2006c).
Systems engineering and performance modeling of the Gemini High-Resolution Near-Infrared Spectrograph (HRNIRS).
In Cullum, M. J. and Angeli, G. Z., editors, *Proc. SPIE 6271, Modeling, Systems Engineering, and Project Management for Astronomy II, 62710W (June 28, 2006*, volume 62710, page 62710W, Orlando, USA. SPIE.

Eikenberry, S. S. (2013).
MIRADAS - The Next-Generation Near-Infrared Spectrograph for the GTC.
In *Revista Mexicana de Astronomia y Astrofisica Conference Series*, volume 42 of *Revista Mexicana de Astronomia y Astrofisica, vol. 27*, pages 93–95.

Eikenberry, S. S., Bennett, J. G., Chinn, B., Donoso, H. V., Eikenberry, S. A., Ettedgui, E., Fletcher, A., Frommeyer, R., Garner, A., Herlevich, M., Lasso, N., Miller, P., Mullin, S., Murphey, C., Raines, S. N., Packham, C., Schofield, S., Stelter, R. D., Varosi, F., Vega, C., Warner, C., Garzón, F., Rosich, J., Gomez, J. M., Sabater, J., Vilar, C., Torra, J., Gallego, J., Cardiel, N., Eliche, C., Pascual, S., Ballester, O., Illa, J. M., Jimenez, J., Cardiel-Sas, L., Galipienzo, J., Carrera, M. A., Hammersley, P., and Cuevas, S. (2012).
MIRADAS for the Gran Telescopio Canarias: system overview.

In McLean, I. S., Ramsay, S. K., and Takami, H., editors, *Proc. SPIE 8446, Ground-based and Airborne Instrumentation for Astronomy IV, 844657*, page 844657, Amsterdam, Holland.

Eikenberry, S. S., Elston, R., Guzman, R., Raines, S. N., Julian, J., Gruel, N., Boreman, G., Hoffmann, J., Rodgers, M., Glenn, P., and others (2006d).
FISICA: The Florida image slicer for infrared cosmology and astrophysics.
New Astronomy Reviews, 50(4-5):365–369.

Eikenberry, S. S., Murphey, C. H., Mullin, S. A., Bennett, J. G., Raines, S. N., Ackley, K., Stelter, R. D., Garner, A., Sabater, J., Eikenberry, S. A., Chinn, B., Donoso, H. V., Vega, C. S., Gómez, J. M., Torra, J., Herlevich, M. D., Frommeyer, R., and Miller, P. (2014).
Demonstration of high-performance cryogenic probe arms for deployable IFUs.
In Ramsay, S. K., McLean, I. S., and Takami, H., editors, *Proc. SPIE 9147, Ground-based and Airborne Instrumentation for Astronomy V, 91470X*, volume 9140, page 91470X, Montreal, Canada. SPIE.

Eikenberry, S. S., Raines, S. N., Stelter, R. D., Garner, A., Dallilar, Y., Ackley, K., Bennett, J. G., Murphey, C. H., Miller, P., Tooke, D., Williams, L., Chinn, B., Mullin, S. A., Schofield, S. L., Warner, C. D., Varosi, F., Zhao, B., Eikenberry, S. A., Vega, C., Donoso, H. V., Sabater, J., Gómez, J. M., Torra, J., Rosich Minguell, J., Garzón López, F., Cardiel, N., Gallego Maestro, J., Marín-Franch, A., Galipienzo, J., Carrera Astigarraga, M. A., Fitzgerald, G. J., Prees, I., Stolberg, T. M., Kornik, P. A., Ramaprakash, A. N., Burse, M. P., Punnadi, S. P., and Hammersley, P. (2016).
MIRADAS for the Gran Telescopio Canarias.
In Evans, C. J., Simard, L., and Takami, H., editors, *Proc. SPIE 9908, Ground-based and Airborne Instrumentation for Astronomy VI, 99081L (August 4, 2016)*, page 99081L, Edinburgh, United Kingdom.

Ellis, R. S., Bland-Hawthorn, J., Bremer, M., Brinchmann, J., Guzzo, L., Richard, J., Rix, H.-W., Tolstoy, E., and Watson, D. (2017).
The Future of Multi-Object Spectroscopy: a ESO Working Group Report.
arXiv preprint arXiv:1701.01976.

Erdmann, M. and Lozano-Pérez, T. (1986).
On multiple moving objects.
volume 3, pages 1419–1424. Institute of Electrical and Electronics Engineers.

Erdmann, M. and Lozano-Pérez, T. (1987).
On Multiple Moving Objects.
Algorithmica, 2(4):477–521.

Everitt, B. S., Landau, S., Leese, M., and Stahl, D. (2011).

Cluster Analysis.
Wiley Series in Probability and Statistics. John Wiley & Sons, Ltd, Chichester, UK.

Eversberg, T. and Vollmann, K. (2015).
Spectroscopic instrumentation: Fundamentals and Guidelines for Astronomers.
Springer.

Faber, S. M., Phillips, A. C., Kibrick, R. I., Alcott, B., Allen, S. L., Burrous, J., Cantrall, T., Clarke, D., Coil, A. L., Cowley, D. J., Davis, M., Deich, W. T. S., Dietsch, K., Gilmore, D. K., Harper, C. A., Hilyard, D. F., Lewis, J. P., McVeigh, M., Newman, J., Osborne, J., Schiavon, R., Stover, R. J., Tucker, D., Wallace, V., Wei, M., Wirth, G., and Wright, C. A. (2003).
The DEIMOS spectrograph for the Keck II Telescope: integration and testing.
page 1657, Waikoloa, Hawai'i, United States.

Fabricant, D. G., Hertz, E. N., and Szentgyorgyi, A. H. (1994).
Hectospec: a 300-optical-fiber spectrograph for the converted MMT.
pages 251–263, Kailua, Kona, HI.

Ferguson, D., Kalra, N., and Stentz, A. (2006).
Replanning with rrts.
In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1243–1248. IEEE.

Ferrari, C., Pagello, E., Ota, J., and Arai, T. (1998).
Multirobot motion coordination in space and time.
Robotics and Autonomous Systems, 25(3-4):219–229.

Fisher, C., Morantz, C., Braun, D., Seiffert, M., Aghazarian, H., Partos, E., King, M., Hovland, L. E., Schwochert, M., Kaluzny, J., Capocasale, C., Houck, A., Gross, J., Reiley, D., Mao, P., Riddle, R., Bui, K., Henderson, D., Haran, T., Culhane, R., Piazza, D., and Walkama, E. (2014).
Developing engineering model Cobra fiber positioners for the Subaru Telescope's prime focus spectrometer.
page 91511Y, Montréal, Quebec, Canada.

Fisher, M. L., Jaikumar, R., and Wassenhove, L. N. V. (1986).
A Multiplier Adjustment Method for the Generalized Assignment Problem.
Management Science, 32(9):1095–1103.

Flaugher, B. and Bebek, C. (2014).
The Dark Energy Spectroscopic Instrument (DESI).
page 91470S, Montréal, Quebec, Canada.

Fraichard, T. (1998).
Trajectory planning in a dynamic workspace: a'state-time space'approach.
Advanced Robotics, 13(1):75–94.

Fraley, C. and Raftery, A. E. (2002).
Model-based clustering, discriminant analysis, and density estimation.
Journal of the American statistical Association, 97(458):611–631.

Freeman, K. and Bland-Hawthorn, J. (2002).
The New Galaxy: Signatures of its Formation.
Annual Review of Astronomy and Astrophysics, 40(1):487–537.

Gale, D. and Shapley, L. S. (1962).
College admissions and the stability of marriage.
The American Mathematical Monthly, 69(1):9–15.

Garner, A., Eikenberry, S. S., Charcos, M., Dallilar, Y., Edwards, M., Lasso-Cabrera, N., Stelter, R. D., Marin-Franch, A., Raines, S. N., Ackley, K., and others (2016).
First results and future plans for the Canarias Infrared Camera Experiment (CIRCE) for the Gran Telescopio Canarias.
In *Ground-based and Airborne Instrumentation for Astronomy VI*, volume 9908, page 99084Q. International Society for Optics and Photonics.

Garzón, F., Abreu, D., Barrera, S., Becerril, S., Cairós, L. M., Díaz, J. J., Fragoso, A. B., Gago, F., Grange, R., González, C., López, P., Patrón, J., Pérez, J., Rasilla, J. L., Redondo, P., Restrepo, R., Saavedra, P., Sánchez, V., Tenegi, F., and Vallbé, M. (2006).
EMIR: the GTC NIR multi-object imager-spectrograph.
In McLean, I. S. and Iye, M., editors, *Proc. SPIE 6269, Ground-based and Airborne Instrumentation for Astronomy, 626918 (June 28, 2006)*, volume 6269, pages 626918–626918–10, Orlando, USA. SPIE.

Garzón, F., Castro-Rodríguez, N., Insausti, M., López-Martín, L., Hammersley, P., Barreto, M., Fernández, P., Joven, E., López, P., Mato, A., Moreno, H., Núñez, M., Patrón, J., Rasilla, J. L., Redondo, P., Rosich, J., Pascual, S., and Grange, R. (2014).
Results of the verification of the NIR MOS EMIR.
In Ramsay, S. K., McLean, I. S., and Takami, H., editors, *Proc. SPIE 9147, Ground-based and Airborne Instrumentation for Astronomy V, 91470U (July 8, 2014*, page 91470U, Montreal, Canada. SPIE.

Ge, J., Lee, B., De Lee, N., Wan, X., Groot, J., Zhao, B., Varosi, F., Hanna, K., Mahadevan, S., Hearty, F., Chang, L., Liu, J., van Eyken, J., Wang, J., Pais, R., Chen, Z., Shelden, A., and Costello, E. (2009).

A new generation multi-object Doppler instrument for the SDSS-III Multi-object APO Radial Velocity Exoplanet Large-area Survey.
page 74400L, San Diego, CA.

Ge, S. and Cui, Y. (2002).
Dynamic Motion Planning for Mobile Robots Using Potential Field Method.
Autonomous Robots, 13(3):207–222.

Gendreau, M. and Potvin, J.-Y., editors (2010).
Handbook of Metaheuristics, volume 146 of *International Series in Operations Research & Management Science*.
Springer US, Boston, MA.

Geraerts, R. and Overmars, M. (2004).
A Comparative Study of Probabilistic Roadmap Planners.
In Boissonnat, J.-D., Burdick, J., Goldberg, K., and Hutchinson, S., editors, *Algorithmic Foundations of Robotics V*, volume 7 of *Springer Tracts in Advanced Robotics*, pages 43–57. Springer Berlin Heidelberg.

Geraerts, R. and Overmars, M. H. (2005).
Reachability Analysis of Sampling Based Planners.
In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 404–410.

Geramifard, A., Chubak, P., and Bulitko, V. (2006).
Biased Cost Pathfinding.
In *Proceedings of the Second Artificial Intelligence and Interactive Digital Entertainment Conference, June 20-23, 2006*, pages 112–114, Marina del Rey, California. AIIDE.

Gilbert, J., Goodwin, M., Heijmans, J., Muller, R., Miziarski, S., Brzeski, J., Waller, L., Saunders, W., Bennet, A., and Tims, J. (2012).
Starbugs: all-singing, all-dancing fibre positioning robots.
In Navarro, R., Cunningham, C. R., and Prieto, E., editors, *Proc. SPIE 8450, Modern Technologies in Space- and Ground-based Telescopes and Instrumentation II, 84501A (September 13, 2012)*, volume 8450, pages 84501A–84501A–14. SPIE.

Goldberg, D. E. and Deb, K. (1991).
A comparative analysis of selection schemes used in genetic algorithms.
In *Foundations of genetic algorithms*, volume 1, pages 69–93. Elsevier.

Goodwin, M., Lorente, N. P. F., Satorre, C., Hong, S. E., Kuehn, K., and Lawrence, J. S. (2014).
Field target allocation and routing algorithms for Starbugs.

In Chiozzi, G. and Radziwill, N. M., editors, *Proc. SPIE 9152, Software and Cyberinfrastructure for Astronomy III, 91520P (July 18, 2014)*, page 91520S, Montreal, Canada. SPIE.

Green, R. M. and Green, R. M. (1985).
Spherical astronomy.
Cambridge University Press.

Harrison, G. R. (1949).
The Production of Diffraction Gratings: II. The Design of Echelle Gratings and Spectrographs1.
J. Opt. Soc. Am., 39(7):522–528.

Hart, J. and Shogan, A. W. (1987).
Semi-greedy Heuristics: An Empirical Study.
Oper. Res. Lett., 6(3):107–114.

Hart, P., Nilsson, N., and Raphael, B. (1968).
A Formal Basis for the Heuristic Determination of Minimum Cost Paths.
Systems Science and Cybernetics, IEEE Transactions on, 4(2):100–107.

Hartenberg, R. and Danavit, J. (1964).
Kinematic synthesis of linkages.
New York: McGraw-Hill.

Hearnshaw, J. (2009).
Astronomical Spectrographs and their History.
Cambridge University Press, Cambridge.

Hearnshaw, J. B. (2014).
The Analysis of Starlight: Two Centuries of Astronomical Spectroscopy.
Cambridge University Press.
Google-Books-ID: waf6AgAAQBAJ.

Hill, G. J., Tuttle, S. E., Vattiat, B. L., Lee, H., Drory, N., Kelz, A., Ramsey, J., Peterson, T. W., DePoy, D. L., Marshall, J. L., Gebhardt, K., Chonis, T., Dalton, G., Farrow, D., Good, J. M., Haynes, D. M., Indahl, B. L., Jahn, T., Kriel, H., Montesano, F., Nicklas, H., Noyola, E., Prochaska, T., Allen, R. D., Bender, R., Blanc, G., Fabricius, M. H., Finkelstein, S., Landriau, M., MacQueen, P. J., Roth, M. M., Savage, R. D., Snigula, J. M., and Anwad, H. (2016).
VIRUS: first deployment of the massively replicated fiber integral field spectrograph for the upgraded Hobby-Eberly Telescope.
page 99081H, Edinburgh, United Kingdom.

Hill, J. M., Angel, J., Scott, J. S., Lindley, D., and Hintzen, P. (1980).
Multiple object spectroscopy-The Medusa spectrograph.

The Astrophysical Journal, 242:L69–L72.

Hsu, D., Kindel, R., Latombe, J.-C., and Rock, S. (2002).
Randomized Kinodynamic Motion Planning with Moving Obstacles.
The International Journal of Robotics Research, 21(3):233–255.

Hsu, D., Latombe, J.-C., and Kurniawati, H. (2006).
On the Probabilistic Foundations of Probabilistic Roadmap Planning.
Int. J. Rob. Res., 25(7):627–643.

Hwang, Y. K. and Ahuja, N. (1992).
A potential field approach to path planning.
IEEE Transactions on Robotics and Automation, 8(1):23–32.

Ichikawa, T., Suzuki, R., Tokoku, C., Uchimoto, Y. K., Konishi, M., Yoshikawa, T., Yamada, T., Tanaka, I., Omata, K., and Nishimura, T. (2006).
MOIRCS: multi-object infrared camera and spectrograph for SUBARU.
page 626916, Orlando, Florida , USA.

INAF, F. G. G. (2019).
TNG :: DOLORES (Device Optimized for the LOw RESolution).

Isto, P. (2002).
Constructing probabilistic roadmaps with powerful local planning and path optimization.
In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2323–2328 vol.3.

Jeon, J. W. and Ha, Y. Y. (2000).
A generalized approach for the acceleration and deceleration of industrial robots and CNC machine tools.
IEEE transactions on industrial electronics, 47(1):133–139.

Jung, Y., Park, H., Du, D.-Z., and Drake, B. L. (2003).
A decision criterion for the optimal number of clusters in hierarchical clustering.
Journal of Global Optimization, 25(1):91–111.

Kant, K. and Zucker, S. W. (1986).
Toward Efficient Trajectory Planning: The Path-Velocity Decomposition.
The International Journal of Robotics Research, 5(3):72–89.

Kashikawa, N., Inata, M., Iye, M., Kawabata, K. S., Okita, K., Kosugi, G., Ohyama, Y., Sasaki, T., Sekiguchi, K., Takata, T., Shimizu, Y., Yoshida, M., Aoki, K., Saito, Y., Asai, R., Taguchi, H., Ebizuka, N., Ozawa, T., and Yadoumaru, Y. (2000).

FOCAS: faint object camera and spectrograph for the Subaru Telescope.
page 104, Munich, Germany.

Kavraki, L. (1995).
Computation of configuration-space obstacles using the fast Fourier transform.
Robotics and Automation, IEEE Transactions on, 11(3):408–413.

Kavraki, L. E., Švestka, P., Latombe, J.-C., and Overmars, M. H. (1996).
Probabilistic roadmaps for path planning in high-dimensional configuration spaces.
Robotics and Automation, IEEE Transactions on, 12(4):566–580.

Ketchen Jr., D. J. and Shook, C. L. (1996).
The Application of Cluster Analysis in Strategic Management Research: an Analysis and Critique.
Strategic Management Journal, 17(6):441–458.

Khatib, O. (1985).
Real-time obstacle avoidance for manipulators and mobile robots.
volume 2, pages 500–505. Institute of Electrical and Electronics Engineers.

Khorshid, M. M., Holte, R. C., and Sturtevant, N. R. (2011).
A polynomial-time algorithm for non-optimal multi-agent pathfinding.
In *Fourth Annual Symposium on Combinatorial Search*.

Kohley, R. (2000).
GTC - Present status and future plans.
In Amico, P. and Beletic, J. W., editors, *Optical Detectors For Astronomy II: State-of-the-Art at the Turn of the Millennium*, pages 51–53. Springer Netherlands, Dordrecht.

Koren, Y. and Borenstein, J. (1991).
Potential field methods and their inherent limitations for mobile robot navigation.
In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 1398–1404 vol.2.

Korf, R. E. (1985).
Depth-first iterative-deepening: An optimal admissible tree search.
Artificial intelligence, 27(1):97–109.

Korte, B. and Vygen, J. (2012).
Combinatorial Optimization, volume 21 of *Algorithms and Combinatorics*.
Springer Berlin Heidelberg, Berlin, Heidelberg.

Kremer, J., Stensbo-Smidt, K., Gieseke, F., Pedersen, K. S., and Igel, C. (2017).

Big universe, big data: machine learning and image analysis for astronomy.
arXiv preprint arXiv:1704.04650.

Krontiris, A., Luna, R., and Bekris, K. E. (2013).
From feasibility tests to path planners for multi-agent pathfinding.
In *Sixth annual symposium on combinatorial search*.

Kuehn, K., Lawrence, J., Brown, D. M., Case, S., Colless, M., Content, R., Gers, L., Gilbert, J., Goodwin, M., Hopkins, A. M., and others (2014).
TAIPAN: optical spectroscopy with StarBugs.
In *Ground-based and Airborne Instrumentation for Astronomy V*, volume 9147, page 914710.
International Society for Optics and Photonics.

Kuhn, H. W. (1955).
The Hungarian method for the assignment problem.
Naval research logistics quarterly, 2(1-2):83–97.

Land, A. H. and Doig, A. G. (1960).
An Automatic Method of Solving Discrete Programming Problems.
Econometrica, 28(3):497.

Larkin, J. E., Moore, A. M., Wright, S. A., Wincentsen, J. E., Anderson, D., Chisholm, E. M., Dekany, R. G., Dunn, J. S., Ellerbroek, B. L., Hayano, Y., Phillips, A. C., Simard, L., Smith, R., Suzuki, R., Weber, R. W., Weiss, J. L., and Zhang, K. (2016).
The Infrared Imaging Spectrograph (IRIS) for TMT: instrument overview.
page 99081W, Edinburgh, United Kingdom.

Latombe, J.-C. (1991).
Robot Motion Planning.
Springer US, Boston, MA.

Lau, B., Sprunk, C., and Burgard, W. (2013).
Efficient grid-based spatial representations for robot navigation in dynamic environments.
Robotics and Autonomous Systems, 61(10):1116–1130.

LaValle, S. and Hutchinson, S. (1998).
Optimal motion planning for multiple robots having independent goals.
IEEE Transactions on Robotics and Automation, 14(6):912–925.

LaValle, S. M. (2006).
Planning algorithms.
Cambridge University Press, Cambridge; New York.

LaValle, S. M., Branicky, M. S., and Lindemann, S. R. (2004).
On the Relationship between Classical Grid Search and Probabilistic Roadmaps.
The International Journal of Robotics Research, 23(7-8):673–692.

LaValle, S. M. and Kuffner, J. J. (2001).
Randomized Kinodynamic Planning.
The International Journal of Robotics Research, 20(5):378–400.

LaValle, S. M., Kuffner, J. J., and Jr. (2000).
Rapidly-Exploring Random Trees: Progress and Prospects.
In *Algorithmic and Computational Robotics: New Directions*, pages 293–308, Wellesley, MA.

Lawler, E. L., Lenstra, J. K., Kan, A. R., Shmoys, D. B., and others (1985).
The traveling salesman problem: a guided tour of combinatorial optimization, volume 3.
Wiley New York.

Lawrence, J., Ben-Ami, S., Brown, D. M., Brown, R., Case, S., Chin, T., Colless, M., Contos, A., Depoy, D., Evans, I., and others (2018).
Wide-field multi-object spectroscopy with MANIFEST.
In *Ground-based and Airborne Instrumentation for Astronomy VII*, volume 10702, page 10702A6. International Society for Optics and Photonics.

Lawrence, J. S., Brown, D. M., Brzeski, J., Case, S., Colless, M., Farrell, T., Gers, L., Gilbert, J., Goodwin, M., Jacoby, G., Hopkins, A. M., Ireland, M., Kuehn, K., Lorente, N. P. F., Miziarski, S., Muller, R., Nichani, V., Rakman, A., Richards, S., Saunders, W., Staszak, N. F., Tims, J., Vuong, M., and Waller, L. (2014a).
The MANIFEST fibre positioning system for the Giant Magellan Telescope.
In Ramsay, S. K., McLean, I. S., and Takami, H., editors, *Proc. SPIE 9147, Ground-based and Airborne Instrumentation for Astronomy V, 914794 (July 8, 2014)*, page 914794, Montreal, Canada. SPIE.

Lawrence, J. S., Brown, D. M., Brzeski, J., Case, S., Colless, M., Farrell, T., Gers, L., Gilbert, J., Goodwin, M., Jacoby, G., and others (2014b).
The MANIFEST fibre positioning system for the Giant Magellan Telescope.
In *Ground-based and Airborne Instrumentation for Astronomy V*, volume 9147, page 914794.
International Society for Optics and Photonics.

Lee, B. and Lee, C. (1987).
Collision-Free Motion Planning of Two Robots.
Systems, Man and Cybernetics, IEEE Transactions on, 17(1):21–32.

Lee, J. (2011).

Introduction to Topological Manifolds, volume 202 of *Graduate Texts in Mathematics*. Springer Science & Business Media.

LeFevre, O., Crampton, D., Flenbok, P., and Monnet, G. (1994). CFHT MOS/SIS spectrograph performance. *Astronomy and Astrophysics*, (282):325–340.

LeFevre, O., Saisse, M., Mancini, D., Brau-Nogue, S., Caputi, O., Castinel, L., D'Odorico, S., Garilli, B., Kissler-Patig, M., Lucuix, C., Mancini, G., Pauget, A., Sciarretta, G., Scodeggio, M., Tresse, L., and Vettolani, G. (2003). Commissioning and performances of the VLT-VIMOS. In Iye, M. and Moorwood, A. F. M., editors, *Proc. SPIE 4841, Instrument Design and Performance for Optical/Infrared Ground-based Telescopes, 1670 (March 7, 2003)*, volume 4841, pages 1670–1681, Waikoloa, Hawai'i, United States.

LeFevre, O., Saisse, M., Mancini, D., Vettolani, G., Maccagni, D., Picat, J. P., Mellier, Y., Mazure, A., Cuby, J. G., Delabre, B., Garilli, B., Hill, L., Prieto, E., Voet, C., Arnold, L., Brau-Nogue, S., Cascone, E., Conconi, P., Finger, G., Huster, G., Laloge, A., Lucuix, C., Mattaini, E., Schipani, P., Waultier, G., Zerbi, F. M., Avila, G., Beletic, J. W., D'Odorico, S., Moorwood, A. F. M., Monnet, G. J., and Reyes Moreno, J. (2000). VIMOS and NIRMOS multi-object spectrographs for the ESO VLT. page 546, Munich, Germany.

Leibe, B., Leonardis, A., and Schiele, B. (2008). Robust Object Detection with Interleaved Categorization and Segmentation. *International Journal of Computer Vision*, 77(1):259–289.

Leroy, S., Laumond, J. P., and Simeon, T. (1999). Multiple Path Coordination for Mobile Robots: A Geometric Algorithm. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, pages 1118–1123, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Lewis, I. J., Cannon, R., Taylor, K., Glazebrook, K., Bailey, J., Baldry, I., Barton, J., Bridges, T., Dalton, G., Farrell, T., and others (2002). The Anglo-Australian Observatory 2df facility. *Monthly Notices of the Royal Astronomical Society*, 333(2):279–298.

Lewis, I. J., Sharples, R. M., Parry, I. R., Jones, L. R., Watson, F. G., Barker, S. A., and Rees, P. C. T. (1997). Autofib-2: commissioning results of a robotic multiobject fiber system for the William Herschel Telescope. pages 1318–1324, Landskrona/Hven, Sweden.

Likhachev, M., Ferguson, D. I., Gordon, G. J., Stentz, A., and Thrun, S. (2005).
Anytime Dynamic A*: An Anytime, Replanning Algorithm.
In *ICAPS*, pages 262–271.

Likhachev, M., Gordon, G. J., and Thrun, S. (2003).
ARA*: Anytime A* with provable bounds on sub-optimality.
In *NIPS*, pages 767–774.

Lindegren, L. (2005).
Cross-matching Gaia objects -Technical Note GAIA-LL-060.
Technical report, Gaia Data Processing and Analysis Consortium (DPAC).

Lindegren, L., Perryman, M. A., Bastian, U., Dainty, C., Hog, E., van Leeuwen, F., Kovalevsky, J., Labeyrie, A., Loiseau, S., Mignard, F., Noordam, J. C., le Poole, R. S., Thejll, P., and Vakili, F. (1994).
GAIA: global astrometric interferometer for astrophysics.
pages 599–608, Kailua, Kona, HI.

Lindemann, S. and LaValle, S. (2005).
Current Issues in Sampling-Based Motion Planning.
In Dario, P. and Chatila, R., editors, *Robotics Research. The Eleventh International Symposium*, volume 15 of *Springer Tracts in Advanced Robotics*, pages 36–54. Springer Berlin Heidelberg.

Little, J. D. C., Murty, K. G., Sweeney, D. W., and Karel, C. (1963).
An Algorithm for the Traveling Salesman Problem.
Operations Research, 11(6):972–989.

Lopez-Sastre, R. J., Onoro-Rubio, D., Gil-Jimenez, P., and Maldonado-Bascon, S. (2012).
Fast Reciprocal Nearest Neighbors Clustering.
Signal Processing, 92:270–275.

Lorente, N. P., Vuong, M. V., Shortridge, K., Farrell, T. J., Smedley, S., Hong, S. E., Bacigalupo, C., Goodwin, M., Kuehn, K., and Satorre, C. (2016).
AAO Starbugs: software control and associated algorithms.
In *Software and Cyberinfrastructure for Astronomy IV*, volume 9913, page 99130U. International Society for Optics and Photonics.

Lorente, N. P. F., Kuehn, K., Lawrence, J. S., Bacigalupo, C., Brown, D., Brown, R., Case, S., Chapman, S., Churilov, V., Content, R., Farrell, T., Goodwin, M., Klauser, U., Mali, S., Muller, R., Nichani, V., Pai, N., Smedley, S., Vuong, M., Waller, L., Zhelem, R., and McGregor, H. (2018).
TAIPAN: the AAO's first Starbug positioner and spectrograph.
In Takami, H., Evans, C. J., and Simard, L., editors, *Ground-based and Airborne Instrumentation for Astronomy VII*, page 46, Austin, United States. SPIE.

Lozano-Pérez, T. (1983).
Spatial Planning: A Configuration Space Approach.
IEEE Transactions on Computers, C-32(2):108–120.

Lozano-Pérez, T. and Wesley, M. A. (1979).
An Algorithm for Planning Collision-free Paths Among Polyhedral Obstacles.
Commun. ACM, 22(10):560–570.

Luna, R. and Bekris, K. E. (2011a).
Efficient and complete centralized multi-robot path planning.
pages 3268–3275. IEEE.

Luna, R. and Bekris, K. E. (2011b).
Push and swap: Fast cooperative path-finding with completeness guarantees.
In *IJCAI*, pages 294–300.

MacQueen, J. and others (1967).
Some methods for classification and analysis of multivariate observations.
In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Maihara, T., Ohta, K., Tamura, N., Ohtani, H., Akiyama, M., Noumaru, J., Kaifu, N., Karoji, H., Iye, M., Dalton, G. B., Parry, I. R., Robertson, D. J., Sharples, R. M., Ren, D., Allington-Smith, J. R., Taylor, K., and Gillingham, P. R. (2000).
Fiber multi-object spectrograph (FMOS) for the Subaru Telescope.
page 1111, Munich, Germany.

Makarem, L., Kneib, J., Gillet, D., Bleuler, H., Bouri, M., Jenni, L., Prada, F., and Sanchez, J. (2014a).
Collision avoidance in next-generation fiber positioner robotic system for large survey spectrograph.
Astronomy & Astrophysics, 556.

Makarem, L., Kneib, J.-P., and Gillet, D. (2016).
Collision-free coordination of fiber positioners in multi-object spectrographs.
In *SPIE Astronomical Telescopes+ Instrumentation*, pages 99130V–99130V. International Society for Optics and Photonics.

Makarem, L., Kneib, J.-P., Gillet, D., Bleuler, H., Bouri, M., Hörler, P., Jenni, L., Prada, F., and Sánchez, J. (2014b).
Collision-free motion planning for fiber positioner robots: discretization of velocity profiles.
In Chiozzi, G. and Radziwill, N. M., editors, *Proc. SPIE 9152, Software and Cyberinfrastructure for Astronomy III, 91520P (July 18, 2014)*, page 91520Q, Montreal, Canada. SPIE.

Manchado-Torres, A., Fuentes, F. J., Prada, F., Ballesteros Ramirez, E., Barreto, M., Carranza, J. M., Escudero, I., Fragoso-Lopez, A. B., Joven-Alvarez, E., Manescau, A., Pi i Puig, M., Rodriguez-Ramos, L. F., and Sosa, N. A. (1998).
LIRIS: a long-slit intermediate-resolution infrared spectrograph for the WHT.
page 448, Kona, HI.

Marshack, A. (1964).
Lunar notation on Upper Paleolithic remains.
Science, 146(3645):743–745.

Marshall, J. L., Burles, S., Thompson, I. B., Shectman, S. A., Bigelow, B. C., Burley, G., Birk, C., Estrada, J., Jones, P., Smith, M., Kowal, V., Castillo, J., Storts, R., and Ortiz, G. (2008).
The MagE spectrograph.
page 701454, Marseille, France.

Massey, P. and Hanson, M. M. (2013).
Astronomical Spectroscopy.
Planets, Stars and Stellar Systems: Volume 2: Astronomical Techniques, Software, and Data, pages 35–98.

Matousek, J. and Gärtner, B. (2007).
Understanding and Using Linear Programming.
Universitext. Springer Berlin Heidelberg, Berlin, Heidelberg.

McCarthy, J. K., Cohen, J. G., Butcher, B., Cromer, J., Croner, E., Douglas, Jr., W. R., Goeden, R. M., Grewal, T., Lu, B., Petrie, H. L., Weng, T., Weber, B., Koch, D. G., and Rodgers, J. M. (1998).
Blue channel of the Keck low-resolution imaging spectrometer.
pages 81–92, Kona, HI.

McCarthy, P. J., Ashby, D. S., Bigelow, B. C., Bouchez, A. H., Burgett, W. S., Chauvin, E., Contos, A., Figueroa, F., Gray, P., Groark, F., Laskin, R., Millan-Gabet, R., Rakich, A., Sandoval, R., Fanson, J. L., Bernstein, R., Angeli, G. Z., Pi, M., and Wheeler, N. (2018).
Overview and status of the Giant Magellan Telescope project.
In Gilmozzi, R., Marshall, H. K., and Spyromilio, J., editors, *Ground-based and Airborne Telescopes VII*, page 34, Austin, United States. SPIE.

McLachlan, G. and Krishnan, T. (2007).
The EM algorithm and extensions, volume 382.
John Wiley & Sons.

McLachlan, G. and Peel, D. (2004).

Finite mixture models.
John Wiley & Sons.

McLean, I. S., Steidel, C. C., Epps, H., Matthews, K., Adkins, S., Konidaris, N., Weber, B., Aliado, T., Brims, G., Canfield, J., Cromer, J., Fucik, J., Kulas, K., Mace, G., Magnone, K., Rodriguez, H., Wang, E., and Weiss, J. (2010).
Design and development of MOSFIRE: the multi-object spectrometer for infrared exploration at the Keck Observatory.
page 77351E, San Diego, California, USA.

Miszalski, B., Shortridge, K., Saunders, W., Parker, Q. A., and Croom, S. M. (2006).
Multi-object spectroscopy field configuration by simulated annealing.
Monthly Notices of the Royal Astronomical Society, 371(4):1537–1549.

Morales, I., Montero-Dorta, A. D., Azzaro, M., Prada, F., Sánchez, J., and Becerril, S. (2012).
Fibre assignment in next-generation wide-field spectrographs.
Monthly Notices of the Royal Astronomical Society, 419(2):1187–1196.

Morales, N., Arnay, R., Toledo, J., Morell, A., and Acosta, L. (2016).
Safe and reliable navigation in crowded unstructured pedestrian areas.
Engineering Applications of Artificial Intelligence, 49:74 – 87.

Moravec, H. P. and Elfes, A. (1985).
High resolution maps from wide angle sonar.
In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 116–121. IEEE.

Munkres, J. (1957).
Algorithms for the Assignment and Transportation Problems.
Journal of the Society for Industrial and Applied Mathematics, 5(1):32–38.

Murtagh, F. (1983).
A Survey of Recent Advances in Hierarchical Clustering Algorithms.
The Computer Journal, 26(4):354–359.

Murtagh, F. (1984).
Complexities of hierarchic clustering algorithms: State of the art.
Computational Statistics Quarterly, 1(2):101–113.

Murtagh, F. and Contreras, P. (2012).
Algorithms for hierarchical clustering: an overview.
Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2(1):86–97.

Murtagh, F. and Legendre, P. (2011).
Ward's hierarchical clustering method: clustering criterion and agglomerative algorithm.
arXiv preprint arXiv:1111.6285.

Márquez, I., Vega, O., Eliche, C., Roca-Fàbrega, S., Carrasco Licea, E. E., Gil de Paz, A., Gallego, J., Iglesias-Páramo, J., Cedazo, R., García-Vargas, M. L., Arrillaga, X., Avilés, J. L., Bouquin, A., Carbajo, J., Cardiel, N., Carrera, M.-A., Castillo-Morales, A., Castillo-Domínguez, E., López Orozco, J. A., Ferrusca, D., Gómez-Álvarez, P., Izazaga-Pérez, R., Lefort, B., Maldonado, M., Martínez Delgado, I., Morales Durán, I., Mújica, E., Páez, G., Pascual, S., Pérez-Calpena, A., Picazo, P., Sánchez-Penim, A., Sánchez-Blanco, E., Tulloch, S., Velázquez, M., Vílchez, J. M., Zamorano, J., Aguerri, J. A., Barrado y Navascues, D., Bertone, E., Cava, A., Catalán-Torrecilla, C., Cenarro, A. J., Chávez, M., Dullo, B. T., García, M., García-Rojas, J., Guichard, J., González-Delgado, R., Guzmán, R., Herrero, A., Huélamo, N., Hughes, D. H., Jiménez-Vicente, J., Kehrig, C., Marino, R. A., Masegosa, J., Mayya, D., Méndez-Abreu, J., Mollá, M., Muñoz-Tuñón, C., Peimbert, M., Pérez-González, P. G., Pérez-Montero, E., Rodríguez, M., Rodríguez-Espinosa, J. M., Rodríguez-Merino, L., Rodríguez-Muñoz, L., Rosa-González, D., Sánchez-Almeida, J., Sánchez Contreras, C., Sánchez-Blázquez, P., Sánchez, S. F., Sarajedini, A., Silich, S., Simón-Díaz, S., Tenorio-Tagle, G., Terlevich, E., Terlevich, R., Torres-Peimbert, S., Trujillo, I., Tsamis, Y., Ortiz, R., and Esteban San Román, S. (2018).
MEGARA, the R=6000-20000 IFU and MOS of GTC.
In Takami, H., Evans, C. J., and Simard, L., editors, *Ground-based and Airborne Instrumentation for Astronomy VII*, page 42, Austin, United States. SPIE.

Neumann, W. (2014).
Fundamentals of dispersive optical spectroscopy systems.
SPIE Press.

Nieuwenhuisen, D. and Overmars, M. H. (2004).
Useful cycles in probabilistic roadmap graphs.
In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 446–452. IEEE.

Nilsson, N. J. (1969).
A Mobile Automaton: An application of artificial intelligence techniques.
Proc. 1st Int. Joint Conf. on Artificial Intelligence, pages 509–520.

O'Donnell, P. and Lozano-Pérez, T. (1989).
Deadlock-free and collision-free coordination of two robot manipulators.
In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 484–489 vol.1.

Oke, J., Cohen, J., Carr, M., Cromer, J., Dingizian, A., Harris, F., Labrecque, S., Lucinio, R., Schaal, W., Epps, H., and others (1995).
The Keck low-resolution imaging spectrometer.
Publications of the Astronomical Society of the Pacific, 107(710):375.

Owen, R. E., Siegmund, W. A., Limmongkol, S., and Hull, C. L. (1994).
Fiber feed for the SDSS spectrograph.
pages 110–114, Kailua, Kona, HI.

Pandey, H. M. (2016).
Performance evaluation of selection methods of genetic algorithm and network security concerns.
Procedia Computer Science, 78:13–18.

Peasgood, M., Clark, C. M., and McPhee, J. (2008).
A complete and scalable strategy for coordinating multiple robots within roadmaps.
Robotics, IEEE Transactions on, 24(2):283–292.

Peng, J. and Akella, S. (2005).
Coordinating multiple robots with kinodynamic constraints along specified paths.
The International Journal of Robotics Research, 24(4):295–310.

Phillips, J., Bedrossian, N., and Kavraki, L. (2004).
Guided Expansive Spaces Trees: a search strategy for motion- and cost-constrained state spaces.
In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, pages 3968–3973 Vol.4, New Orleans, LA, USA. IEEE.

Phytron (2010).
Phytron MCC2, Manual 1235-A012 GB.

Phytron (2012).
MiniLog Programming for MCC, Manual 1240-A006 EN.

Pickering, E. C. (1912).
The Objective Prism.
Proceedings of the American Philosophical Society, 51(207):564–567.

Pogge, R. W., Atwood, B., Brewer, D. F., Byard, P. L., Derwent, M. A., Gonzalez, R., Martini, P., Mason, J. A., O'Brien, T. P., Osmer, P. S., Pappalardo, D. P., Steinbrecher, D. P., Teiga, E. J., and Zhelem, R. (2010).
The multi-object double spectrographs for the Large Binocular Telescope.
page 77350A, San Diego, California, USA.

Pérez-Calpena, A., García-Vargas, M. L., Arrillaga, X., Gil de Paz, A., Sánchez-Blanco, E., Martínez-Delgado, I., Carrera, M. A., Gallego, J., Carrasco, E., Sánchez-Moreno, F. M., and Iglesias-Páramo, J. (2014).
MEGARA fiber bundles.
page 91475N, Montréal, Quebec, Canada.

Raines, S. N., Stelter, R. D., Ackley, K., Bennett, J. G., Chinn, B., Mullin, S., Schofield, S., Warner, C., Varosi, F., Zhao, B., Eikenberry, S. A., Vega, C., Donoso, V. H., Carrera, C., Almeida, D., Fuentes, G., Sabater, J., Gomez, J. M., Garzón López, F., Lopez, P., Rosich, J. ., Russo, A., Marin-Franch, A., Larman, A., Carrera, M.-A., Fitzgerald, G. J., Prees, I., Stolberg, T. M., Kornik, P., Ramaprakash, A. N., Burse, M., Punnadi, S., Hammersley, P., Eikenberry, S. S., Garner, A., and Dallilar, Y. (2018).
MIRADAS: the facility multi-object medium-resolution NIR spectrograph for the GTC.
In Takami, H., Evans, C. J., and Simard, L., editors, *Ground-based and Airborne Instrumentation for Astronomy VII*, page 50, Austin, United States. SPIE.

Raskin, G. and Van Winckel, H. (2008).
HERMES: a high-resolution fiber-fed spectrograph for the Mercator Telescope.
page 70145D, Marseille, France.

Regele, R. and Levi, P. (2006).
Cooperative Multi-Robot Path Planning by Heuristic Priority Adjustment.
In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5954–5959.

Reif, J. H. (1979).
Complexity of the mover's problem and generalizations.
20th Annual IEEE Symposium on Foundations of Computer Science, 0:421–427.

Resende, M. G. and Ribeiro, C. C. (2010).
Greedy randomized adaptive search procedures: Advances, hybridizations, and applications.
In *Handbook of metaheuristics*, pages 283–319. Springer.

Resende, M. G. and Ribeiro, C. C. (2016).
Optimization by GRASP.
Springer New York, New York, NY.

Resende, M. G. C. and Ribeiro, C. C. (2003).
Greedy Randomized Adaptive Search Procedures.
In *Handbook of Metaheuristics*, International Series in Operations Research & Management Science, pages 219–249. Springer US, Boston, MA.

Riera-Ledesma, J. and Salazar-González, J. J. (2005).
The biobjective travelling purchaser problem.
European Journal of Operational Research, 160(3):599–613.

Rimon, E. and Koditschek, D. E. (1992).
Exact robot navigation using artificial potential functions.
IEEE Transactions on Robotics and Automation, 8(5):501–518.

Robinson, L. B. (1988).
Instrumentation for Ground-Based Optical Astronomy: Present and Future. The Ninth Santa Cruz Summer Workshop in Astronomy and Astrophysics, July 13–July 24, 1987, Lick Observatory.
Springer Verlag, New York, NY, USA.

Rockosi, C., Stover, R., Kibrick, R., Lockwood, C., Peck, M., Cowley, D., Bolte, M., Adkins, S., Alcott, B., Allen, S. L., Brown, B., Cabak, G., Deich, W., Hilyard, D., Kassis, M., Lanclos, K., Lewis, J., Pfister, T., Phillips, A., Robinson, L., Saylor, M., Thompson, M., Ward, J., Wei, M., and Wright, C. (2010).
The low-resolution imaging spectrograph red channel CCD upgrade: fully depleted, high-resistivity CCDs for Keck.
page 77350R, San Diego, California, USA.

Romero, E. (2018).
Proposal Information for SALT Call for Proposals: 2019 Semester 1.

Rousseeuw, P. J. (1987).
Silhouettes: A graphical aid to the interpretation and validation of cluster analysis.
Journal of Computational and Applied Mathematics, 20:53–65.

Sabater, J., Gómez, J. M., López, M., Torra, J., Raines, S. N., and Eikenberry, S. S. (2014).
Kinematic modeling and path planning for MIRADAS arms.
In Navarro, R., Cunningham, C. R., and Barto, A. A., editors, *Proc. SPIE 9151, Advances in Optical and Mechanical Technologies for Telescopes and Instrumentation, 91515S*, page 91515S, Montreal, Canada. SPIE.

Sabater, J., Riera-Ledesma, J., Torres, S., Garzón, F., Torra, J., and Gómez, J. M. (2016).
Target allocation and prioritized motion planning for MIRADAS probe arms.
In Chiozzi, G. and Guzman, J. C., editors, *Proc. SPIE 9913, Software and Cyberinfrastructure for Astronomy IV, 99132P (July 26, 2016)*, page 99132P, Edinburgh, United Kingdom.

Sabater, J., Torres, S., and Garzón, F. (2018a).
Roadmap search based motion planning for MIRADAS probe arms.
Journal of Astronomical Telescopes, Instruments, and Systems, 4(03):1.

Sabater, J., Torres, S., Garzón, F., and Gómez, J. M. (2018b).
Using clustering for disperse objects fields segmentation in MIRADAS instrument.
In Guzman, J. C. and Ibsen, J., editors, *Proc. SPIE 10707, Software and Cyberinfrastructure for Astronomy V, 107070N (6 July 2018)*, page 25, Austin, Texas, United States. SPIE.

Saha, M. and Isto, P. (2006).
Multi-Robot Motion Planning by Incremental Coordination.
In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5960–5963.

Sajid, Q., Luna, R., and Bekris, K. E. (2012).
Multi-Agent Pathfinding with Simultaneous Execution of Single-Agent Primitives.
In *SoCS*.

Saunders, W., Smedley, S., Gillingham, P., Forero-Romero, J. E., Jouvel, S., and Nord, B. (2014).
Target allocation yields for massively multiplexed spectroscopic surveys with fibers.
In *Modeling, Systems Engineering, and Project Management for Astronomy VI*, volume 9150, page 915023. International Society for Optics and Photonics.

Saunders, W., Smith, G., Gilbert, J., Muller, R., Goodwin, M., Staszak, N., Brzeski, J., Miziarski, S., and Colless, M. (2012).
'MOHAWK: a 4000-fiber positioner for DESpec.
page 84464W, Amsterdam, Netherlands.

Schaefer, C. E. R., Makarem, L., and Kneib, J.-P. (2016).
Target-based fiber assignment for large survey spectrographs.
page 991335.

Schrijver, A. (1986).
Theory of Linear and Integer Programming.
John Wiley & Sons.

Schrijver, A. (2002).
On the history of the transportation and maximum flow problems.
Mathematical Programming, 91(3):437–445.

Schwartz, J. T. and Sharir, M. (1983a).
On the piano movers' problem I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers.
Communications on Pure and Applied Mathematics, 36(3):345–398.

Schwartz, J. T. and Sharir, M. (1983b).

On the piano movers' problem II. General techniques for computing topological properties of real algebraic manifolds.
Advances in applied Mathematics, 4(3):298–351.

Schwartz, J. T. and Sharir, M. (1983c).
On the piano movers' problem III. Coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal barriers.
The International Journal of Robotics Research, 2(3):46–75.

Sharp, R., Saunders, W., Smith, G., Churilov, V., Correll, D., Dawson, J., Farrel, T., Frost, G., Haynes, R., Heald, R., Lankshear, A., Mayfield, D., Waller, L., and Whittard, D. (2006).
Performance of AAOmega: the AAT multi-purpose fiber-fed spectrograph.
page 62690G, Orlando, Florida , USA.

Sharples, R., Bender, R., Agudo Berbel, A., Bennett, R., Bezawada, N., Bouché, N., Bramall, D., Casali, M., Cirasuolo, M., Clark, P., Cliffe, M., Davies, R., Davies, R., Drory, N., Dubbeldam, M., Fairley, A., Finger, G., Genzel, R., Haefner, R., Hess, A., Jeffers, P., Lewis, I., Montgomery, D., Murray, J., Muschielok, B., Förster Schreiber, N., Pirard, J., Ramsay-Howat, S., Rees, P., Richter, J., Robertson, D., Robson, I., Rolt, S., Saglia, R., Schlichter, J., Tecza, M., Todd, S., Wegner, M., and Wiezorrek, E. (2010).
Recent progress on the KMOS multi-object integral-field spectrograph for ESO VLT.
In McLean, I. S., Ramsay, S. K., and Takami, H., editors, *Proc. SPIE 7735, Ground-based and Airborne Instrumentation for Astronomy III*, page 773515, San Diego, USA.

Sharples, R., Bender, R., Bennett, R., Burch, K., Carter, P., Casali, M., Clark, P., Content, R., Davies, R., Davies, R., Dubbeldam, M., Finger, G., Genzel, R., Haefner, R., Hess, A., Kissler-Patig, M., Laidlaw, K., Lehnert, M., Lewis, I., Moorwood, A., Muschielok, B., Förster Schreiber, N., Pirard, J., Ramsay Howat, S., Rees, P., Richter, J., Robertson, D., Robson, I., Saglia, R., Tecza, M., Thatte, N., Todd, S., and Wegner, M. (2006).
Design of the KMOS multi-object integral-field spectrograph.
In McLean, I. S. and Iye, M., editors, *Proc. SPIE 6269, Ground-based and Airborne Instrumentation for Astronomy, 62691C (June 28, 2006)*, volume 6269, pages 62691C–62691C–7, Orlando, USA. SPIE.

Sharples, R., Bender, R., Bezawada, N., Castillo, R., Cirasuolo, M., Davidson, G., Davies, R., Dubbeldam, M., Fairley, A., and others (2013).
First Light for the KMOS Multi-Object Integral-Field Spectrometer.
The Messenger (ESO), (151):21–23.

Sharples, R. M., Bender, R., Lehnert, M. D., Ramsay Howat, S. K., Bremer, M. N., Davies, R. L., Genzel, R., Hofmann, R., Ivison, R. J., Saglia, R., and Thatte, N. A. (2004).

KMOS: an infrared multiple-object integral field spectrograph for the ESO VLT.
In Moorwood, A. F. M. and Iye, M., editors, *Proc. SPIE 5492, Ground-based Instrumentation for Astronomy, 1179 (September 30, 2004*, volume 5492, pages 1179–1186. SPIE.

Sheinis, A. I. (2016).
First light results from the HERMES spectrograph at the AAT.
page 99081C, Edinburgh, United Kingdom.

Sheinis, A. I., Wolf, M. J., Bershady, M. A., Buckley, D. A. H., Nordsieck, K. H., and Williams, T. B. (2006).
The NIR upgrade to the SALT Robert Stobie Spectrograph.
page 62694T, Orlando, Florida , USA.

Sibson, R. (1973).
SLINK: An optimally efficient algorithm for the single-link cluster method.
The Computer Journal, 16(1):30–34.

Siciliano, B. and Khatib, O., editors (2008).
Springer Handbook of Robotics.
Springer Berlin Heidelberg, Berlin, Heidelberg.

Siciliano, B., Oriolo, G., Sciavicco, L., and Villani, L. (2009).
Robotics Modelling, Planning and Control.
Springer London, London.

Silver, D. (2005).
Cooperative Pathfinding.
1st Conference on Artificial Intelligence and Interactive Digital Entertainment, 1:117–122.

Simeon, T., Leroy, S., and Laumond, J.-P. (2002).
Path coordination for multiple mobile robots: a resolution-complete algorithm.
Robotics and Automation, IEEE Transactions on, 18(1):42–49.

Skillen, I., Balcells, M., and Trager, S. (2016).
Multi-Object Spectroscopy in the Next Decade: Big Questions, Large Surveys, and Wide Fields, volume 507.
Astron. Soc. Pac. San Francisco.

Snyman, J. (2005).
Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms, volume 97 of *Applied Optimization*.
Springer-Verlag, New York.

Standley, T. and Korf, R. (2011).
Complete algorithms for cooperative pathfinding problems.
In *IJCAI*, pages 668–673. Citeseer.

Staszak, N. F., Lawrence, J., Brown, D. M., Brown, R., Zhelem, R., Goodwin, M., Kuehn, K., Lorente, N. P., Nichani, V., Waller, L., and others (2016).
Taipan instrument fibre positioner and starbug robots: engineering overview.
In *Advances in Optical and Mechanical Technologies for Telescopes and Instrumentation II*, volume 9912, page 99121W. International Society for Optics and Photonics.

Stuetzle, W. (2003).
Estimating the cluster tree of a density by analyzing the minimal spanning tree of a sample.
Journal of classification, 20(1):025–047.

Su, D.-q., Cui, X., Wang, Y.-n., and Yao, Z. q. (1998).
Large-sky-area multiobject fiber spectroscopic telescope (LAMOST) and its key technology.
pages 76–90, Kona, HI.

Sánchez, B., Aguiar-González, M., Barreto, R., Becerril, S., Bland-Hawthorn, J., Bongiovanni, A., Cepa, J., Correa, S., Chapa, O., Ederoclite, A., Espejo, C., Farah, A., Fragoso, A. B., Fernández, P., Flores, R., Fuentes, F. J., Gago, F., Garfias, F., Gigante, J. V., González, J., González-Escalera, V., Hernández, B., Hernandez, E., Herrera, A., Herrera, G., Joven, E., Langarica, R., Lara, G., López, J. C., López, R., Militellon, C., Moreno, H., Peraza, L., Pérez, A., Pérez, J., Rasilla, J. L., Rosich, J., Tejada, C., Tinoco, S., Vaz, T., and Villegas, A. (2012).
OSIRIS tunable imager and spectrograph for the GTC: from design to commissioning.
page 84464T, Amsterdam, Netherlands.

Sánchez, G. and Latombe, J.-c. (2001).
A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking.
In *In Int. Symp. Robotics Research*, pages 403–417.

Sánchez, G. and Latombe, J.-C. (2002a).
On delaying collision checking in PRM planning: Application to multi-robot coordination.
The International Journal of Robotics Research, 21(1):5–26.

Sánchez, G. and Latombe, J.-C. (2002b).
Using a PRM planner to compare centralized and decoupled planning for multi-robot systems.
volume 2, pages 2112–2119. IEEE.

Talbi, E.-G. (2009).
Metaheuristics: from design to implementation.
Wiley Series on Parallel and Distributed Computing. John Wiley & Sons, Hoboken, N.J.

Tamura, N., Takato, N., Iwamuro, F., Akiyama, M., Kimura, M., Tait, P., Dalton, G. B., Murray, G. J., Smedley, S., Maihara, T., Ohta, K., Moritani, Y., Yabe, K., Sumiyoshi, M., Totani, T., Sugai, H., Karoji, H., Wang, S.-Y., and Ohyama, Y. (2012).
Subaru FMOS now and future.
page 84460M, Amsterdam, Netherlands.

Tamura, N., Takato, N., Shimono, A., Moritani, Y., Yabe, K., Ishizuka, Y., Ueda, A., Kamata, Y., Aghazarian, H., Arnouts, S., Barban, G., Barkhouser, R. H., Borges, R. C., Braun, D. F., Carr, M. A., Chabaud, P.-Y., Chang, Y.-C., Chen, H.-Y., Chiba, M., Chou, R. C. Y., Chu, Y.-H., Cohen, J., de Almeida, R. P., de Oliveira, A. C., de Oliveira, L. S., Dekany, R. G., Dohlen, K., dos Santos, J. B., dos Santos, L. H., Ellis, R., Fabricius, M., Ferrand, D., Ferreira, D., Golebiowski, M., Greene, J. E., Gross, J., Gunn, J. E., Hammond, R., Harding, A., Hart, M., Heckman, T. M., Hirata, C. M., Ho, P., Hope, S. C., Hovland, L., Hsu, S.-F., Hu, Y.-S., Huang, P.-J., Jaquet, M., Jing, Y., Karr, J., Kimura, M., King, M. E., Komatsu, E., Le Brun, V., Le Fèvre, O., Le Fur, A., Le Mignant, D., Ling, H.-H., Loomis, C. P., Lupton, R. H., Madec, F., Mao, P., Marrara, L. S., Mendes de Oliveira, C., Minowa, Y., Morantz, C., Murayama, H., Murray, G. J., Ohyama, Y., Orndorff, J., Pascal, S., Pereira, J. M., Reiley, D., Reinecke, M., Ritter, A., Roberts, M., Schwochert, M. A., Seiffert, M. D., Smee, S. A., Sodre, L., Spergel, D. N., Steinkraus, A. J., Strauss, M. A., Surace, C., Suto, Y., Suzuki, N., Swinbank, J., Tait, P. J., Takada, M., Tamura, T., Tanaka, Y., Tresse, L., Verducci, O., Vibert, D., Vidal, C., Wang, S.-Y., Wen, C.-Y., Yan, C.-H., and Yasuda, N. (2016).
Prime Focus Spectrograph (PFS) for the Subaru telescope: overview, recent progress, and future perspectives.
page 99081M, Edinburgh, United Kingdom.

Tao, D., Makarem, L., Bouri, M., Kneib, J.-P., and Gillet, D. (2018).
Priority coordination of fiber positioners in multi-objects spectrographs.
In *Ground-based and Airborne Instrumentation for Astronomy VII*, volume 10702, page 107028K. International Society for Optics and Photonics.

The Open Source Geospatial Foundation (2013).
GEOS — Geometry Engine, Open Source.

Toussaint, G. (1983).
Solving Geometric Problems with the Rotating Calipers.
In *In Proc. IEEE MELECON '83*, pages 10–02.

Turon, C., Primas, F., Binney, J., and others (2008).
ESA-ESO Working Group on Galactic Populations, Chemistry and Dynamics, Tech. rep.

Van Den Berg, J. P. and Overmars, M. H. (2005a).

Prioritized motion planning for multiple robots.
In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 430–435. IEEE.

Van Den Berg, J. P. and Overmars, M. H. (2005b).
Roadmap-based motion planning in dynamic environments.
Robotics, IEEE Transactions on, 21(5):885–897.

Wagner, G. and Choset, H. (2015).
Subdimensional expansion for multirobot path planning.
Artificial Intelligence, 219:1–24.

Wagner, R. M., Edwards, M. L., Kuhn, O., Thompson, D., and Veillet, C. (2014).
An overview and the current status of instrumentation at the Large Binocular Telescope Observatory.
page 914705, Montréal, Quebec, Canada.

Wang, K.-H. C., Botea, A., and Kilby, P. (2011).
On improving the quality of solutions in large-scale cooperative multi-agent pathfinding.
In *Fourth Annual Symposium on Combinatorial Search*.

Ward Jr, J. H. (1963).
Hierarchical grouping to optimize an objective function.
Journal of the American statistical association, 58(301):236–244.

Warren, C. (1990).
Multiple robot path coordination using artificial potential fields.
In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 500–505 vol.1.

Watson, F. (2007).
Stargazer: The Life and Times of the Telescope.
Allen and Unwin, Australia.

Wegner, M. and Muschielok, B. (2008).
KARMA: the observation preparation tool for KMOS.
In *Proc. SPIE 7019, Advanced Software and Control for Astronomy II, 70190T (July 14, 2008)*, page 70190T, Marseille, France. SPIE.

Welzl, E. (1991).
Smallest Enclosing Disks (balls and Ellipsoids).
In *Results and New Trends in Computer Science*, pages 359–370. Springer-Verlag.

Wilson, R. N. (2013).
Reflecting telescope optics II: manufacture, testing, alignment, Modern Techniques.
Springer Science & Business Media.

Wise, K. D. and Bowyer, A. (2000).
A survey of global configuration-space mapping techniques for a single robot in a static environment.
The International Journal of Robotics Research, 19(8):762–779.

Yasuda, T., editor (2011).
Multi-Robot Systems, Trends and Development.
InTech.

Yershova, A. and LaValle, S. M. (2007).
Improving Motion-Planning Algorithms by Efficient Nearest-Neighbor Searching.
IEEE Transactions on Robotics, 23(1):151–157.

Yu, J. and LaValle, S. M. (2013).
Multi-agent path planning and network flow.
In *Algorithmic foundations of robotics X*, pages 157–173. Springer.

Yuan, H., Zhang, H., Zhang, Y., Lei, Y., and Dong, Y. (2013).
Development of target allocation methods for LAMOST focal plate.
Proceedings of the International Astronomical Union, 9(S298):452.

Zombeck, M. V. (2006).
Handbook of space astronomy and astrophysics.
Cambridge University Press.

Čáp, M., Novák, P., Kleiner, A., and Selecký, M. (2015).
Prioritized Planning Algorithms for Trajectory Coordination of Multiple Mobile Robots.
IEEE Transactions on Automation Science and Engineering, 12(3):835–849.

Čáp, M., Novák, P., Vokrínek, J., and Pěchouček, M. (2013).
Multi-agent RRT: sampling-based cooperative pathfinding.
In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1263–1264. International Foundation for Autonomous Agents and Multiagent Systems.

Švestka, P. and Overmars, M. H. (1998).
Coordinated path planning for multiple robots.
Robotics and Autonomous Systems, 23(3):125–152.