Sección de Matemáticas
Universidad de La Laguna

Pablo Yanes Riquelme

Un problema de optimización combinatoria en puertos marítimos

A combinatorial optimization problem in seaports

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, Julio de 2023

Dirigido por
Hipólito Hernández Pérez
María del Socorro García Román

Hipólito Hernández Pérez
Departamento de Matemáticas, Estadística e Investigación Operativa
Universidad de La Laguna
38200 La Laguna, Tenerife

María del Socorro García Román
Departamento de Álgebra
Universidad de La Laguna
38200 La Laguna, Tenerife

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas e instituciones que han sido parte fundamental en la realización de este Trabajo de Fin de Grado.

En primer lugar, me gustaría agradecer a mi tutor, Hipólito Hernández Pérez, por su orientación experta, dedicación y apoyo durante todo el proceso de investigación. Sus conocimientos y consejos han sido de gran valor para el desarrollo de este trabajo, y estoy realmente agradecido por su guía y paciencia.

También quiero expresar mi gratitud a mi familia por su incondicional apoyo y comprensión a lo largo de mi carrera académica. Sus palabras de aliento y su amor incondicional han sido mi fuente de motivación en los momentos más desafiantes. No puedo agradecerles lo suficiente por estar a mi lado y creer en mí.

Además, me gustaría reconocer y agradecer a Híades Business Patterns SL, haciendo una especial mención a María del Socorro García Román por su generosa colaboración y por brindarme la oportunidad de trabajar en un entorno empresarial real. Su apoyo y recursos han sido fundamentales para llevar a cabo la implementación y evaluación de este trabajo, permitiéndome adquirir experiencia práctica y enriquecedora.

Por último, pero no menos importante, quiero agradecer a mis amigos por su constante apoyo, ánimo y compañerismo. Su presencia en mi vida ha hecho que este camino sea más significativo y memorable. Su confianza en mí y su amistad han sido un regalo invaluable, y estoy agradecido por cada momento compartido y por su inquebrantable apoyo.

A todos ustedes, mi tutor, mi familia, la empresa colaboradora y mis amigos, les estoy profundamente agradecido por su contribución y por ser parte esencial en el éxito de este Trabajo de Fin de Grado. Su influencia y apoyo han dejado una huella duradera en mi camino académico y personal. ¡Gracias de corazón!

Pablo Yanes Riquelme
La Laguna, 4 de marzo de 2024

Resumen · Abstract

Resumen

La optimización combinatoria, como rama de las matemáticas, se ocupa de resolver problemas en los que se busca encontrar la mejor solución dentro de un conjunto finito, pero generalmente muy grande, de opciones. En este contexto, vamos a desarrollar e implementar en Python un problema específico que está orientado a optimizar la entrada y salida de buques en un puerto.
Utilizaremos los datos proporcionados por la empresa Híades Business Patterns SL, que nos brinda información relevante sobre los buques, los recursos disponibles en el puerto de Valencia y los requisitos de descanso de los operarios. Nuestro objetivo será obtener diferentes resultados óptimos considerando las diferentes combinaciones de recursos disponibles y requisitos de descanso.
Al aplicar técnicas de optimización combinatoria, podremos evaluar si los recursos disponibles en el puerto son insuficientes, suficientes o excesivos para manejar eficientemente el flujo de entrada y salida de buques. Además, podremos analizar cómo diferentes requisitos de descanso de los operarios afectan a la optimización del proceso.
Al finalizar el análisis y la implementación en Python, podremos llegar a conclusiones fundamentadas acerca de la capacidad del puerto de Valencia para manejar el tráfico de buques de manera eficiente, determinando si se requieren cambios en la asignación de recursos o en los requisitos de descanso para lograr una optimización adecuada.

Palabras clave: *Optimización combinatoria – Problemas de puertos.*

Abstract

Combinatorial optimization, as a branch of mathematics, deals with solving problems in which the goal is to find the best solution within a finite but typically very large set of options. In this context, we will develop and implement a specific problem in Python aimed at optimizing the entry and exit of ships in a port.
We will utilize the data provided by "Híades Business Patterns SL", which provides relevant information about the ships, the available resources in the port of Valencia, and the rest requirements of the operators. Our objective will be to obtain different optimal results considering the various combinations of available resources and rest requirements.
By applying combinatorial optimization techniques, we will evaluate whether the available resources in the port are insufficient, sufficient, or excessive to efficiently handle the flow of ship entries and exits. Additionally, we will analyze how different rest requirements for the operators impact the optimization process.
Upon completing the analysis and implementation in Python, we will draw well-founded conclusions regarding the capacity of the port of Valencia to handle ship traffic efficiently, determining whether changes in resource allocation or rest requirements are necessary to achieve proper optimization.

Keywords: *Combinatorial Optimization – Port Problems.*

Contenido

Introducción

En el contexto de esta investigación, se ha llevado a cabo un estudio en el campo de la optimización combinatoria para brindar una solución computacional a un problema real en el sector portuario. Este tema es importante, ya que actualmente la toma de decisiones relacionadas con la entrada y salida de buques se realiza de forma manual, y existe la necesidad de implementar enfoques más eficientes y automatizados.

Agentes clave en esta investigación son los buques que solicitan el servicio de entrada/salida, así como los remolcadores y prácticos encargados de realizar dichos servicios. Es fundamental tener en cuenta que el retraso en la entrada o salida de un buque puede generar costos significativos y afectar la toma de decisiones sobre qué puertos priorizar en términos de mejores servicios.

Los objetivos de esta disertación se centran en desarrollar una solución computacional utilizando herramientas de optimización combinatoria para mejorar la eficiencia y la toma de decisiones en el proceso de entrada y salida de buques en un puerto específico. A través de este enfoque, se busca proporcionar una primera versión de un producto que pueda ser implementado en la empresa colaboradora y, en última instancia, extrapolarlo a otros recintos portuarios.

Este estudio se lleva a cabo como resultado del interés personal en resolver problemas reales utilizando los conocimientos adquiridos durante la carrera, aprovechando la oportunidad brindada por la empresa en la que se realizaron las prácticas externas. Aunque los resultados actuales se basan en una versión simplificada del problema, existe la posibilidad de continuar desarrollando y mejorando la solución hasta convertirla en un producto plenamente implementable en la empresa.

Con esta estructura y enfoque, se espera que esta disertación contribuya al campo de la optimización combinatoria aplicada al sector portuario, ofreciendo una solución computacional que mejore la eficiencia en el proceso de entrada y salida de buques, y brinde beneficios tanto a la empresa colaboradora como a otros recintos portuarios en el futuro.

El Capítulo 1 de esta disertación abordará la presentación de la empresa colaboradora, proporcionando una descripción detallada de su contexto y relevancia en el ámbito de estudio. Al mismo tiempo, se introducirán los conceptos teóricos fundamentales de la programación combinatoria, estableciendo un marco sólido que servirá como base para el desarrollo y la comprensión del problema específico abordado en la disertación.

En el Capítulo 2, se llevará a cabo la implementación práctica del problema concreto, describiendo minuciosamente los pasos y las estrategias utilizadas.

Finalmente, en el Capítulo 3, se explicará cómo se ha hecho la recopilación y el procesamiento de datos, se presentarán de forma clara y concisa los resultados computacionales obtenidos, junto con los hallazgos y las conclusiones derivadas del análisis de los datos recopilados y del proceso de optimización. Además, se mostrarán representaciones gráficas de soluciones para facilitar su comprensión e interpretación.

Estos capítulos, interconectados de manera coherente, permitirán a los lectores seguir el desarrollo de la disertación y apreciar plenamente los avances realizados en la investigación.

1

La empresa e introducción a la optimización combinatoria

A continuación daremos una introducción a los diferentes aspectos que se van a trabajar en esta memoria, comenzando por la empresa a la que está asociado este trabajo con sus productos, continuando con una introducción a la optimización combinatoria y a diferentes problemas relacionados con el que trabajaremos en esta memoria, y por último, las herramientas computacionales que se utilizarán.

1.1. La empresa: Híades Business Patterns SL

Híades Business Patterns SL es una empresa tecnológica con más de 12 años de experiencia que está especializada en el sector portuario, con presencia en España y varios países de Latinoamérica, en más de veinte puertos, con sus productos 'AMURA', que permiten la digitalización de los datos y eventos de las maniobras y operaciones que tienen lugar en la lámina de agua de la zona portuaria. En los últimos años HÍADES, ha iniciado una nueva línea de negocio en el Departamento de I+D+i, dedicada a la explotación de los datos a través de analítica avanzada para la resolución de casos de uso que presentan los distintos agentes del sector portuario, principalmente aplicando algoritmos de *Machine Learning* para la creación de modelos predictivos, como puede ser una predicción de la hora de llegada de un buque a un puerto o la clusterización de rutas a partir de datos AIS(*Automatic Identification System)*, los datos AIS son una señal que emite el buque en tiempo real cada 3 o 6 segundos informando de su posición, velocidad, tamaño y destino entre otros datos.

Figura 1.1: Logo empresa

`Amura pilots`

`Amura pilots` es la solución tecnológica enfocada para prácticos. Los prácticos son los encargados del atraque y la salida de los buques de un puerto, son capitanes de barco, que conocen a la perfección el puerto y sus características por lo que se encargan de esta complicada tarea, que unifica toda la gestión administrativa y operativa en una interfaz intuitiva y flexible que se adapta a todas las necesidades del practicaje marítimo (labor que realizan los prácticos). Entrando más en profundidad en sus funcionalidades notamos que `Amura pilots` tiene las siguientes funcionalidades:

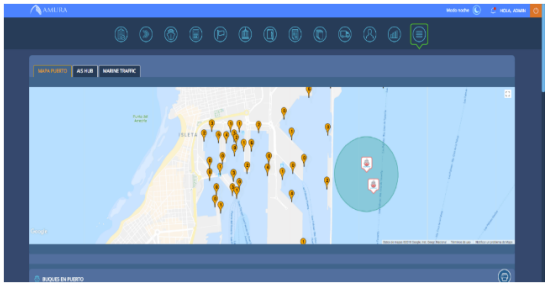

Figura 1.2: Mapa del puerto mostrado desde `Amura pilots`

- Previsión Centralizada del tráfico en Buques, la cual aporta control sobre entradas y salidas, información en tiempo real acerca del zarpe, calados, mareas y atraques, además de un alto nivel de seguridad en conexión a tiempo

real (a través de aplicaciones como *Vessel Finder* o *Marine Traffic*, las cuales permiten ubicar buques en tiempo real).

- `Amura pilots` Pro. Es una aplicación de software móvil diseñada para los Prácticos para la programación de buques y maniobras. Fichas técnicas de embarcaciones, maniobras o corporación. Firmas electrónicas e informes.

- Gestión de movimiento. Libro digital de servicios ágil e intuitivo para gestionar estados de movimiento de buques y documentación. Gestión de maniobras de pilotaje y recursos (pilotos, lanchas, remolcadores). Cálculo de tarifas. Fichas de seguridad de navegación. Informes y gráficos.

- Consignatarias. Gestión financiera personalizada y automatizada de consignatarias y navieras. Control de cobros y alertas de vencimientos de facturas. Acceso a un portal web exclusivo para agentes marítimos para consulta de previsión en servicios.

- Embarcaciones y personal. Administración del personal asignado en las embarcaciones con monitoreo cerrado (Web y App). Turnos laborales y reportes históricos por actividad, gráficos. Disponible sistema de detección y alarma antifatiga.

- Optimización del proceso administrativo y facturación. Trazabilidad de los procesos de practicaje y las operaciones financieras: tarifas, facturas y pagos. Auditoría con inspección inmediata de datos. Conexión a sistemas estatales de facturación y contabilidad. Bolsa salarial y finiquitos.

- Planificación de mantenimiento de a infraestructuras. Control de activos físicos, mantenimiento preventivo y correctivo producidos para la prestación del servicio de practicaje.

- Cuadros de mando e informes. Estadísticas e Indicadores con informes de eficiencia, análisis de productividad y facturación clasificados por prácticos, consignatarias, buques. Informes históricos y gráficos exportables PDF o Excel.

Figura 1.3: Listado de previsiones, Listado de Mareas y Prácticos de Guardias:

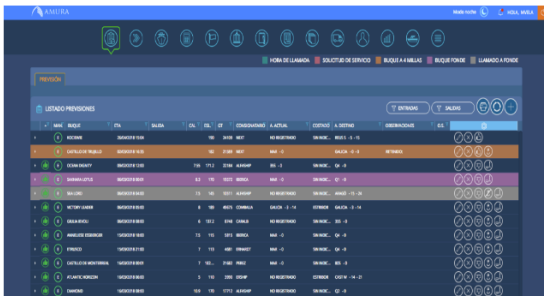

Figura 1.4: Ficha de una Maniobra de Un Práctico:

Esta memoria va orientada a la explotación de estos datos obtenidos a través del producto `Amura pilots` y de los datos AIS, en un puerto.

1.2. Introducción a la optimización combinatoria:

La optimización combinatoria es un campo de estudio fundamental en la optimización matemática. Se centra en la resolución de problemas en los cuales se busca encontrar la mejor solución posible dentro de un conjunto finito de opciones. Estos problemas implican la selección o combinación óptima de elementos, teniendo en cuenta diversas restricciones y objetivos predefinidos.

Los problemas de optimización combinatoria suelen tener un espacio de búsqueda exponencial, es decir, que la cantidad de posibles soluciones crece de manera exponencial en relación al tamaño del problema. Esto hace que, en la mayoría de los casos, encontrar la mejor solución sea un desafío computacional.

Algunos ejemplos comunes de problemas de optimización combinatoria incluyen el problema del viajante (*Traveling Salesman Problem*), el problema de la mochila (*Knapsack Problem*), el problema del emparejamiento estable (*Stable Matching Problem*) y el problema del conjunto independiente máximo (*Maximum Independent Set Problem*), entre muchos otros.

Para resolver problemas de optimización combinatoria, se utilizan diferentes enfoques, como algoritmos exactos, heurísticas y metaheurísticas. Los algoritmos exactos garantizan encontrar la solución óptima, pero pueden ser computacionalmente costosos para problemas grandes. Las heurísticas y metaheurísticas ofrecen aproximaciones rápidas y eficientes, aunque no garantizan encontrar la solución óptima.

En otras palabras, la optimización combinatoria se dedica a resolver situaciones donde se debe elegir la mejor combinación de elementos, determinando cómo seleccionarlos y ordenarlos de manera más favorable. Esto implica analizar todas las posibles combinaciones y evaluarlas en función de criterios específicos. La optimización combinatoria tiene aplicaciones en diversos campos, como la planificación de rutas, la asignación de recursos, la programación de horarios y muchos otros problemas del mundo real.

1.3. Problema del viajante de comercio

El Problema del Viajante de Comercio (TSP, por sus siglas en inglés, *Traveling Salesman Problem*) es uno de los problemas más prominentes en el campo de la optimización combinatoria. En este problema, un viajante desea visitar un

conjunto de n ciudades, recorriendo cada una de ellas una sola vez y regresando al lugar de partida. Supongamos que se conocen las distancias entre todas las parejas de ciudades. El objetivo es determinar la secuencia óptima de visitas para minimizar la distancia total recorrida.

El TSP se puede visualizar como una red de nodos, que pueden representar ciudades o ubicaciones dentro de una ciudad. Se parte de un lugar inicial y se deben recorrer todas las localizaciones sin repetir ninguna, regresando al lugar inicial. Cada arco entre nodos tiene asociado un valor que representa la distancia o el costo de desplazarse de un nodo a otro. El objetivo es encontrar el orden de visita de los nodos que minimice la distancia total recorrida.

Es importante destacar que el TSP es un problema NP-duro, lo que significa que no existe un algoritmo polinomial que pueda resolverlo de manera exacta en tiempo razonable. La complejidad del problema radica en el gran número de posibles soluciones, que crece de manera exponencial con el número de ciudades. Por ejemplo, con 10 ciudades hay más de 3 millones de posibilidades, y con 100 ciudades, el número de posibilidades supera el 10^{156}.

Existen dos variantes principales del TSP, el TSP simétrico y el TSP asimétrico. En el TSP simétrico, se asume que la distancia entre dos ciudades es la misma en ambas direcciones, lo que forma un grafo no dirigido. Esto reduce a la mitad el número de posibles soluciones. Por otro lado, en el TSP asimétrico, puede haber caminos que solo sean posibles en una dirección o las distancias pueden ser diferentes en cada dirección, formando así un grafo dirigido. Un ejemplo de falta de simetría sería considerar las calles de una ciudad, donde puede haber calles de sentido único.

A pesar de la dificultad computacional del TSP, existen algoritmos heurísticos y aproximados que pueden proporcionar soluciones subóptimas en tiempos razonables. Estos algoritmos ofrecen soluciones que se acercan al óptimo, pero no pueden garantizar la optimalidad debido a la complejidad intrínseca del problema.

Formulacion del TSP.

Formulación de Dantzig-Fulkerson-Johnson (1954). El caso más general es el caso asimétrico, aquel en el que el coste de ir de la localización i a la j no tiene porqué ser igual al de ir de la j in i.

Sea $G = (V, A)$ un grafo donde $V = \{1, 2, \ldots, n\}$ es el conjunto de vértices (nodos) y A es el conjunto de arcos. Cada arco $a \in A$ también lo denotaremos por (i, j) donde i es el origen y j es el destino de los dos vértices que conecta.

Para cada $(i, j) \in A$, denotaremos por c_{ij} el casto de ir del vértice i al j, también lo denotaremos por c_a donde $a = (i, j)$.

Para la formulación matemática de este problema introducimos una variable x_{ij} para cada arco (i, j) en A donde:

$$x_{ij} = \begin{cases} 1 & \text{si el arco } (i,j) \text{ está en la ruta} \\ 0 & \text{en otro caso} \end{cases}$$

Entonces el problema puede ser formulado como:

$$\text{mín} \sum_{i,j \in V} c_{ij} x_{ij}$$

sujeto a:

$$\sum_{j \in V : j \neq i} x_{ij} = 1 \qquad \forall i \in V \tag{1.1}$$

$$\sum_{j \in V : j \neq i} x_{ji} = 1 \qquad \forall i \in V \tag{1.2}$$

$$x_{ji} \leq |S| - 1 \qquad \forall S \in V \tag{1.3}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V, j \neq i \tag{1.4}$$

1.4. Problemas de optimización combinatoria relacionadas con puertos

Existen varios problemas de optimización combinatoria relacionados con puertos marítimos y actividades portuarias. Algunos ejemplos de estos problemas incluyen:

1. El problema de asignación de sitios de atraque: Cuando los buques llegan a un puerto, deben esperar a que se les asigne una posición donde atracar en el muelle de la terminal destino. El muelle es una plataforma que sobresale en el agua para facilitar la carga y descarga de los contenedores. Los lugares donde se llevan a cabo estas tareas en el puerto se denominan atracaderos, o sitios de atraque. Los sitios de atraque están equipados con grúas de muelle

que se utilizan para cargar y descargar contenedores; quienes se transfieren, respectivamente, desde y hacia el patio por una flota de vehículos. El problema consiste en asignar los buques que llegan a una terminal, a los sitios de atraque. Desde una perspectiva general, este problema involucra dos decisiones relacionadas entre sí: dónde y cuándo asignar los buques a los sitios de atraque, estas decisiones de asignación de sitios de atraque tienen inferencia directa en los costos de operación de una terminal y los tiempos de atención a buques.

El problema de asignación de sitios de atraque consiste en determinar la mejor manera de asignar cada uno de los buques que llegan a una terminal de contenedores a un sitio de atraque, para operaciones de carga y descarga, de forma tal que se optimice alguna función de costo. Las características de los buques, de la terminal y de los recursos de la misma, determinan qué restricciones deben cumplirse para obtener una solución factible. Además, pueden considerarse diferentes funciones de costo para medir la eficiencia de la terminal; aunque el tiempo total que el buque pasa en la terminal (hora de salida del buque menos su hora de llegada) continua siendo un criterio importante para evaluar la eficiencia de un puerto.

Sean $V = \{1, ..., T\}$ el conjunto de sitios de atraque disponibles en dicha terminal y $B = \{1, ..., I\}$ el conjunto de buques que llegan a una terminal de contenedores. Para cada buque $j \in V$ denotamos por A_j su tiempo estimado de llegada a la terminal, para cada sitio de atraque $i \in B$ denotamos por S_i el tiempo a partir del cual está disponible, y, para cada buque $j \in V$ y sitio de atraque $i \in B$ denotamos por c_{ij} el tiempo estimado de servicio en operaciones de carga y descarga, en el sitio de atraque i. A cada sitio de atraque, le asignan un orden inverso sobre el cual los buques serán atendidos, tal que $O = \{1, ..., T\}$ es el conjunto de lugares disponibles para el servicio en un sito de atraque. A partir de lo anterior, se definen los siguientes subconjuntos:

$$P_k = \{m \in O \mid m > k\} \text{ para todo } k \in O$$

$$W_i = \{j \in V \mid A_j > S_i\} \text{ para todo } i \in B$$

donde P_k representa el conjunto de posiciones posteriores al orden k, y W_i es el conjunto de todos los buques que llegan después de que el sitio de atraque i está disponible. Finalmente, considerando el orden inverso en el cual los buques j se atienden en el sitio de atraque i, se definen las siguientes variables de decisión:

- $x_{ijk} \in \{0, 1\}$. Vale 1 si el buque j se sirve en la última k-ésima posición disponible en el sitio de atraque i, y vale 0 en otro caso.

- y_{ijk} = es el tiempo ocioso del sitio de atraque i antes de la llegada del buque j, que será atendido en la última k-ésima posición.

$$\text{mín} \sum_{i\in B}\sum_{j\in V}\sum_{k\in O}(kC_{ij}+S_i-A_j)X_{ijk}+\sum_{i\in B}\sum_{j\in V}\sum_{k\in O}ky_{ijk} \qquad \forall j\in V \quad (1.5)$$

sujeto a:

$$\sum_{i\in B}\sum_{k\in O}x_{ijk}=1 \qquad \forall j\in V \qquad (1.6)$$

$$\sum_{i\in V}x_{ijk}\leq 1 \qquad \forall i\in B, k\in O \qquad (1.7)$$

$$\sum_{l\in V}\sum_{m\in P_k}(C_{il}x_{ijk}+y_{ilm})+y_{ijk}\geq (A_j-S_j)x_{ijk} \qquad \forall i\in B, j\in W_i, k\in O \quad (1.8)$$

$$x_{ijk}\in\{0,1\} \qquad \forall i\in B, j\in V, k\in O \quad (1.9)$$

$$y_{ijk}\geq 0 \qquad \forall i\in B, j\in V, k\in O \quad (1.10)$$

$$(1.11)$$

La función objetivo (1.5) minimiza los tiempos totales de espera y trabajo en la atención de todos los buques que llegarán durante el horizonte planificado, así como los tiempos ociosos de los sitios de atraque. Las restricciones (1.6) aseguran que todos los buques sean atendidos, mientras que las restricciones (1.7) establecen una condición de factibilidad, evitando que dos o más buques se atiendan en un mismo sitio de atraque al mismo tiempo. Las restricciones (1.8) computan los tiempos de ocio de los sitios de atraque, además de que condicionan la atención de los buques a un tiempo después de su llegada. Finalmente, (1.9) y (1.10) hacen referencia a las restricciones técnicas.

2. Problema de carga y descarga de contenedores: En este problema, se busca optimizar la secuencia de carga y descarga de contenedores en un barco o en una terminal de contenedores del puerto. El objetivo es maximizar la eficiencia en el manejo de los contenedores, minimizando los tiempos de espera y los movimientos innecesarios, teniendo en cuenta las restricciones de peso, tamaño y prioridades de carga/descarga.

3. Problema de planificación de grúas y equipos portuarios: Este problema implica asignar las grúas y otros equipos portuarios de manera óptima para realizar las operaciones de carga y descarga de los barcos o para el movimiento

de contenedores dentro del puerto. El objetivo es minimizar el tiempo total de procesamiento y los costos operativos, teniendo en cuenta la disponibilidad de equipos, los tiempos de cambio y las restricciones de seguridad.

4. Problema de enrutamiento de vehículos en el puerto: Este problema se relaciona con la planificación de rutas para los vehículos que transportan carga dentro del puerto, como camiones o carretillas elevadoras. El objetivo es minimizar la distancia recorrida, optimizar los tiempos de entrega y minimizar los congestionamientos de tráfico dentro del puerto, considerando las restricciones de seguridad y los tiempos de espera en las áreas de carga y descarga.

Estos son solo algunos ejemplos de problemas de optimización combinatoria relacionados con puertos. Cada uno de ellos presenta desafíos específicos que requieren técnicas y enfoques adecuados para encontrar soluciones óptimas o aproximadas. La optimización en la gestión portuaria es fundamental para mejorar la eficiencia, reducir los costos y maximizar la capacidad de los puertos en un contexto global de comercio y logística.

1.5. Herramientas computacionales

Gurobi es una poderosa herramienta de optimización para resolver problemas de programación lineal. Es un software comercial líder en el campo de la optimización matemática y es ampliamente utilizado en diversas industrias y ámbitos académicos.

Una de las características destacadas de Gurobi es su capacidad para resolver problemas de programación lineal de gran escala de manera eficiente y rápida. Utiliza algoritmos avanzados y técnicas de optimización de vanguardia para encontrar soluciones óptimas en un tiempo mínimo. Esto es especialmente beneficioso cuando se enfrentan problemas con un gran número de variables y restricciones, ya que Gurobi está diseñado para manejar problemas complejos de manera eficiente.

La interfaz de Gurobi es fácil de usar, una de las opciones es utilizar su librería para Python y es así como nosotros la hemos usado. Permite a los usuarios definir variables, restricciones y la función objetivo de manera clara y concisa, lo que facilita la formulación de modelos de programación lineal. Además, Gurobi proporciona una amplia documentación y recursos de soporte, lo que ayuda a

los usuarios a comprender y aprovechar al máximo todas las capacidades del software.

Gurobi también ofrece una funcionalidad robusta para manejar problemas de programación lineal mixta (PLM), donde se combinan variables continuas y variables binarias o enteras. Además de la programación lineal, Gurobi también es capaz de resolver problemas de programación cuadrática y problemas de programación no lineal, lo que lo convierte en una herramienta versátil para abordar una amplia gama de desafíos de optimización.

En resumen, Gurobi es una herramienta confiable y eficiente para resolver problemas de programación lineal y mixta. Su capacidad para manejar problemas de gran escala y su rendimiento rápido hacen de Gurobi una elección popular en la comunidad de optimización. Ya sea en la industria o en la investigación académica, Gurobi proporciona una solución poderosa para optimizar y mejorar la toma de decisiones en una variedad de problemas de optimización.

2

Formulación del problema

En el Capítulo 1, se exploraron diversos ejemplos de problemas de optimización combinatoria, lo que nos permitió comprender mejor la naturaleza y los desafíos inherentes a este campo. Además, se presentó una breve introducción a la empresa colaboradora que ha facilitado los datos necesarios para llevar a cabo este proyecto.

A continuación, nos enfocaremos en desarrollar el problema específico que estamos abordando. Comenzaremos proporcionando una descripción detallada del problema en cuestión, donde examinaremos la dinámica de entrada y salida de buques en el puerto de Valencia. Consideraremos aspectos como los recursos disponibles, las capacidades operativas y los requisitos de descanso de los operarios que influyen en la optimización del proceso.

Posteriormente, procederemos a formular el problema de manera matemática. Esta formulación nos permitirá definir claramente las variables, restricciones y la función objetivo que buscamos optimizar.

En resumen, en este capítulo continuaremos desarrollando nuestro proyecto, centrándonos en un problema concreto relacionado con la gestión de entrada y salida de buques en un puerto. A través de una descripción precisa y una formulación matemática adecuada, estableceremos las bases necesarias para abordar el problema desde una perspectiva de optimización combinatoria.

2.1. Descripción del problema

El problema que vamos a abordar busca minimizar el tiempo de espera de las operaciones de entrada y salida de buques en un puerto. Haciendo uso de la

programación lineal partiremos de una primera aproximación, la cual será lo más sencilla posible, obviando todas aquellas características que por su complejidad hacen mucho más difícil resolver el problema, y/o que por su irrelevancia no aportan valor a la solución.

Para nuestra primera aproximación tenemos que:

- Hay un número de prácticos disponibles cada día
- Hay un número de remolcadores disponibles cada día
- Existe un número de operaciones a atender cada día.
- Cada operación requiere de la participación de un práctico y de r remolcadores (r será normalmente un número entre 0 y 2).
- Cada operación tiene un tiempo de proceso conocido.
- Los agentes tardan un tiempo conocido en desplazarse de una operación a la siguiente.
- Hay algunas otras restricciones existentes que hay que tener en cuenta.
- La función objetivo será minimizar el tiempo de espera de los buques.

2.1.1. Restricciones existentes

A continuación estudiaremos las restricciones existentes para estas operaciones:

- Los barcos de pasajeros tienen preferencia sobre los buques de mercancías.
- Los desplazamientos frecuentes tienen preferencia sobre los ocasionales.
- A los prácticos se les exige un descanso de 15 minutos después de 2 horas continuadas de trabajo.

2.2. Primer modelo

2.2.1. Notación

Para describir nuestro primer modelo para este problema, primero definimos una serie de conjuntos, luego listamos los parámetros del problema y finalmente definimos un grafo.

Definición de conjuntos:

- $I = \{1, ..., n\}$ es el conjunto de operaciones (indexadas por el índice i).
- $\{0, 0'\}$ son dos operaciones especiales que denotan el comienzo y el final (respectivamente) de un agente.
- $V = I \cup \{0, 0'\}$.
- $P = \{1, ..., p\}$ es el conjunto de prácticos.
- $R = \{p + 1, ..., p + r\}$ es el conjunto de remolcadores.
- $K = P \cup R$: conjunto de los agentes involucrados.

Lista de parámetros:

- w_i es el coste de espera de la operación i por unidad de tiempo.
- e_i es la hora *estimada* en la que la operación i puede comenzar. Equivale a la hora estimada en que un buque llega a puerto o puede salir del puerto dependiendo de si se trata de una operación de entrada o salida.
- p_i es el tiempo de proceso de la operación i.
- t_{ij}^k es el tiempo que el agente k necesita para, una vez finalizada la operación i, comience la operación j.
- r_i es el número de remolcadores necesarios para la operación i.

Así, definimos $G(V, A)$ como un grafo dirigido, donde $V = I \cup \{0, 0'\}$ es el conjunto de todas las operaciones y $A \subseteq \{(i, j) : i, j \in V, i \neq j\}$ se corresponde con los pares de operaciones (i, j) de forma que la operación j se puede realizar justo después de la operación i.

2.2.2. Variables de decisión

Para cada $(i, j) \in A$ y cada $k \in K$, tenemos la variable:

$$X_{ij}^k = \begin{cases} 1 \text{ si el agente } k \text{ va de la operación } i \text{ a la operación } j \\ 0 \text{ en otro caso} \end{cases}$$

La idea es que para cada agente k, las variables X_{ij}^k describen un camino desde 0 hasta $0'$ con las secuencias de operaciones que realiza dicho agente k.

Para cada $i \in I$ definimos la variable:

$$S_i = \text{tiempo de comienzo de la operación } i$$

Para cada $i \in I$ y cada $k \in K$ definimos la variable:

$$Y_i^k = \begin{cases} 1 \text{ si el agente } k \text{ realiza la operación } i \\ 0 \text{ en otro caso} \end{cases}$$

2.2.3. Modelo matemático

Nuestra función objetivo será la suma de los tiempos de espera de cada buque ponderada por los pesos w_i.

$$\text{mín} \sum_{i \in I} (S_i - e_i) wi \tag{2.1}$$

sujeto a:

$$\sum_{a \in \delta^+(0)} X_a^k = \sum_{a \in \delta^-(0')} X_a^k = 1 \qquad \forall k \in K \tag{2.2}$$

$$\sum_{a \in \delta^+(i)} X_a^k = \sum_{a \in \delta^-(i)} X_a^k \qquad \forall k \in K, \forall i \in V \tag{2.3}$$

$$S_j \geq S_i + P_i + t_{ij}^k - M(1 - X_{ij}^k) \qquad \forall k \in K \; \forall i \in V \tag{2.4}$$

$$\sum_{k \in R} Y_i^k = r_i \qquad \forall i \in I \tag{2.5}$$

$$\sum_{k \in P} Y_i^k = 1 \qquad \forall i \in I \tag{2.6}$$

$$S_i \geq e_i \qquad \forall i \in I \tag{2.7}$$

Interpretación de las restricciones:

- La función objetivo (2.1) minimiza los tiempos de esperas de los buques ponderados por el peso w_i.
- La familia de restricciones (2.2) nos dice que todos los agentes deben salir del nodo inicial y llegar al nodo final.
- La familia de restricciones (2.3) exige que los agentes que entran a una operación deben ser los mismos que los que salen.
- La familia de restricciones (2.4) establece que el tiempo de inicio de la operación j tiene que ser mayor igual al tiempo de inicio de la operación i más el tiempo de proceso de la operación i más el tiempo que tardan los agentes en ir de la operación i a la j, siempre y cuando haya un arco que vaya de i a j en nuestro Grafo.
- La familia de restricciones (2.5) establece que el número de remolcadores que realizan la operación i sea igual al requerido.
- La familia de restricciones (2.6) dice que a cada operación debe acudir un práctico.
- La familia de restricciones (2.7) simplemente establece que la operación i no puede comenzar antes de que llegue el buque.

2.3. Variante con descansos de los prácticos

Para esta variante añadimos a las variables de decisión anteriores la siguiente familia de variables.

$$Z_i^k = \begin{cases} 1 & \text{si el agente } k \text{ descansa tras la operación } i \\ & 0 \text{ otro caso} \end{cases}$$

Con esta nueva familia de variables y las variables del modelo anterior formulamos el problema de la siguiente forma.

$$\text{mín} \sum_{i \in I} (S_i - e_i) wi$$

Sujeto a:

$$\sum_{a \in \delta^+(0)} X_a^k = \sum_{a \in \delta^-(0')} X_a^k = 1, \forall k \in K$$

$$\sum_{a\in\delta^+(i)} X_a^k = \sum_{a\in\delta^-(i)} X_a^k, \forall k \in K, \forall i \in V$$

$$S_j \geq S_i + Pi + t_{ij}^k(1-Z_i^k) + (15 + t_{id}^k + t_{dj}^k) + Z_i^k - M(1-X_{ij}^k), \forall k \in K, \forall i \in V \quad (2.8)$$

$$\sum_{k\in R} Y_i^k = r_i, \forall i \in I$$

$$\sum_{k\in P} Y_i^k = 1, \forall i \in I$$

$$S_i \geq e_i, \forall i \in I$$

$$X_{ij}^k \leq Z_i^k + Z_j^k, \forall k \in P, \forall i, j \in I \quad (2.9)$$

Interpretación de la variante:

- En primer lugar, se produce una variación en la familia de restricciones (2.8) ya que hay que añadir los 15 minutos de descanso en caso de que sea necesario, junto el tiempo que tarda el práctico en ir a la zona de descanso (t_{id}^k) y el tiempo que tarda en ir de la zona de descanso a la siguiente operación (t_{dj}^k).

- Observamos que hay una nueva familia de restricciones, la (2.9) la cual nos dice que si el práctico k realiza las operaciones i y j, necesariamente tiene que descansar después de i o de j, siendo esta una forma de simplificar la restricción que nos exige un descanso cada dos horas.

3

Resultados computacionales

En los capítulos anteriones hemos visto los conceptos teóricos que se usan en este trabajo, las herramientas computacionales y el modelo matemático que se va a utilizar. A continuación mostraremos los resultados computacionales obtenidos y una representación de los mismos para dos instancias diferentes del puerto de Valencia, los días 17 y 30 de enero de 2023 con 17 y 33 operaciones respectivamente.

3.1. Obtención de datos

Como se ha mencionado anteriormente los datos con los que vamos a trabajar a continuación han sido facilitados por la empresa Híades Business Patterns SL, la cual tiene acceso a la información sobre todos los sucesos en tiempo real que transcurren en el puerto de Valencia, gracias a una combinación del uso de datos AIS y el producto de la empresa, AMURA PILOTS. Así podemos saber, entre otras cosas, el número de prácticos disponibles, el número de remolcadores disponibles, el número de remolcadores necesarios para una operación (r_i) y el tiempo de proceso de cada operación (p_i).

Sin embargo, hay otros parámetros del problema que hemos estimado. Así, la hora de llegada/salida de los buques (e_i), se genera aleatoriamente entre 0 y 30 minutos antes con respecto a la hora que se inició de la operación i de los datos disponibles recabados. También, el tiempo que tardan los agentes en desplazarse de una operación a la siguiente (t_{ij}^k) se calculó de forma proporcional a la distancia de desplazamiento de dichos agentes. Finalmente, se ha optado por poner un mismo peso $w_i = 1$ para todas las operaciones dado que no teníamos un parámetro que de forma directa nos dijera si un buque es de pasajeros o de mercancías (solo teníamos el nombre del buque).

3.2. Distribución del puerto de Valencia en zonas

Para facilitar el cálculo de los tiempos que tardan los agentes (t_{ij}^k) hemos dividido el puerto de Valencia en siete zonas diferenciadas y hemos calculado el tiempo que tardan los agentes en desplazarse de una zona a otra, creando así una matriz de 8 por 8, incluyendo una octava localización que sería la zona de descanso, con los valores de estos desplazamientos en minutos. Las zonas abarcarán los siguentes elementos:

- **Zona 0:** Canal de entrada.
- **Zona 1:** Muelle Norte, Muelle de Desguace, Espigón Norte.
- **Zona 2:** Muelle Levante ([1,30]).
- **Zona 3:** Muelle Levante ([30,70]), Transversal Levante, Transversal Poniente, Muelle Poniente, Muelle Espigón.
- **Zona 4:** Turia Sur, Muelle del Turia, Muelle Sur.
- **Zona 5:** Muelle Este, Muelle Príncipe Felipe ([1,30]).
- **Zona 6:** Muelle Príncipe Felipe ([30,70]), Muelle MSC, Muelle Costa.
- **Zona 7:** Zona de Descanso.

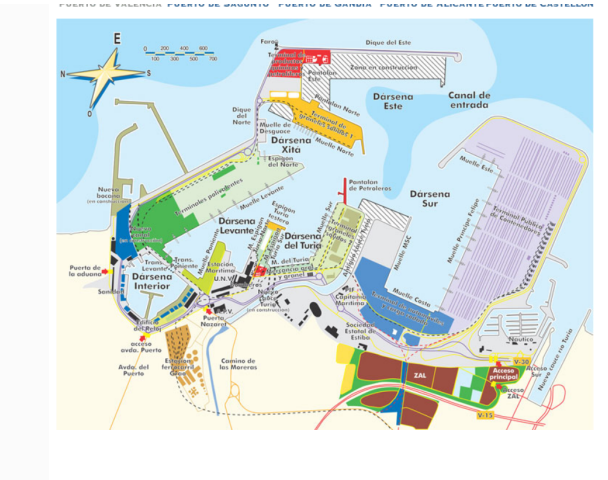

Figura 3.1: Mapa del Puerto de Valencia

3.3. Parámetros de la primera instancia

A continuación mostraremos en una tabla con los parámetros del primer ejemplo con el que trabajaremos. Para el encabezado de las columnas hemos utilizado la misma notación de la Sección 2.2.1 a la que hemos añadido zona de inicio (Zona ini.) y la zona de fin (Zona fin) de cada operación. La hora de llegada/salida e_i se mide en minutos desde la media noche y el tiempo de proceso P_i también se mide en minutos.

Op. (i)	e_i	P_i	r_i	Zona ini.	Zona fin
1	294	38	2	0	3
2	325	67	2	0	5
3	837	56	2	0	6
4	1043	58	2	0	5
5	1127	26	0	0	3
6	1151	39	1	0	2
7	1195	33	0	0	3
8	1224	25	2	0	5
9	1311	41	1	0	4
10	1327	69	2	0	1
11	871	92	2	3	4
12	950	44	2	2	1
13	682	35	2	3	0
14	1191	22	1	5	0
15	1268	25	2	6	0
16	1342	34	1	3	0
17	1370	24	1	2	0

Tabla 3.1: Datos iniciales a utilizar para el día 17 de Enero

Podemos ver que: las operaciones de la 1 a la 10 son operaciones de entrada, ya que empiezan en el canal de entrada y terminan en alguna de las zonas interiores del puerto; las operaciones 11 y 12 son movimientos internos, los cuales empiezan y terminan dentro del puerto; y, por último, las operaciones de la 13 a la 17 son operaciones de salida, que terminan fuera del puerto.

3.4. Resultados computacionales

En esta sección se muestran los distintos resultados que se obtienen al variar nuestras condiciones iniciales. Siendo estas el número de prácticos, que varía entre 2 y 3, el número de remolcadores, que varía entre 3 y 5, y el tiempo de

descanso de los prácticos, que toma los valores en conjunto $\{0, 15, 30\}$ minutos. Hay que tener en cuenta que las condiciones reales para este día fueron 3 prácticos, 5 remolcadores y descansos de 15 minutos y que el resto de instancias se ejecutaron para evaluar cómo variaba la solución en diferentes situaciones. Siendo los costos obtenidos el tiempo en minutos que deben esperar los buques para ser atendidos. Se ha omitido el caso de 2 prácticos y 5 remolcadores, pues ninguna operación de ese día requería más de 2 remolcadores y al haber tan solo 2 prácticos como máximo se podían usar 4 remolcadores.

Recordamos que P es el conjunto de los prácticos y R es el conjunto de los remolcadores. El "Costo" es el valor objetivo del problema (medido en minutos de espera total) y "Tiempo" es el tiempo computacional que tardó el algoritmo en resolverse (medido en segundos).

$\|P\|$	$\|R\|$	Descanso	Costo	Tiempo (s)
3	5	30	0	2.1
3	5	15	0	4.7
3	5	0	0	1.9
3	4	30	0	1.9
3	4	15	0	2.0
3	4	0	0	1.5
3	3	30	136	18.8
3	3	15	136	15.6
3	3	0	136	5.3
2	4	30	67	9.2
2	4	15	43	8.2
2	4	0	41	4.0
2	3	30	164	11.5
2	3	15	140	20.8
2	3	0	138	12.3

Tabla 3.2: Resultados computacionales del día 17 de Enero.

$\|P\|$	$\|R\|$	Descanso	Costo	Tiempo (s)
3	5	30	17	51.7
3	5	15	16	37.5
3	5	0	13	14.9
3	4	30	25	50.0
3	4	15	24	172.9
3	4	0	21	40.2
3	3	30	–	1800.0
3	3	15	–	1800.0
3	3	0	–	1800.0
2	4	30	–	1800.0
2	4	15	–	1800.0
2	4	0	–	1800.0
2	3	30	–	1800.0
2	3	15	–	1800.0
2	3	0	–	1800.0

Tabla 3.3: Resultados computacionales del día 30 de Enero.

A raíz de estos resultados podemos sacar algunas conclusiones sobre la Tabla 3.2:

- Para las condiciones del día trabajado, si había tres prácticos no era necesario tener 5 remolcadores, ya que con 4 se logran los mismos resultados.
- En el caso de 3 prácticos y 5 o 4 remolcadores los descansos no suponen un mayor costo.
- Con tan solo 3 remolcadores aumenta notablemente el costo independientemente si hay 2 o 3 prácticos.
- Con 2 prácticos y 4 remolcadores los costos aumentan, pero no demasiado, lo cual abre la puerta a considerar si es beneficiosa la retirada de un práctico a pesar del aumento de costo que supone.
- Por último se puede apreciar que con dos prácticos los descansos sí suponen un aumento de los costos.

Si nos fijamos ahora en la Tabla 3.3 vemos que:

- La instancia solo se resuelve en menos de 30 minutos (1800 segundos) para los casos en los que hay 3 prácticos y 4 o 5 remolcadores.

- Observamos que ni el cambio en el tiempo de descanso ni el paso de 5 remolcadores a 4 suponen una gran diferencia en el costo.

- Podemos concluir que en este día todos los prácticos eran necesarios.

3.5. Representación de los resultados:

En concreto vamos a representar los resultados para el caso de 2 prácticos, 4 remolcadores y descansos de 15 minutos:

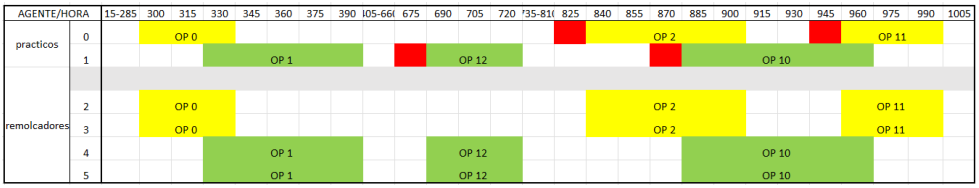

Figura 3.2: Primera parte del día 17 de Enero

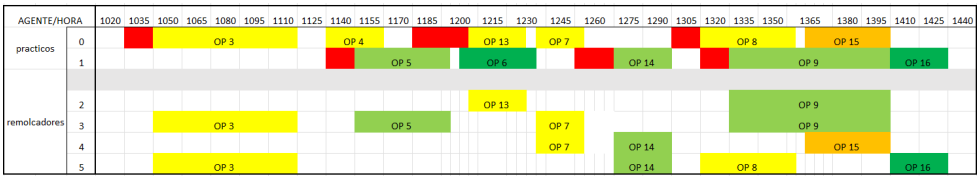

Figura 3.3: Segunda parte del día 17 de Enero

En las imágenes anteriores se puede observar cómo se han distribuido los recursos para realizar las diferentes operaciones, viéndose reflejado en las columnas los diferentes períodos de tiempo y en las filas los distintos agentes.

Las operaciones marcadas en amarillo son aquellas realizadas por el práctico 0 y aquellas marcadas en verde las realizadas por el práctico 1. Siendo los diferentes descansos que realizan los prácticos las diferentes franjas rojas y las operaciones más oscuras aquellas que tuvieron un costo distinto a 0 (el buque tuvo que esperar ya que no había recursos disponibles).

Cabe señalar que el día se ha dividido en períodos de 15 minutos y que se ha coloreado cualquier operación que comience en ese intervalo para facilitar la representación, salvo en casos en los que, debido a la proximidad de las operaciones esto habría supuesto solapamientos, en estos casos se ha detallado más

dentro de los intervalos para representar correctamente lo sucedido. A continuación mostramos en una tabla los datos equivalentes al diagrama anterior:

- Op.= número de la operación
- e_i = hora de llegada a puerto
- S_i = hora de comienzo del proceso
- P_i = duración del proceso

La Tabla 3.4 muestra los valores de las variables S_i que indica el tiempo de comienzo de cada operación, siendo el tiempo de finalización $S_i + P_i$. Observe que las únicas operaciones con tiempo de espera son las operaciones 6, 15 y 16.

Op. (i)	e_i	S_i	P_i	$S_i + P_i$
0	294	294	38	332
1	325	325	67	392
2	837	837	56	893
3	1043	1043	58	1101
4	1127	1127	26	1153
5	1151	1151	39	1190
6	1195	1197	33	1230
7	1224	1224	25	1249
8	1311	1311	41	1352
9	1327	1327	69	1396
10	871	871	92	963
11	950	950	44	994
12	682	682	35	717
13	1191	1191	22	1213
14	1268	1268	25	1293
15	1342	1355	34	1389
16	1370	1398	24	1422

Tabla 3.4: Representación de la solución para el día 17 de Enero

La Tabla (3.5) muestra los resultados referentes al día 30 de Enero para el caso de 3 prácticos, 5 remolcadores y 15 minutos de descanso cada 2 operaciones. Ahora, las únicas operaciones que tienen un retraso son la 14 y 32.

Op. (i)	e_i	S_i	P_i	$S_i + P_i$
0	266	266	54	320
1	328	328	49	377
2	430	430	40	470
3	436	436	40	476
4	486	486	79	565
5	783	783	34	817
6	1009	1009	30	1039
7	1000	1000	42	1042
8	1013	1013	54	1067
9	1076	1076	25	1101
10	1142	1142	32	1174
11	1135	1135	34	1169
12	1173	1173	55	1228
13	1236	1236	24	1260
14	1332	1335	84	1419
15	430	430	60	490
16	524	524	109	633
17	826	826	32	858
18	924	924	39	963
19	1228	1228	36	1264
20	368	368	35	403
21	903	903	23	926
22	953	953	18	971
23	1054	1054	26	1080
24	1075	1075	44	1119
25	1190	1190	20	1210
26	1259	1259	43	1302
27	1284	1284	13	1297
28	1333	1333	16	1349
29	1323	1323	11	1334
30	1364	1364	19	1383
31	1356	1356	18	1374
32	1365	1378	10	1388

Tabla 3.5: Representación de la solución para el día 30 de Enero

Conclusiones

A nivel personal, esta memoria me ha permitido comprender mejor varios conceptos estudiados durante mi carrera, así como reconocer la importancia que puede tener la programación combinatoria en el mundo real. También he aprendido cómo se pueden implementar estos conocimientos en una empresa para generar resultados valiosos e interesantes.

En el transcurso de este trabajo, hemos logrado desarrollar e implementar en Python un problema específico centrado en optimizar la entrada y salida de buques en un puerto. Utilizando los datos proporcionados por la empresa Híades Business Patterns SL, hemos sido capaces de encontrar soluciones para instancias con 17 y 33 operaciones. Además, hemos evaluado cómo varía la solución al realizar cambios en los recursos disponibles, ya sea modificando el tiempo de descanso o el número de operarios.

Es importante tener en cuenta algunas limitaciones de este trabajo. Algunos de los datos utilizados han sido estimados, lo que significa que la solución real y los costos reales podrían diferir. Esto abre la posibilidad de realizar modificaciones y profundizar aún más en esta línea de investigación. Además, la restricción relacionada con los descansos de los prácticos también presenta oportunidades de mejora. Por ejemplo, podrían darse tres operaciones cortas y consecutivas que no superen las dos horas, algo que nuestro modelo no contempla en la actualidad.

En resumen, esta memoria ha sido enriquecedora al permitirme aplicar los conocimientos teóricos aprendidos a un problema práctico y real. Aunque existen algunas limitaciones y áreas de mejora, considero que los resultados obtenidos son valiosos y pueden ser utilizados como punto de partida para futuros desarrollos en la gestión de operaciones portuarias.

A

Apéndice

A.1. Código utilizado

```
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
from ortools.linear_solver import pywraplp
```

Listing A.1: Celda de código 1

Importamos los paquetes necesarios para el desarrollo del código

```
df = pd.read_excel('C:/Users/Pablo/Documents/TFG/TFG1/TFG1/30
    _ENERO_SIMPLIFICADO.xlsx')

# PARAMETROS
gurobi = True

def pedir_parametros():
    P1 = input('ingrese el numero de practicos: ')
    P2 = input("Ingrese el numero de remolcadores: ")
    P3 = input("Ingrese el tiempo de descanso de los practicos: ")
    return P1, P2, P3
P, R, TIEMPO_DESC= pedir_parametros()
P=int(P)
R=int(R)
TIEMPO_DESC=int(TIEMPO_DESC)

op=df.shape[0] #numero de operaciones
I=range(op+2)
I1=range(op) #operaciones sin los extremos
I0=range(op+1) #operaciones sin la op inicia
w=np.ones(op+2)#pesos de las operaciones
import random
random.seed(7)
dif  = [(random.randint(0,30)) for i in I1] #tiempo de espera del buque
    i
```

```
e=np.zeros(op+2)
for i in I1:
    e[i]= df['POB'][i]-dif[i]#hora estimada de llegada/salida de la op
    i



A=P+R #numero agentes
K= range(A)
PR= range(P)
REM= range(P,R+P)
p_i= df['DIF'] #tiempo de proceso de la op i
num_loc=df.shape[0]
L= range(num_loc)
num_zonas = 7
T1= np.zeros((num_zonas+1,num_zonas+1)) #tiempo que tarda el practico
    en ir de i a j
T1=[ [1,4,7,9,7,2,4,3],
     [4,1,2,3,3,4,9,3],
     [7,2,1,2,2,4,9,2],
     [9,3,2,1,3,7,10,3],
     [7,3,2,3,1,4,7,2],
     [2,4,4,7,4,1,2,2],
     [4,9,9,10,7,2,1,3],
     [3,3,2,3,2,2,3,1]]
#tiempo que tarda la lancha de un practico en ir de la localizacion i a
    la j
desc = num_zonas
T2= np.zeros((num_zonas+1,num_zonas+1)) #tiempo que tarda remolcador en
    ir de i a j
T2= 2*T1
t0= df['zona_inicio']
tf= df['zona_final']

r_i= df['R'] #remolcaadores necesarios para la op i
```

Listing A.2: Celda de código 2

Generamos aleatoriamente los valores más difíciles de obtener para la primera aproximación y guardamos los valores necesarios a través del uso de la librería pandas, la cual permite la lectura y manipulación de datos cargados archivo xlsx

```
if gurobi :
    solver = pywraplp.Solver('TFG1', pywraplp.Solver.
    GUROBI_MIXED_INTEGER_PROGRAMMING)
else :
    solver = pywraplp.Solver('TFG1', pywraplp.Solver.
    CBC_MIXED_INTEGER_PROGRAMMING)

# Decision variables
x = { (i,j,k) : solver.BoolVar('x[%i,%i,%i]' % (i,j,k)) for i in I for
    j in I if i!=j for k in K  } #el agente k va de la op i a la op j
```

```
y = { (i,k) : solver.BoolVar('y[%i,%i]' % (i,k)) for i in I for k in K
    } #el agente k realiza la op i
s= { (i) : solver.NumVar(e[i], solver.infinity(), 's[%i]' % (i)) for i
   in I } # tiempo de comienzo de la op i
z = { (i,k) : solver.BoolVar('z[%i,%i]' % (i,k)) for i in I for k in PR
     } #el agente k realiza un descanso despues de la op i


# Objective Function
solver.Minimize(solver.Sum( (s[i]-e[i])*w[i] for i in I1 ))
```

Listing A.3: Celda de código 3

A continuación definimos las variables de decisión y la función objetivo además del solucionador a utilizar

```

#RESTRICTIONs
[ solver.Add( solver.Sum( x[op,j,k] for j in I if j!=op) ==1) for k in
   K]
[ solver.Add( solver.Sum( x[j,op+1,k] for j in I if j!=(op+1)) ==1) for
    k in K]
[ solver.Add( solver.Sum( x[i,j,k] for j in I if j!=i) == y[i,k] )
for i in I1 for k in K]
[ solver.Add( solver.Sum( x[j,i,k] for j in I if j!=i) == y[i,k] )
for i in I1 for k in K]
M=1440 #cantidad suficientemente grande
[ solver.Add( solver.Sum( y[i,k] for k in REM)== r_i[i] ) for i in I1]
[ solver.Add( solver.Sum( y[i,k] for k in PR)== 1 ) for i in I1]
[ solver.Add( x[op+1,j,k]==0) for j in I if j!=(op+1) for k in K]
[ solver.Add( x[j, op,k]==0)  for j in I if j!=op  for k in K]

#nuevas 2

[ solver.Add( x[i,j,k]<=z[i,k]+z[j,k]) for j in I1 for i in I1 if i!=j
   for k in PR]
[ solver.Add( z[i,k]<=y[i,k]) for i in I1 for k in PR]

[ solver.Add( s[j] >=  s[i]+p_i[i]+ y[i,k]*T1[tf[i]][t0[j]]+(
   TIEMPO_DESC+T1[tf[i]][desc]+T1[desc][t0[j]])*z[i,k]-M*(1-x[i,j,k]) )
    for k in PR for i in I1 for j in I1 if i!=j ]
[ solver.Add( s[j] >=  s[i]+p_i[i]+ y[i,k]*T2[tf[i]][t0[j]]-M*(1-x[i,j,
   k]) ) for k in REM for i in I1 for j in I1 if i!=j ]
```

Listing A.4: Celda de código 4

Añadimos las restricciones, nótese que si fijamos z[i,j] a cero tenemos la solución sin descansos.

```
EPS=0.0001
# we solve it
status = solver.Solve()
if status == pywraplp.Solver.OPTIMAL:
```

```
    print("Valor optimo= ", solver.Objective().Value(),"en" , solver.
    WallTime()/1000, " segundos")
    selected = [(i,j,k) for i in I1 for j in I1 if i!=j for k in K if x
    [i,j,k].solution_value() > EPS]
    print(selected)
    for i in I1:
        print(s[i].solution_value())
    for k in K:
        for i in I:
            for j in I:
                if i!=j:
                    if x[i,j,k].solution_value()>EPS:
                        print('x[',i, ',', j ,',', k, ']= ',x[i,j,k].
    solution_value(), ' ', s[i].solution_value())

    for k in K:
        for i in I:
            if y[i,k].solution_value()>EPS:
                print('y[',i, ',', k, ']= ',y[i,k].solution_value(), '
    ', s[i].solution_value())
    for k in PR:
        for i in I:
            if z[i,k].solution_value()>EPS:
                print('z[',i, ',', k, ']= ',z[i,k].solution_value() )
   # print(s) #encontrar valores de s
#    dibuja(selected)
else:
    print('The problem does not have an optimal solution.')
```

Listing A.5: Celda de código 5

Solucionamos el problema e imprimimos los resultados obtenidos

A.2. Otras opciones que se tuvieron en cuenta

```
#nuevas
[ solver.Add( s[desc[k][0]] <= 120 )   for k in PR ]
[ solver.Add( s[desc[k][i+1]] <= 120+ s[desc[k][i]])  for i in range(
    len(desc[k])-1) for k in PR ]
[ solver.Add( solver.Sum( x[i,j,k] for j in Id0 if j!=i) ==1) for i in
    desc[k] for k in PR]
[ solver.Add( solver.Sum( x[j,i,k] for j in Id0 if j!=i) ==1) for i in
    desc[k] for k in PR]

[ solver.Add( x[i,j,k]==0) for kp in PR  for k in PR for j in Id0  for
    i in desc[kp] if j!=i if k!=kp] #evitar que un agente visite un
    descanso de otro
[ solver.Add( x[j,i,k]==0) for kp in PR  for k in PR for j in Id0  for
    i in desc[kp] if j!=i if k!=kp]
```

Listing A.6: Celda de código 6

Este fragmento muestra la primera idea que se barajó para establecer las restricciones con respecto al descanso de los prácticos, pero debido a su mayor complejidad se descartó.

Bibliografía

[►]

[1] Juan José Salazar González: Programación Matemática. Editorial Díaz de Santos. Madrid, 2001.

[2] Un análisis de los factores que afectan la productividad de los sitios de atraque en una terminal de contenedores.Julio Mar-Ortiz , María Gracia D, Universidad Autónoma de Tamaulipas, Facultad de Ingeniería [Fecha de consulta:4 de marzo de 2024]. Disponible en: https://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1405-77432017000200169.

[3] Gurobi Optimization. [Fecha de consulta: 25-06-2023]. Disponible en: https://www.gurobi.com/..

[4] Python [Fecha de consulta: 15-03-2023] https://www.python.org/..

A combinatorial optimization problem in seaports

Pablo Yanes Riquelme
Facultad de Ciencias • Sección de Matemáticas
Universidad de La Laguna
alu01012406626@ull.edu.es

Abstract

Combinatorial optimization, as a branch of mathematics, focuses on finding the best solution within a large set of options. In this project, we will develop and implement a specific problem in Python to optimize ship entry and exit in a port.
Using data provided by "Híades Business Patterns SL", we will analyze ship information, available port resources, and operator rest requirements. The goal is to determine optimal solutions considering different combinations of resources and rest requirements.
Through combinatorial optimization techniques, we will assess whether the port's resources are sufficient to handle ship flow efficiently. We will also examine how varying operator rest requirements impact the optimization process.
By the end of the project, we will draw informed conclusions about the port's capacity to handle ship traffic effectively. This will help identify any necessary adjustments in resource allocation or rest requirements to achieve optimal optimization.

1. Introduction

Below, we will enumerate several key points of this work that have been addressed and explored in detail:

- Combinatorial optimization: Finding the best solution within a finite but large set of options.
- Decision-making: In the port context, it involves making decisions regarding ship entry and exit, with a need to improve efficiency and automate these processes.
- Ships, tugs, and pilots: They are the key actors in the study.
- Costs and priorities: Delays in ship entry and exit can generate significant costs.
- Development of a computational solution: The research aims to create a solution using combinatorial optimization tools to improve efficiency and decision-making in the port process.
- Implementation in the collaborating company: The goal is to provide an initial version of the product that can be applied in the collaborating company and potentially in other ports in the future.

2. The model

$$X_{ij}^k = \begin{cases} 1 \text{ if the agent } k \text{ goes from the operation } i \text{ to } j \\ 0 \text{ in other case} \end{cases}$$

$$S_i = \text{start of the operation } i$$

$$Y_i^k = \begin{cases} \text{if the agent } k \text{ made the operation } i \\ 0 \text{ in other case} \end{cases}$$

Our objective function will be the sum of the waiting times for each ship, weighted by w_i.

$$\min \sum_{i \in I} (S_i - e_i) wi \tag{1}$$

subject to:

$$\sum_{a \in \delta^+(0)} X_a^k = \sum_{a \in \delta^-(0')} X_a^k = 1 \qquad \forall k \in K \tag{2}$$

$$\sum_{a \in \delta^+(i)} X_a^k = \sum_{a \in \delta^-(i)} X_a^k \qquad \forall k \in K, \forall i \in V \tag{3}$$

$$S_j \geq S_i + Pi + t_{ij}^k - M(1 - X_{ij}^k) \qquad \forall k \in K \; \forall i \in V \tag{4}$$

$$\sum_{k \in R} Y_i^k = r_i \qquad \forall i \in I \tag{5}$$

$$\sum_{k \in P} Y_i^k = 1 \qquad \forall i \in I \tag{6}$$

$$S_i \geq e_i \qquad \forall i \in I \tag{7}$$

3. Computational results

We can visually observe the representation of one of the solutions generated by the program for the case of 2 pilots, 4 tugs, and 15-minute breaks.

Figure 1: *First part of the day* 17^{th} *of January*

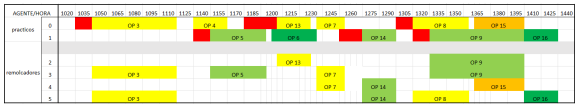

Figure 2: *Second part of the day* 17^{th} *of January*

The previous images display the distribution of resources and operations across different time intervals. Operations performed by each pilot are highlighted using colors, such as yellow for pilot 0 and green for pilot 1. The rest periods for the pilots are represented by red stripes. Operations that incurred a non-zero cost are depicted in darker shades, indicating instances where ships had to wait due to limited resource availability. In summary, these images provide a clear visual representation of resource allocation and waiting situations within the context of the ship entry and exit process at the port.