UNIVERSIDAD DE LA LAGUNA

The traveling purchaser problem

Autor: Riera Ledesma, Jorge

Director: Juan José Salazar González

Departamento de Estadística, Investigación Operativa y Computación

The Traveling Purchaser Problem

Dissertation
for the Degree of
Doctor of Philosophy
in
Computer Sciences

The Traveling Purchaser Problem

JORGE RIERA LEDESMA

supervised by

Juan José Salazar González

2002

UNIVERSIDAD DE LA LAGUNA
Departamento de Estadística, Investigación Operativa
y Computación

JUAN JOSÉ SALAZAR GONZÁLEZ, PROFESOR TITULAR DE UNIVERSIDAD DEL ÁREA DE ESTADÍSTICA E INVESTIGACIÓN OPERATIVA DE LA UNIVERSIDAD DE LA LAGUNA

CERTIFICA:

Que la presente memoria, titulada “The Traveling Purchaser Problem”, ha sido realizada bajo mi dirección por el Licenciado D. Jorge Riera Ledesma, y constituye su Tesis para optar al grado de Doctor en Informática por la Universidad de La Laguna.

Y para que conste, en cumplimiento de la legislación vigente, y a los efectos que hayan lugar, firmo la presente, en La Laguna, a 26 de septiembre de dosmil dos.

Acknowledgments

I am truly grateful to my supervisor Juan José Salazar González , who helped me to get acquainted with scientific research spending a lot of time on teaching me Polyhedral Theory. I also thank to Juan José his willingness to always help me in my permanent struggle with the computer. Thanks to his generosity the process of writing my thesis has been completed successfully.

Chapters 3, 4 and 5 are the result of a nice collaboration with Gilbert Laporte.

I would like to thank to Jaques Renaud for having kindly provided us with the source code used by Boctor, Laporte and Renaud [22].

I would like to thank my colleagues from the *Departamento de Estadística, Investigación Operativa y Computación* for the nice working enviroment.

I would like to thank to Universidad de La Laguna, DIMACS (Center for Discrete Mathematics & Theoretical Computer Science Founded as National Science Foundation Science and Technology Center), IBM Watson Research Center, EURO (Association of European Operational Research Societies), "Gobierno de Canarias" (Research Project PI2000/116), "Ministerio de Educación y Ciencia" (Research Project TIC2000-1750-C06-02), "Ministerio de Asuntos Exteriores" (Research Project approved by "XXII Comisión Mixta Permanente en Apliación del Acuerdo Cultural Entre España y Bélgica, Comunidad Francesa") and the Programme "High-Level Scientific Conferences" of the European Community for selected young researchers and post-graduate students, for their finantial suport that has made this PhD-research possible.

J. R. L.

Preface

The boundaries between Operations Research and Computer Science have become blurred. Important new theories and whole fields, like Polyhedral Combinatorics, have been and are being developed jointly by computer scientists, operations researchers, and applied mathematicians.

Placed in this scope, the aim of this disertation is to embrace some points of these topics. Thus, the Traveling Purchaser Problem (TPP) provides an excuse to go deep in Polyhedral Combinatorics and in several topics which contribute to the resolution of the optimization problems.

The grateful experience of realizing how the theoretical results leads to successful computational results by mean the computer programm has supported our hipotesis and verified that our research results become very applied. The preponderant role played by the engineer in Computer Sciences in this area makes this research to finish the travel starting at the mathematical model and finishing at the computational resolution of the problem.

Sumarizing, the aim of this thesis is to carry out a thorough study on the TPP which lead to the development of several algorithms with a further computational evaluation. These algorithms solve the TPP in both exact and approximated ways. To achieve this goal we have carried out

i) an exhaustive study on the previous works, taking into account every single previous algorithm. Each technique involved in those algorithm has been also studied testing the quality of the results, and compiling benchmark instances.

ii) a study on the real world applications of the TPP as well as the scope where it is included.

iii) the development of mathematical models, which will be part of our exact approach, and of valid inequalities allowing to strengh the linear programming relaxation of the above mentioned models.

iv) the development of exact and heuristic algorithms for different variations of the TPP, based on the theoretical results computationally evaluated on intances proposed by the previous works and additional families of random instances.

A preliminary version of an exact algorithm for the TPP was submitted to the *European Chapter on Combinatorial Optimization* (ECCO) XII which took place in Bendol Island, Marselle. This branch-and-cut approach was improved, and the new results were exposed in the *Workshop on Discrete Mathematics 99* (DO'99) given in RUTCOR, Rutger, New Jersey. A worthy contribution of this conference was an interesting discussion with the Professor Peter Brucker about the classification of the TPP as a job scheduling problem. Finally, this article was submitted to *Operations Research* and currently is under revision.

A study on the bicriterion TPP was submitted to the conference *Congreso sobre Técnicas de Ayuda a la Decisión en la Defensa* wich took place in Madrid. In IRIT Laboratory in Toulouse, during the XIX *EURO Summer Institute* was also presented an algorithm for the bicriterion TPP, as well as a new and useful technique to speed up the computational time computing the non-dominated points. This works was submitted to *European Journal of Operational Research* and currently is under revision.

Because of its computational complexity, most of the research on TPP has been directed towards the development of heuristic approaches. Despite of this, we have develop a heuristic approach for the TPP which is able to solve this problem with more precision and faster than the previous approaches. Preliminary results were presented in the ECCO XIV in Bonn, and the article was submitted to *European Journal of Operational Research*.

Finally, the only aspect we had not approached was the asymmetric version of the TPP. Only one previous article had focused it. Thus, a more efficient exact algorithm was presented in the *Combinatorial Optimization* 2002 in Paris, and the related article has been recently submitted to *Discrete Applied Mathematics*.

Those results obtained in this thesis were sumarized and exposed during the seminar *The Travelling Salesman Problem* in the International Conferece and Research Center for Computer Science in *Dagstuhl*, Germany, and in the seminar *Mathematical Methods in Manufacturing and Logistic (Mathematisches Forschungsinstitut Oberwolfach)* in Oberwolfach-Walke, also in Germany.

These four articles have been treated in eigth chapters, and the obtained results have been sumarized in the chapter of Conclusions. The first chapter introduces the basic concepts underlying in the development of the proposed algorithms. These topics are Graph Theory, Computational Complexity, Polyhedral Theory, Polyhedral Combinatorics and Multicriterion Optimization. In addition, some basic optimization problems which take part of our algorithms are also described in this chapter. Chapter

2 provides an introduction to the TPP which also includes a literature review and a description of related problems in order to place the TPP in its scope. Additionally, a transformation of the TPP into the Generalized Traveling Salesman Problem is also described in this chapter. The next four chapters are devoted to the different aspects of the exact algorithms developed for the directed and undirected versions of the TPP. Chapter 3, provides the mathematical formulation of both the directed and undirected versions of the TPP. Chapter 4 gives and polyhedral analysis for the TPP, and as in the previous chapter, for both cases. And Chapters 5 and 6 described the specific characteristics of each of the two exact algorithm. The Bicriterion TPP is approached in Chapter 7 of this thesis, and a heuristic approach for the TPP is given in Chapter 8.

Contents

1

Mathematical Background

This chapter is devoted to the introduction of different concepts from Graph Theory, Computational Complexity, Polyhedral Theory and Multicriterion Optimization, that will be used later on. These concepts are the basis of the theoretic background that underlies behind the algorithms we introduce in this thesis. However, this is only a brief overview on each area so we also refer to related bibliography in each section for a deeper treatment of these topics. In addition, a description of all those problems mentioned as subproblems during the development of our algorithms, is also given in the last section of this chapter.

1.1 GRAPH THEORY

Graph Theory is a way of representing and analyzing mathematical problems. We refer the reader not familiar with graphs to the textbooks of Berge [20], Christofides [30], and Bondy and Murty [23]. Many combinatorial optimization problems can be formulated as problems in graphs, which is the case of the problem analyzed in this thesis. In this section we give a brief review of some elementary concepts and properties on Graphs Theory.

A (*undirected*) *graph* $G = (V, E)$ is a pair of sets, where $V := \{v_1, \ldots, v_n\}$ is a finite and non-empty set and $E = \{e_1, \ldots, e_m\}$ is a set of pairs of elements of V. The elements of V are called *vertices*, and the elements of E are called *edges* and are represented by $e_k = [v_i, v_j]$, where $v_i, v_j \in V$. We will consider only *simple* graphs, i.e., graphs containing at most one edge linking each pair of vertices.

Given the graph $G = (V, E)$ and $S \subset V$, the edge set

$$\delta(S) := \{[v_i, v_j] \in E : v_i \in S, v_j \in V \setminus S\}$$

is called the *cut* induced by S. We write $\delta_G(S)$ to make clear (in order to avoid possible ambiguities) with respect to which graph the cut induced by S is considered. We will write $\delta(v)$ instead of $\delta(\{v\})$. The *degree* of a vertex v is the cardinality of $\delta(v)$. The set

$$E(S) := \{[v_i, v_j] \in E : v_i, v_j \in S, i < j\}$$

is the set of edges having both end vertices in S. We denote by $G(S) = (S, E(S))$ the subgraph induced by edges having both end vertices in S. If $E(S)$ is empty, S is an *independent set*.

The *complementary graph* of G, denoted by $\overline{G}$, is the simple graph with the same vertex set as G, and with edges all pair $[v_i, v_j]$ of vertices which are not in E. A graph $G = (V, E)$ is said to be *complete* if it contains edge $[v_i, v_j] \in E$ for all vertices $v_i, v_j \in V$. We denote the complete graph of n vertices by $K_n = (V_n, E_n)$ and assume unless otherwise stated that $V_n = \{v_1, v_2, \ldots, v_n\}$.

Two graphs $G' = (V', E')$ and $G'' = (V'', E'')$ are *isomorphic* if there exists a bijective mapping $f : V' \to V''$ such that $[v_i, v_j] \in E'$ if and only if $[f(v_i), f(v_j)] \in E''$.

A graph $G = (V, E)$ is called *bipartite* if its vertex set V can be partitioned into two nonempty disjoint sets V_1, V_2 with $V_1 \cup V_2 = V$ such that no two vertices in V_1 and no two vertices in V_2 are connected by an edge. If $|V_1| = n_1, |V_2| = n_2$ and $E = \{[v_i, v_j] : v_i \in V_1, v_j \in V_2\}$ then we call G the *complete bipartite graph* K_{n_1,n_2}.

An edge set $P = \{[v_1, v_2], [v_2, v_3], \ldots, [v_{k-1}, v_k]\}$ is called a *walk* between v_1 and v_k. The vertices v_1 and v_k are the *starting point* and the *end point* of the walk, respectively, or just the *end points*. If $v_i \neq v_j$ for all $i \neq j$ then P is called *path*. The *length* of a walk (or path) is the number of its edges and is denoted by $|P|$. If $v_1 = v_k$ in a walk we speak of a *closed walk*.

A walk $C = \{[v_1, v_2], \ldots, [v_{k-1}, v_k], [v_k, v_1]\}$ with $v_i \neq v_j$ for all $i \neq j$ is called a *cycle* (or k-cycle), also named simple cycle. An edge $[v_i, v_j]$, $1 \leq i \neq j \leq k$, not in C is called *chord* of C. The length of a cycle C is denoted by $|C|$. For convenience we shall sometimes abbreviate the cycle $\{[v_1, v_2], \ldots, [v_{k-1}, v_k], [v_k, v_1]\}$ by $(v_1, \ldots, v_k)$ and also say that a graph G is a cycle if its edge set forms cycle. A graph or edge set is called *acyclic* if it contains no cycle. An acyclic graph is also called a *forest*.

A graph $G = (V, E)$ is said to be *connected* if it contains a path for every pair of vertices; otherwise G is called *disconnected*. Concreteness by path induces a equivalence relation on the vertices. Its classes are called the *(connected) components* of the graph. A *tree* is a connected forest containing all vertices of the graph. It is not difficult to see that the following are equivalent for a given simple graph $G = (V, E)$:

(i) G is a tree;

(ii) G contains no circuits and $|E| = |V| - 1$;

(iii) G is connected and $|E| = |V| - 1$;

(iv) any two vertices of G are connected by exactly one simple path.

If we add one new edge connecting two vertices of the tree, we obtain a graph with a unique circuit. Each tree with at least two vertices has some vertices of degree one, which are called leaf of the tree.

A subgraph $G' = (V', E')$ of $G = (V, E)$ is a *spanning (sub)tree* of G if $V' = V$ and G' is a tree. Then G has a spanning subtree if and only if G is connected. A *maximal* forest in $G = (V, E)$ is a subgraph (V, E') which is a forest, where E' not contained in the edges of a larger fortes. This implies that (V, E') has the same components as (V, E).

Sometimes it is useful to associate a direction with the edges of a graph. A *directed graph* (or *digraph*) $D = (V, A)$ consists of a finite set of *vertices* V and a set of *arcs* $A \subseteq V \times V \setminus \{(v, v) : v \in V\}$ (we do not consider loops or multiple arcs). If $e = (v_i, v_j)$ is an arc of D with end vertices v_i and v_j then we call v_i its *tail* and v_j its *head*. The arc e is said to be *directed* or *incident from* v_i *to* v_j. The number of arcs incident to a vertex v is called *indegree* of v and the number of arcs incident from v is called the *outdegree* of v. The *degree* of v is the sum of its indegree and outdegree. For a vertex v the sets of arcs incident from v, incident to v, and incident from or to v are denoted by $\delta^+(v)$, $\delta^-(v)$, $\delta(v)$, respectively. Two vertices are *adjacent* if there is an arc connecting them.

Most of the definitions for undirected graphs carry over in a straightforward way to directed graphs. For example, *diwalks*, *dipaths*, and *dicycles* are defined analogously to walks, path, and cycles with the additional requirement that the arcs are directed in the same orientation.

A digraph $D = (V, A)$ is said to be *complete* if for all $v_i, v_j \in V$ it contains both arcs (v_i, v_j) and (v_j, v_i). We denote the complete digraph on n nodes by $D_n = (V_n, A_n)$. For each graph $D = (V, A)$ we can construct its *underlying graph* $G = (V, E)$ by setting $E = \{[v_i, v_j] : v_i \text{ and } v_j \text{ are adjacent in } D\}$.

A walk (diwalk) that traverses every edge (arc) of a graph (digraph) exactly once is called *Eulerian trail* (*Eulerian ditrail*). If such a walk (diwalk) is closed we speak of a *Eulerian tour*. A graph (digraph) is *Eulerian* if its edge (arc) set can be traversed by a Eulerian tour.

A cycle (dicycle) of length n in a graph (digraph) on n nodes is called *Hamiltonian cycle* (*Hamiltonian dicycle*) or *Hamiltonian tour*. A path (dipath) of length n is called *Hamiltonian path* (*Hamiltonian dipath*). A graph (digraph) containing a Hamiltonian tour is called *Hamiltonian*.

Often we have to deal with graphs where a rational number (edge weight) is associated with each edge. We call a function $c : E \to \mathbb{Q}$ (where $\mathbb{Q}$ denotes a set of rational numbers) a *weight function* defining a weight c_e for every edge $e = [v_i, v_j] \in E$. The weight of a set of edges $F \subseteq E$ is defined as

$$c(F) := \sum_{e \in F} c_e.$$

The weight of a tour is usually called its *length*. A tour of smallest weight is called *shortest tour*.

1.2 COMPUTATIONAL COMPLEXITY

The main purpose of the Complexity Theory is to determine how difficult a problem may be to solve. This involves not only to establish a complexity classification according to time or space complexity of its best exact algorithm, but also to seek the different types of performance guarantees that are possible in a heuristic approach. Some concepts related to Complexity Theory are defined briefly in this section. For precise discussions we refer to the books by Aho, Hopcroft and Ulman [1] and Garey and Johnson [59].

Ground objects when formalizing problem complexity are symbols and string of symbols. Let Σ be a finite set called the *alphabet*. The elements of Σ are called *symbols* or *letters*. An ordered finite sequence of symbols from Σ is called a *string* or a *word*. Σ^* stands for the collection of all strings of symbols from Σ. The *size* of a string is the numbers of its components. The string of size 0 is the *empty string*, denoted by ϵ.

A string can have the form of rational number, vectors, matrices, graphs, linear equations or inequalities, and so on. There are some standard ways of transformations to encode these objects uniformly as proper string of symbols from some fixed alphabet like $\{0, 1\}$. Depending on the chosen transformation, this induces a concept of size of these objects.

A *problem* will be a general question to be answered, usually processing several *parameters*, or free variables, whose values are left unspecified. A problem is described by giving a general description of its parameters, and a statement of what properties the answer, (or *solution*) must satisfy. An *instance* of a problem is obtained by specifying particular values for all problem parameters. Formally, a problem is a subset Π of $\Sigma^* \times \Sigma^*$, where Σ is some alphabet. The corresponding mathematical problem the is:

> given a string $z \in \Sigma^*$, find a string y such that $(z, y) \in \Pi$, or decide that no such string y exist.

Here the string z is called an *instance* or the *input* of the problem, and y is a *solution* or *output*.

A problem Π is called a *decision problem* or a *yes/no problem* if, for each (z, y) in Π, has only two possible values, y is ϵ the empty string. In that case, the problem is often identified with the set L (called the *language of the problem*) of strings z in Σ^* for which (z, ϵ) belongs to Π. The problem is to decide whether z belongs to L.

An *algorithm* is a list of instructions to solve a problem, and it can be formalized in terms of a *Turing Machine* (see Turing [149] and Aho, Hopcroft and Ullman [1]).

For a given input $z \in \Sigma^*$, an algorithm for problem $\Pi \subseteq \Sigma^* \times \Sigma^*$ determines an output y such that (z, y) is in Π, or stops without delivering an output if there exists no such y. One says that an algorithm A solves a problem Π, or A is an algorithm

for Π, if for any instance z of Π, when giving the string (A, z) to a 'universal Turing machine', the machine stops after a finite number of steps, while delivering a string y with $(z, y) \in \Pi$, or delivering no string in the case where such a string y does not exist.

The *running time* of an algorithm A for certain problem instance z can be defined as the number of moves the 'head' that a universal Turing Machine makes before stopping, when it is given the algorithm A and the input z. We define the *running time function* of an algorithm A as the function $f : \mathbb{Z}^+ \to \mathbb{Z}^+$ with

$$f(n) := \max_{z, \text{size}(z) \leq n} (\text{running time of } A \text{ for input } z) \text{ for } n \in \mathbb{Z}^+.$$

If $f, g_1, \ldots, g_m$ are real-valued functions, then f is said to be *polynomially bounded by* $g_1, \ldots, g_m$ if there is a function ϕ such that $\phi \geq f$ and such that ϕ arises by sequence of compositions from the functions $g_1, \ldots, g_m$ and from some polynomials.

In the special case that $g_1, \ldots, g_m$ are polynomials, it follows that when f is polynomially bounded by $g_1, \ldots, g_m$ then f is bounded above by a polynomial. In that case, f is called a *polynomially bounded* function.

An algorithm is called *polynomial-time* (or simply *polynomial*) if its running time function is polynomially bounded. A problem is said to be *solvable in polynomial time* or *polynomially solvable* if the problem can be solved by a polynomial-time algorithm. We are interested mostly in the asymptotic behaviour of the running time of the algorithm. Therefore, one often says that the running time is $O(g(n))$, for some function $g(n)$, meaning that there is a constant C such that the running time is upper bounded by $Cg(n)$.

The class of decision problems solvable in polynomial time is denoted by $\mathcal{P}$. Another, possibly larger, complexity class is the class $\mathcal{NP}$. Informally, the class $\mathcal{NP}$ can be described as the class of those decision problems satisfying:

> for any $z \in L$, the fact that z is in L has a proof of length polynomially bounded by the size of z.

More formally, a decision problem $L \subseteq \Sigma^*$ belongs to $\mathcal{NP}$ if there exist a polynomially solvable decision problem $L \subseteq \Sigma^* \times \Sigma^*$ and a polynomial ϕ such that for each z in Σ^*:

$$z \in L \leftrightarrow \exists y \in \Sigma^* : (z, y) \in L' \text{ and } \text{size}(y) \leq \phi(\text{size}(z)).$$

As an interpretation, y here fulfills the role of a polynomial length proof of the fact that z is in L. This proof can be checked in polynomial time, as L' is polynomially solvable. The crucial point is that it is not required that y must be found in polynomial time.

The *complement* of a decision problem $L \subseteq \Sigma^*$ is the decision problem $\Sigma^* \setminus L$. The class of decision problems L whose complement is in $\mathcal{NP}$ is denoted by co-$\mathcal{NP}$. So co-$\mathcal{NP}$ consist of those decision problems L for which the fact that a certain string z is not in L has a proof of length polynomially bounded by size(z). Since the complement of every polynomially solvable decision problem is trivially polynomially solvable again, we know that $\mathcal{P} \subseteq$co-$\mathcal{NP}$, and hence $\mathcal{P} \subseteq \mathcal{NP} \cap$co-$\mathcal{NP}$.

The class $\mathcal{NP}\cap$co-$\mathcal{NP}$ consists of those decision problems for which both a positive answer and a negative answer have a proof of polynomial length. That is, it consists of all problem $L \subseteq \Sigma^*$ for which there exist polynomially solvable decision problem L',L'' and a polynomial ϕ, such that for each string $z \in \Sigma^*$:

$$z \in L \leftrightarrow (z,x) \in L' \text{ for some string } x \text{ with size}(x) \leq \phi(\text{size}(z))$$

$$z \notin L \leftrightarrow (z,y) \in L'' \text{ for some string } y \text{ with size}(y) \leq \phi(\text{size}(z))$$

A problem is *well-characterized* if it belongs to $\mathcal{NP}\cap$co-$\mathcal{NP}$. To any well-characterized problem there corresponds a *good characterization*, which is the theorem asserting that, in the above notation:

$$\exists x : (z,x) \in L' \text{if and only if } \forall y : (z,y) \notin L'',$$

where L' and L'' satisfy, for a certain polynomial ϕ:

$$\text{if}(z,x) \in L' \text{ then } (z,x') \in L' \text{ for some string } x' \text{ with size}(x') \leq \phi(\text{size}(z))$$

$$\text{if}(z,y) \in L'' \text{ then } (z,y') \in L'' \text{ for some string } y' \text{ with size}(y') \leq \phi(\text{size}(z)).$$

It is known that certain problems in the class $\mathcal{NP}$ are hardest among all problems in $\mathcal{NP}$, under a certain ordering of the problems by difficulty.

A *polynomial transformation* from a language $L_1 \subseteq \Sigma_1^*$ to a language $L_2 \subseteq \Sigma_2^*$ is a function $f : \Sigma_1^* \to \Sigma_2^*$ that satisfies the following two conditions:

(i) There is a polynomial time transformation that computes f.

(ii) For all $x \in \Sigma_1^*, x \in L_1$ if and only if $f(x) \in L_2$.

If there is a polynomial transformation from L_1 to L_2 we write $L_1 \propto L_2$. Trivially, if L_2 is polynomially solvable, and L_1 is reducible to L_2, then also L_1 is polynomially solvable. Similarly, if L_2 belongs to $\mathcal{NP}$, and L_1 is reducible to L_2, then also L belong to $\mathcal{NP}$. The same applies to co-$\mathcal{NP}$.

A problem L is called $\mathcal{NP}$-complete if it is in $\mathcal{NP}$ and each problem in $\mathcal{NP}$ is reducible to L. So if any $\mathcal{NP}$-complete problem is polynomially solvable, then all problems in $\mathcal{NP}$ are polynomially solvable, and hence $\mathcal{P} = \mathcal{NP}$. Similarly, if any $\mathcal{NP}$-complete problem has a good characterization, then $\mathcal{NP}$=co-$\mathcal{NP}$ =$\mathcal{NP}\cap$ co-$\mathcal{NP}$.

Note that if L is reducible to problem $L' \in \mathcal{NP}$, and L is $\mathcal{NP}$-complete, then also L' is $\mathcal{NP}$-complete.

1.3 POLYHEDRAL THEORY

In this section we summarize some concepts and results from Linear Algebra and Polyhedral Theory which are necessary for our dissertation. However, a detailed treatment of the Theory of Polyhedra is presented in Bachem and Groötschel [7],

Grünbaum [82], Rockafellar [132], Stoer and Witzgall [144] and Pulleyblank [125], as well as in some books on Integer Linear Programming as Schrijver [138] and Nemhauser and Wolsey [114].

Definition 1.1. A vector $x \in \mathbb{R}^n$ is called a *linear combination* of the vectors $x_1, \ldots, x_k$ if $x = \lambda_1 x_1 + \ldots + \lambda_k x_k$ with $x_1, \ldots, x_k \in \mathbb{R}^n$ and $\lambda_1, \ldots \lambda_k \in \mathbb{R}$.

Definition 1.2. If in addition the λ_i satisfy $\lambda_1 + \ldots + \lambda_k = 1$, then x is called an *affine combination* of vectors $x_1, \ldots, x_k$. And if $x = \lambda_1 x_1 + \ldots + \lambda_k x_k$ is an affine combination such that $\lambda_i \geq 0$ for $i = 1, \ldots, k$, then x is called a *convex combination* of the vectors $x_1, \ldots, x_k$.

Definition 1.3. If $\varnothing \neq S \subseteq \mathbb{R}^n$, then the set of all linear (affine, convex) combinations of finitely many vectors in S is called the *linear (affine, convex) hull* of S and it is denoted by lin(S) (aff(S), conv(S)); by convention lin($\varnothing$)=$\{0\}$, aff($\varnothing$)= conv($\varnothing$)=$\varnothing$.

Definition 1.4. A set $S \subseteq \mathbb{R}^n$ with S=lin(S) (S=aff(S), S=conv(S)) is called a *linear subspace* (affine subspace, convex set).

It can be shown that a set $L \subset \mathbb{R}^n$ is a linear (affine) subspace if and only if there is an (m, n)-matrix A (an (m, n)-matrix A and a vector $b \in \mathbb{R}^m$) such that $L = \{x \in \mathbb{R}^n : Ax = 0\}$ ($L = \{x \in \mathbb{R}^n : Ax = b\}$). Affine subspaces of particular interest are *hyperplanes*, i.e. sets of the form $\{x \in \mathbb{R}^n : a^T x = a_0\}$ where $a \in \mathbb{R}^n \setminus \{0\}$ and $a_o \in \mathbb{R}$. Clearly, every affine subspace different from $\mathbb{R}^n$ is the intersection of hyperplanes.

Definition 1.5. A nonempty set $S \subseteq \mathbb{R}^n$ is called *linearly (affinely) independent*, if for every finite set $\{x_1, x_2, \ldots, x_k\} \subseteq S$, the equations $\lambda_1 x_1 + \ldots + \lambda_k x_k = 0$ ($\lambda_1 x_1 + \ldots + \lambda_k x_k = 0$ and $\lambda_1 + \ldots + \lambda_k = 1$) imply $\lambda_i = 0$, $i = 1, \ldots, k$; otherwise S is called *linearly (affinely) dependent*.

Every linearly (affinely) independent set in $\mathbb{R}^n$ contains at most n ($n+1$) elements. Moreover, for sets S with at least two elements, linear (affine) independence means that no $x \in S$ can be represented as a linear (affine) combination of the vector in $S \setminus \{x\}$. All sets $\{x\}, x \neq 0$, are affinely and linearly independent, $\{0\}$ is linearly dependent but affinely independent. By convention, the empty set is linearly and affinely independent.

Definition 1.6. The *rank (affine rank)* of set $S \in \mathbb{R}^n$ is the cardinality of the largest linearly (affinely) independent subset of S, and the *dimension* of S, denoted by dim(S), is the affine rank of S minus one.

Definition 1.7. A set $S \subseteq \mathbb{R}^n$ is called *full dimensional* if $\dim(S) = n$; this is equivalent to say that there is no hyperplane containing S.

It is clear from the definition that the affine rank of a set is equal to the affine rank of its affine hull. Moreover, if $0 \notin \text{aff}(S)$, i.e. if S is contained in a hyperplane $\{x : a^T x = a_0\}$ with $a_0 \neq 0$, then dim(S) is the maximum cardinality of a linearly independent set in S minus one.

Definition 1.8. The maximum number of linearly independent rows or columns of a matrix A is the *rank* and is denoted by rank(A).

Definition 1.9. An (m, n)-matrix is said to have *full rank* if its rank is equal to $\min\{m, n\}$.

Definition 1.10. A set $H \subset \mathbb{R}^n$ is called a *halfspace* if there is a vector $a \in \mathbb{R}^n$ and a scalar $a_0 \in \mathbb{R}$ such that $H = \{x \in \mathbb{R}^n : a^T x \leq a_0\}$. It is said that H is the halfspace defined by the inequality $a^T x \leq a_0$, and it is also said that (if $a \neq 0$) the hyperplane $\{x : a^T x \leq a_0\}$ is the hyperplane defined by $a^T x \leq a_0$.

Definition 1.11. An inequality $a^T x \leq b$ is called *valid* with respect to $S \subseteq \mathbb{R}^n$ if $S \subseteq \{x \in \mathbb{R}^n : a^T x \leq b\}$, i.e. if S is contained in the halfspace defined by $a^T x \leq b$.

Definition 1.12. A valid inequality $a^T x \leq b$ for S is called *supporting* if $S \cap \{x \in \mathbb{R}^n : a^T x = b\} \neq \varnothing$.

Definition 1.13. An inequality $a^T x \leq b$ valid with respect to S is called a *proper valid inequality* if S is not contained in the hyperplane $\{x \in \mathbb{R}^n : a^T x = b\}$.

Definition 1.14. A valid inequality for S which is not proper is called an *implicit equation* for S.

Definition 1.15. A *polyhedron* is the intersection of finitely many halfspaces, i.e. every polyhedron P can be represented in the form $P = \{x \in \mathbb{R}^n : Ax \leq b\}$.

Since an equation system $Dx = c$ can be written as $Dx \leq c$, $-Dx \leq -c$, every set of the form $\{x \in \mathbb{R}^n : Ax \leq b, Dx = c\}$ is a polyhedron.

Definition 1.16. A bounded polyhedron (i.e. a polyhedron P with $P \subseteq \{x \in \mathbb{R}^n : ||x|| \leq B\}$ for some $B > 0$ where $||x||$ is, for example, the Euclidean norm of x) is called *polytope*. Polytopes are precisely those sets in $\mathbb{R}^n$ which are the convex hulls of finitely many points, i.e. every polytope P can be written as $P =$conv(X) for a finite set $X \subseteq \mathbb{R}^n$.

Definition 1.17. Let us define a *face* as a subset F of a polyhedron P such that there exist an inequality $a^T x \leq a_0$ valid with respect to P and $F = \{x \in P : a^T x \leq a_0\}$. Thus, we say that inequality $a^T x \leq a_0$ defines F.

Definition 1.18. A face F is called *proper* if $F \neq P$ and $F \neq \varnothing$.

In fact, if $P = \{x \in \mathbb{R}^n : a_i^T x \leq b_i, i = 1, \ldots, k\}$ is a polyhedron and F is a face of P, then it can be showed that there exists an index set $I \subseteq \{1, \ldots, k\}$ such that $F = \{x \in P : a_i^T x \leq b_i, i \in I\}$. Similarly, if $P =$conv(X) for a finite set $X \subseteq \mathbb{R}^n$ and if F is a face of the polytope P, then there exist a set $W \subseteq V$ with $F =$conv(W).

Definition 1.19. It is said that two valid inequalities $a_i^T x \leq a_0$ and $b_i^T x \leq b_0$ for a polyhedron P are *equivalent* with respect to P if $\{x \in P : a_i^T x \leq a_0\} = \{x \in P : b_i^T x \leq b_0\}$ (i.e. both inequalities 'defines' or 'induce' the same face).

Definition 1.20. A face which contains one element only is called a *vertex*. If $\{x\}$ is a vertex of P we shall simply say that x is a vertex or extreme point of P.

Definition 1.21. A *facet* F of a polyhedron P is a proper, nonempty face (i.e. a face satisfying $\varnothing \neq F \neq P$) which is maximal with respect to set inclusion.

In Combinatorial Optimization, a polyhedron is usually given by an inequality system. However, one wants to find inequality systems with as few inequalities as possible. For this reason facet-defining inequalities are of particular importance. The following theorem provides the two basic methods to prove that a given inequality $a^T x \leq a_0$ defines a facet for a polyhedron P.

Theorem 1.1. *Let $P \subseteq \mathbb{R}^n$ be a polyhedron and assume that A is an (m, n)-matrix, $b \in \mathbb{R}^n$ such that aff(P)$= \{x \in \mathbb{R}^n : Ax = b\}$. Let F be a nonempty face of P, then the following statements are equivalent:*

(a) F is a facet of P.

(b) F is a maximal proper face of P.

(c) dim(F)=dim(P)-1.

(d) There exists an inequality $a^T x \leq a_0$ valid with respect to P with the following three properties:

(d_1) $F \subseteq \{x \in P : a^T x = a_0\}$.

(d_2) There exists $\bar{x} \in P$ with $a^T x < a_0$, i.e. the inequality is proper.

(d_3) If any other inequality $c^T x \leq c_0$ valid with respect to P satisfies $F \subseteq \{x \in P : c^T x = c_0\}$, then there exists an scalar $\alpha \geq 0$ and a vector $\lambda \in \mathbb{R}^m$ such that

$$c^T = \alpha a^T + \lambda^T A,$$
$$c_0 = \alpha a_0 + \lambda^T b.$$

Conditions (c) and (d) provide the two basic methods to prove that a given inequality $a^T x \leq a_0$ defines a facet of a polyhedron P (see Chapter 4 for several examples of proofs of facets). In both cases it has to be checked that $a^T x \leq a_0$ is valid for P and that P is not contained in $\{x \in P : a^T x = a_0\}$. This is usually trivial.

The first method consist of exhibiting a set of $k =$dim(P) vectors (usually vertices of P) $x_1, \ldots, x_k \in P$ satisfying $a^T x_i = a_0$, $i = 1, \ldots, k$, and showing that these vectors are affinely independent. (If $c_0 \neq 0$ this is equivalent to showing that these k vectors are linearly independent.) Let us call this method the *direct method*.

In most cases the second *indirect method* based on condition (d) of Theorem 1.1, is more suitable, and it is as follows. One assumes the existence of a valid inequality $c^T x \leq c_0$ with $\{x \in P : a^T x = a_0\} \subseteq \{x \in P : c^T x = c_0\}$. Using the known equation systems $Ax = b$ for P, one can determine a vector $\lambda \in \mathbb{R}^m$ such that $\bar{c} := c + A^T \lambda$ has certain useful properties, i.e. some of the coefficients of $\bar{d}$ are equal to the corresponding coefficients of the given c.

Then using known properties of the points x in P satisfying $a^T x = a_0$, one determines the still unknown coefficients of $\bar{c}$ iteratively. If it turns out that $\bar{c} = \alpha c + A^T \mu$ for some $\alpha \geq 0$ and $\mu \in \mathbb{R}^m$, then condition (d) of Theorem 1.1 implies that $a^T x \leq a_0$ defines a facet of P.

Facets are of importance since they have to be known in order to obtain a minimal inequality representation of a polyhedron.

Definition 1.22. Let $P \neq \mathbb{R}^n$ be a polyhedron. Then a system of equations and inequalities $Dx = c, Ax \leq b$ is said to be *complete* with respect of P if $P = \{x \in \mathbb{R}^n : Dx = c, Ax \leq b\}$.

Definition 1.23. Let us call a system $Ax \leq b$ *non-redundant* if it contains no implicit equations and if the deletion of any equation or inequality of the system results in a polyhedron different from P. Any equation or inequality which can be deleted without changing the polyhedron is called *redundant*.

Theorem 1.2. *Let $P \subseteq \mathbb{R}^n$ be a polyhedron and $Ax \leq b, Dx = c$ be a complete and non-redundant system for P, where D is an (m, n)-matrix and A is a (k, n)-matrix. Then the following hold:*

(a) aff(P)= $\{x \in \mathbb{R}^n : Dx = c\}$ and m =rank(D).

(b) aff(P) and P have dimension $n - m$.

(c) Every inequality $a_i^T x \leq b_i$ of the system $Ax \leq b$ defines a facet F_i of P, where $F_i = \{x \in P : a_i^T x = b_i\}$, $i = 1, \ldots, k$.

(d) if $\bar{a}_i^T x \leq \bar{b}_i$, $i = 1, \ldots, \bar{k}$; $\bar{d}_i^T x \leq \bar{c}_i$, $i = 1, \ldots, \bar{m}$; is any other complete and non-redundant system for P, then

(d_1) $k = \bar{k}, m = \bar{m}$,

(d_2) $\bar{d}_i^T = (\lambda^i)^T D$ for some $\lambda^i \in \mathbb{R}^m - \{0\} (i = 1, \ldots, m)$

(d_3) $\bar{a}_i^T = \alpha_i a_j^T) + (\lambda^i)^T D$ for some $\alpha_i > 0, \lambda^i \in \mathbb{R}^m$, and $j \in \{1, \ldots, k\}$ $(i = 1, \ldots, \bar{k})$

Theorem 1.2(d) implies that for a full-dimensional polyhedron P there is a complete non-redundant inequality system $a_i^T x \leq b_i, i = 1, \ldots, k$, such that every complete and non-redundant inequality system $\bar{a}_i^T x \leq \bar{b}_i, i = 1, \ldots, \bar{k}$, satisfy $k = \bar{k}$ and $\bar{a}_i = \alpha_i a_i$ for some $\alpha_i > 0$ and $i = 1, \ldots, k$. This justifies the statement that a full-dimensional polyhedron is defined by a *unique* non-redundant and complete inequality system. Moreover, for every facet F of P there is a unique inequality defining F.

Sequential Lifting

We shall now introduce a technique, called *sequential lifting*, which leads to new facet-defining inequalities for a polyhedron from a known facet-defining inequality of a face.

The following theorem has been extracted from Padberg [120]. This is the method in which our theorems for obtaining facet defining are mainly based on. Let us consider a convex polytope in $\mathbb{R}^n$ given by

$$P = \{x \in \mathbb{R}^n : Ax \leq b, 0 \leq x \leq 1\}$$

where $A \in \mathbb{R}^{n \times m}$ with $A \geq 0$ and integer, and in which a_j is a column vector, for $j \in N = \{1, \ldots, n\}$. Let us denote by

$$P_I = \text{conv}(x \in P : x \in \{0,1\}^n).$$

We observe first that the inequalities $x_j \geq 0$ are facets of P_I, provide that $a_j \geq a_0$ for all $j = 1, \ldots, n$. We shall call the inequalities $x_j \geq 0$, $j = 1, \ldots, n$, *trivial* facets of P_I. Observe that for any nontrivial facet $\pi x \leq \pi_0$ of P_I we have $\pi_j \geq 0$, $j = 1, \ldots, n$ and $\pi_0 > 0$. Consequently, requirement in Theorem 1.1 (c) states that there must exist $d = \dim(P_I)$ affinely independent vertices of P_I satisfying this condition. We now assume explicitly that $a_j \leq a_0$ for all $j \in N$. Hence, $\dim(P_I) = n$.

Let T be a nonempty proper subset of N. Let us denote by P^T the polytope obtained from P by setting the variables x_j, $j \in T$, equal to zero, i.e.,

$$P^T = P \cap \bigcap_{j \in T} \{x \in \mathbb{R}^n : x_j = 0\} \tag{1.1}$$

and define P_I^T to be the convex hull of the zero-one points of P^T.

Let $T = \{j_1, \ldots, j_t\}$, where $t = |T|$ and the elements of T are arbitrary ordered. For $q = 1, \ldots, t$ define T_q to be

$$T_q = T_{q-1} \cup j_q,$$

with the convention that $T_0 = \emptyset$. Similar to P^T and P_I^T, we denote by P^{T-T_q} the polytope obtained from P by setting the variables x_j, $j \in T - T_q$, equal to zero and define P^{T-T_q} to be the convex hull of the zero-one points of P^{T-T_q}. Note that with the above definitions $P^{T-T_0} = P^T$ and $P^{T-T_t} = P$. Furthermore, by the above assumptions we have that $\dim(P^{T-T_q}) = \dim(P^{T-T_q}) = (n - t + q)$.

Let $\pi x \leq \pi_0$ be any valid inequality for P_I, that is a (nontrivial) facet for the $(n-t)$-dimensional polytope P_I^T and consider the zero-one problem

$$z = \max \left\{ \pi x, x \in P_I^{T-T_I} \cap \{x \in \mathbb{R}^n : x_{j_1} = 1\} \right\}, \tag{1.2}$$

where we have set the variables x_j,$j \in T - T_1$, equal to zero and the variable x_{j_1} equal to 1.

Let us define the vector π^1 as follows. $\pi_j^1 = \pi_j$ for all $j \in N - T$, $\pi_{j_1}^1 = \pi_0 - \bar{z}$, $\pi_j^1 = 0$ otherwise, where $\bar{z}$ is the optimal objective function value of (1.2). It follows easily that $\pi_{j_1}^1 \geq 0$ and that the inequality $\pi^1 x \leq \pi_0$ is a valid inequality for $P_I^{T-T_1}$. Continuing the above process with j_2, etc., until T is exhausted, we obtain a (nontrivial) facet for the n-dimensional polytope P_I. To be more specific, suppose

that π_j, $j \in N - T$, and $\pi_0 > 0$ are given. Let us define a sequence of maximization problems (H_q) as

$$z_q = \max \sum_{j \in N-T} \pi_j x_j + \sum_{j \in T_{q-1}} \pi_j x_j$$

subject to

$$\sum_{j \in N-T} a_j x_j + \sum_{j \in T_{q-1}} a_j x_j \leq a_0 - a_{j_q}$$

$$x_j \in \{0, 1\} \qquad \text{for all } j \in (N - T) \cup T_{q-1}$$

where the π_j, $j \in T_{q-1}$, are defined recursively by

$$\pi_{j_q} = \pi_0 - z_q$$

and z_q is the optimal value of the objective function of the problem (H_q), $q = 1, \ldots, t$.

Theorem 1.3. *(Padberg [120]) Let $T = \{j_1, \ldots, j_t\}$, where $1 \leq t = |T| \leq n - 1$, be an arbitrarily ordered subset of N and let $\pi x \leq \pi_0$ be a nontrivial facet of P_I^T as defined in (1.1). Let π' be defined by $\pi'_j = \pi_j$ for all $j \in N - T$, $\pi'_{j_q} = \pi_0 - z_q$ for $q = 1, \ldots, t$, where z_q are obtained by solving the problems (H_q) for $q = 1, \ldots, t$. Then $\pi' x \leq \pi_0$ is a nontrivial facet of P_I.*

1.4 POLYHEDRAL COMBINATORICS

The area of research in which polyhedra arising from combinatorial optimization problems are investigated is often referred to as *Polyhedral Combinatorics* and its principal ideas are discussed next. Schrijver [138], Nemhauser and Wolsey [114] and Wolsey [152] are some of the books where those techniques are described in details.

Connections between Combinatorial Optimization and continuous or zero-one Linear Optimization can be established as follows.

Definition 1.24. Given a finite set E, let $\mathcal{I} \subseteq 2^E$ be a collection of *feasible solutions*, and let $c : E \to \mathbb{R}$ be the so called *objective function*. For each set $F \subseteq E$ let $c(F) := \sum_{e \in F} c(e)$. A *linear combinatorial optimization problem* is to find a set $I^* \in \mathcal{I}$ with

$$c(I^*) = \max\{c(I) : I \in \mathcal{I}\}.$$

We denote a linear combinatorial optimization problem by $(E, \mathcal{I}, c)$.

For the finite ground set E let $\mathbb{R}^E$ be the $\mathbb{R}$-vector space indexed by the elements of E.

Definition 1.25. Given a finite set E, and a set $F \subseteq E$, the *incidence vector* $x^F \in \mathbb{R}^E$ is defined by

$$x_e^F = \begin{cases} 1, \text{ if } e \in F \\ 0, \text{ if } e \notin F \end{cases}$$

With a combinatorial optimization problem $(E, \mathcal{I}, c)$ we associate the polytope

$$P_{\mathcal{I}} = \text{conv}\{x^I : I \in \mathcal{I}\}$$

Because the incidence vectors are 0-1-vectors, they are exactly the vertices of the polytope $P_{\mathcal{I}}$. If we associate with the function $c : E \to \mathbb{R}$ of a combinatorial optimization problem by a vector $c \in \mathbb{R}^E$, we can solve the combinatorial optimization problem by solving the optimization problem $\max\{c^T x : x \in P_{\mathcal{I}}\}$. Unfortunately we do not know any efficient algorithm to solve an optimization problem, when the solution space is only defined as the convex hull of an implicitly described set of points. However, according to classical result of Farkas, Weyl and Minkowsky (see Schrijver [138]) there exists a finite set of inequalities $Ax \leq b$, such that $P_{\mathcal{I}} = \{x : Ax \leq b\}$. Hence we could transform the combinatorial optimization problem $(E, \mathcal{I}, c)$ to the linear program $\max\{x : Ax \leq b\}$. There are finite algorithms to transform one representation of the polytope $P_{\mathcal{I}}$ into the other that can be used for very small problem instances.

As we have already mentioned, since the number of constraints may be too large to be represented in a computer, or too large to be handled by the LP-solver, we can still attempt to solve the problem with the following approach. We start with a small subset of constraints and compute an optimal solution subject to these constraints. We now check if any of the constraints not in the current linear program is not satisfied. If such contraints are identified, we add one or more of them to the current linear program and resolve it. If no constraint is violated, then the current optimum solution also solves the original problem. This is the basic principle of the so called *cutting plane approach*, whose name originates from the fact that the constraints added to the current linear program *cut off* the current solution because it is infeasible for the original combinatorial problem.

Note the important fact that the approach does not require that an explicit list of the constraints defining the original problem must be present. It is only required a method for identifying inequalities that are valid for the original problem but violated by the current solution.

Definition 1.26. Given a bounded rational polyhedron $P \subseteq \mathbb{R}^n$ and a rational vector $v \in \mathbb{R}^n$, the *separation problem* is, either conclude that v belongs to P or, if not, find a rational vector $w \in \mathbb{R}^n$ such that $w^T x < w^T v$ for all $x \in P$.

According to this, the following theorem gives the equivalence between solving an optimization problem and solving the equivalent separation problem.

Theorem 1.4. *For any proper class of polyhedra, the optimization problem is polynomially solvable if and only if the separation problem is polynomially solvable.*

This theorem is a consequence of the more general result of Grötschel, Lovász, and Schrijver [78]. Its proof involves some advanced topics in Linear Programming, including the Ellipsoid Method for solving Linear-Programming problems.

An algorithm that solves the general separation problem is called *exact separation algorithm*. Unfortunately, exact algorithm are often not known for classes of valid

```
Initialize the constraint system (A', b') with a small subset of
the constraints system (A, b).
repeat
  Compute an optimum solution of c^T x̄ = max{c^T x : A'x ≤ b', x ∈ ℝ}
  if ( x̄ not feasible )
    Generate a cutting plane (f, f_0), f ∈ ℝ^n with
      (i)  f^T x̄ > f_0
      (ii) f^T x̄ ≤ f_0 for all y ∈ {x : Ax ≤ b, x_i integer for all i ∈ ℐ}
     Add the inequality f^T x ≤ f_0 to the constraint system (A', b')
  endif
until ( x̄ be feasible )
```

Fig. 1.1 Cutting plane algorithm.

inequalities and in some cases it can even be shown that the separation problem in its optimization form for a certain class of inequalities is $\mathcal{NP}$-hard. In this case, we usually have to resort to a *heuristic separation algorithm*, which may find violated inequalities, but it may also fails since, it is not guaranteed that no constraint of the class is violated.

Figure 1.1 shows a generic cutting plane algorithm for solving a mixed Integer Linear Programming $\max\{c^T x^I : Ax^I \leq b, x^I \text{ integer for all } I \in \mathcal{I}\}$.

Cutting plane algorithms using specific cutting planes, (e.g., facet defining inequalities) often have to stop without finding an optimum solution. This can have two different reasons. First, the complete linear description for $\mathcal{NP}$-hard combinatorial optimization problem is unknown. Second, even if a big class of facets is known, no efficient algorithm may be available for the solution of the exact separation problem of this class. At this point we can apply another basic algorithmic technique for solving hard mixed integer optimization problems: *branch-and-bound*.

Branch-and-bound is a divided-and-conquer approach trying to solve the original problem by splitting it into smaller problems, denoted as subproblems, for which upper and lower bounds are computed. The crucial part of a successful branch-and-bound algorithm is the computation of upper and lower bounds for these subproblems. Here one uses the fundamental concept of relaxation.

Definition 1.27. Let $F = \{x^I : Ax^I \leq b, x^I, \text{ integer for all } i \in \mathcal{I}\}$ be the set of feasible solutions of a mixed integer optimization problem $\max\{c^T x : x \in F\}$. A maximization problem

$$\max\{r(x) : x \in R\}$$

is a relaxation of the mixed integer optimization problem, if

$$F \subseteq R \text{ and } c^T x \leq r(x) \qquad \text{for all } x \in F.$$

Hence, a solution of the relaxed problem gives an upper bound on the optimum objective function value of the problem it was derive from. The tighter the relaxation, the better this bound will be. But a relaxation is only useful if it can be treated at least practically efficiently by optimization algorithms.

By dropping the integrality conditions of the integer variables of a mixed integer optimization problem we get a *linear programming relaxation*, which is basic in the context of the cutting plane algorithms. This relaxation can be tightened by adding further valid inequalities.

A branch-and-bound algorithm maintains a list of subproblems of the original problem, which is initialized with the original problem itself. In each major iteration step the algorithm selects a subproblem from this list, computes a local upper bound for this subproblem, and tries to improve the global lower bound. If the local upper bound does not exceed the global lower bound, the active subproblem is fathomed, because its solution cannot be better than the best known feasible solution. Otherwise, we check if the optimal solution of the relaxation of the subproblem is a feasible solution of the original problem. In this case, we have solved the subproblem and thus, it is fathomed.

If the local upper bound exceeds the global lower bound and no feasible solution was found for the active problem, we perform a branching step by splitting the active subproblem into a collection of new subproblems whose union of feasible solutions contains all feasible solutions of the active subproblem. The simplest branching strategy consists of defining two new subproblems by changing the bounds of the variable. Suppose $i \in \mathcal{I}$ has a fractional value $\bar{x_i}$ in the LP-solution. Then, the new upper bound of the variable i in the fist new subproblem is $\lfloor \bar{x_i} \rfloor$, whereas its lower bound remains unchanged. In the second subproblem the upper bound keeps its old value, but the new lower bound of the variable i is $\lceil \bar{x_i} \rceil$.

If the list of subproblems becomes empty, then the memorized feasible solution (whose objective function value is equal to the global upper bound) is the optimum solution.

A *Branch-and-Cut* algorithm is a branch-and-bound algorithm in which cutting planes are generated throughout the branch-and-bound tree. Although this may seem to be a minor difference, in practice there is a change of philosophy. Rather than reoptimizing fast at each node, the new philosophy is to do as much work as necessary to get a tight upper bound for the subproblem. Now the goal is not only to reduce the number of required branching in the tree significantly by using cuts and improved formulations, but also to try anything else that may be useful such as preprocessing at each node, a primal heuristic at each node, and so forth. This technique will be used in Chapter 5.

1.5 MULTICRITERIA OPTIMIZATION

Some concepts on Multicriterion Optimization as well as the description of the basic algorithm for solving bicriterion problems are provided in this section. They will be used in Chapter 7. For more details on this topic we refer to Steuer [143], Goicoechea, Hansen and Duckstein [70] and Ehrgott [49].

Optimization can be viewed as a discipline which comprises the whole interactive process of analysis and design resulting in an optimal system. Instead of one scalar objective function, usually several conflicting and non-commensurate (i.e. such

quantities which have different units) criteria appear in an optimization problem. This situation forces the designer or analyst to look for a good compromise solution by considering trade-off between the competing criteria. Consequently, he/she must take a decision-maker's role in an interactive design process where typically several optimization problems must be solved. Multicriterion (multiobjective, Pareto, vector) optimization offers a flexible approach for the designer to deal with this decision-making problem in a systematic way.

An important drawback of considering such problems lies in the difficulty of defining an appropriate notion of optimality and, given such a notion, finding an optimal solution. Obviously, the situation becomes more complicated when more criteria are involved, unless the criteria are not in conflict with each other; roughly speaking, two criteria are not in conflict if a solution that performs well on one criterion is likely to perform well on the other criterion. If the criteria are conflicting, then the different solutions have to be weighted against each other. To that end, various options exist. The first one is to specify a score on the value on the most important criterion; a solution is then selected to perform well on the other criteria while satisfying the bound. The second option is to aggregate the criteria into a single objective function; a solution is chosen that is optimal for this objective function. The third option is based upon an interactive version of decision making: an analyst determines a candidate solution and presents it to a decision maker, who either decides to accept it or tells the analyst on which criterion the score should be improved.

Another important issue concerns the question what constitutes a representative set of candidates solutions. An obvious choice is the set of all *non-dominated* solutions. A solution is said to be non-dominated if it outperforms any other solution on at least one criterion. If the number of non-dominated solutions is large, then an analyst may impose extra restrictions upon the set of candidate solutions; for example, he/she can impose an upper bound on the value of a criterion.

In Chapter 7 of this thesis a *simultaneous* minimization, in contrast to *hierarchical* minimization, is performed in order to combine conflicting criteria. In case of hierarchical minimization, the performance criteria are ranked in order of importance; the less important criterion is minimized subject to the constraint that the solution of the problem is optimal with respect to the more important criterion. In case if simultaneous minimization, the criteria are aggregated into a single composite objective function, which is then minimized. Note that simultaneous minimization turns into hierarchical minimization for an appropriate choice of the composite objective function.

We assume that any composite objective function is non-decreasing in both arguments. This assumptions reflects the opinion that a dominated solution should not be chosen as the optimal solution.

Definition 1.28. Let f_1 and f_2 be two performance criteria. Then the *criterion space* is

$$Z = \{z \in \mathbb{R}^2 : z = (f_1(\sigma), f_2(\sigma)), \sigma \in \mathcal{I}\},$$

where $\mathcal{I}$ is the set of feasible solutions of an optimization problem.

Definition 1.29. Let $(\tilde{f}_1, \tilde{f}_2) \in Z$. Then $(\tilde{f}_1, \tilde{f}_2)$ is *non-dominated* if there does not exist another $(f_1', f_2') \in Z$ such that $\tilde{f}_1 \leq f_1'$ and $\tilde{f}_2 \leq f_2'$, and at least one of the inequalities is strict.

Definition 1.30. Let $(\tilde{f}_1, \tilde{f}_2) \in Z$. Then $(\tilde{f}_1, \tilde{f}_2)$ is *efficient* if there exist a real nonnegative value $\omega \in [0, 1]$ such that $\omega\tilde{f}_1 + (1-\omega)\tilde{f}_2 \leq \omega f_1' + (1-\omega)f_2'$ for all $(f_1', f_2') \in Z$.

Definition 1.31. The *efficient frontier* is the shortest curve that connects all efficient points.

Definition 1.32. $(\tilde{f}_1, \tilde{f}_2) \in Z$ is *extreme* with respect to f_1 and f_2 if it corresponds to a vertex of the efficient frontier.

Definition 1.33. A feasible solution σ is *Pareto optimal* with respect to two performance criteria f_1 and f_2 if there is no feasible solution π such that $f_1(\pi) \leq f_1(\sigma)$ and $f_2(\pi) \leq f_2(\sigma)$, where at least one of the inequalities is strict.

Theorem 1.5. *If the composite objective function F of (f_1, f_2) is non-decreasing in both arguments, then there exists a Pareto optimal point for (f_1, f_2) in which the function F attains its minimum.*

Proof. Let $(f_1^{(1)}, f_2^{(1)})$ be a point in which F attains its minimum. If $(f_1^{(1)}, f_2^{(1)})$ is not Pareto optimal, then there exists a Pareto optimal point $(f_1^{(2)}, f_2^{(2)})$, with $f_1^{(2)} \leq f_1^{(1)}$ and $f_2^{(2)} \leq f_2^{(1)}$. Hence, $F(f_1^{(2)}, f_2^{(2)}) \leq F(f_1^{(1)}, f_2^{(1)})$, implying that F also attains its minimum in $(f_1^{(2)}, f_2^{(2)})$. □

Figures 1.2 and 1.3 show the non-dominated solutions as well as the efficient frontier in a problem minimizing two performance criteria f_1 and f_2.

We describe now an algorithm to obtain all non-dominated solutions and to generate the efficient frontier solving iteratively single objective problems. This algorithm is based on the general scheme of the hybrid method described in Goicoechea, Duckstein and Fogel [69] that combines both the *weighting method* and the *ϵ-constraint method*. This scheme combines both criteria linearly and introduce a weighting factor for each of them. If these weighting coefficients, denoted here by ω_1 and ω_2, are interpreted as parameters we obtain a linear weighting method which can be used for the generation of non-dominated points. Without loss of generality the normalization $w_1 + w_2 = 1$ may be applied, so we will only relate to w_1. By varying the weight ω_1 and by solving this scalar problem separately for each fixed parameter combination, we can compute all non-dominated points.

The basic steps of the general method are showed in Figure 1.4. The initial step computes the two starting points ($f_1^{(1)}$,$f_2^{(1)}$) and ($f_1^{(2)}$,$f_2^{(2)}$) by optimizing hierarchically both criteria f_1 and f_2 (see Figure 1.6). These two initial points constitute the first interval to be examined as well as the first two non-dominated points. The set I stores the remaining intervals and ND is the set of non-dominated points. Both the weight ω_1 and the bounds are computed as long as a new interval is selected from

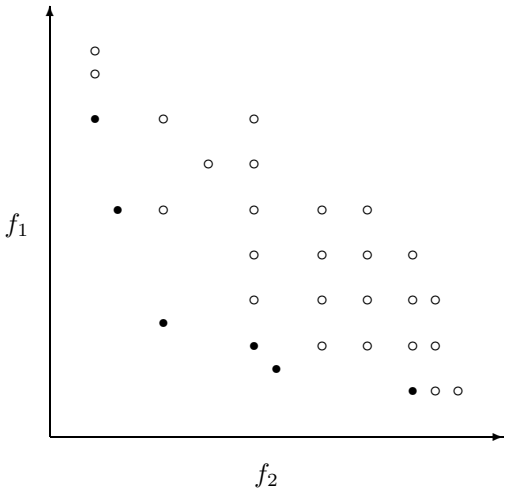


Fig. 1.2 Set of non-dominated solutions with respect to criteria f_1 and f_2.

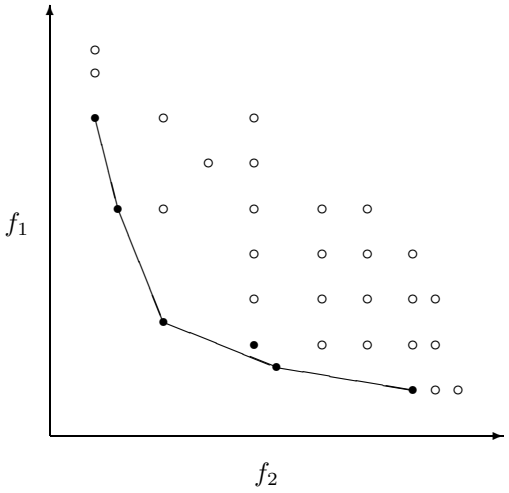


Fig. 1.3 Efficient frontier with respect to criteria f_1 and f_2.

Input: f_1, f_2, σ ***Output:*** ND
$f_1^{(1)} := \min_{\sigma \in P} f_1(\sigma)$
$f_2^{(1)} := \min_{\sigma \in P} (f_2(\sigma) : f_1(\sigma) \leq f_1^{(1)})$
$f_2^{(2)} := \min_{\sigma \in P} f_2(\sigma)$
$f_1^{(2)} := \min_{\sigma \in P} (f_1(\sigma) : f_2(\sigma) \leq f_2^{(2)})$
I:=[($f_1^{(1)}$,$f_2^{(1)}$),($f_1^{(2)}$,$f_2^{(2)}$)]
$ND := (f_1^{(1)}, f_2^{(1)}) \cup (f_1^{(2)}, f_2^{(2)})$
while $I \neq \varnothing$
Select from I an interval $[(f_1^{(1)}, f_2^{(1)}), (f_1^{(2)}, f_2^{(2)})]$
$I := I \setminus \{[(f_1^{(1)}, f_2^{(1)}), (f_1^{(2)}, f_2^{(2)})]\}$
($f_1^{(2)}$,$f_2^{(2)}$):=stack.pop()
$m := \frac{f_2^{(2)} - f_2^{(1)}}{f_1^{(1)} - f_1^{(2)}}$
$\omega := \frac{m}{m-1}$
$\sigma^* := \arg$ WSCP(ω,$f_1^{(1)}$,$f_2^{(2)}$)
if $\sigma^* \neq \varnothing$
$ND := ND \cup (f_1(\sigma^*), f_2(\sigma^*))$
$I := I \cup [(f_1^{(1)}, f_2^{(1)}), (f_1(\sigma^*), f_2(\sigma^*))] \cup [(f_1(\sigma^*), f_2(\sigma^*)), (f_1^{(2)}, f_2^{(2)})]$

Fig. 1.4 Pseudocode of the hybrid algorithm.

the set of intervals I. Accordingly, the weighted single criterion problem (WSCP) is solved. If the current WSCP is feasible, a new non-dominated point and two new intervals are generated.

We now provide the basis for the construction the algorithm described in Chapter 7.

Theorem 1.6. *Let S be the set of feasible solutions. If S has an efficient point, then at least one extreme point of S is efficient.*

Definition 1.34. Let C_B denote the basic columns of the criterion matrix C; C_N the non basic columns, and N the non-basic columns of the constraints matrix A. Then let W denote the $k \times (n-m)$ *reduced cost* matrix where $W = C_N - C_B B^{-1} N$.

Definition 1.35. B is a *efficient basis* if and only if B is an optimal basis of the weighted-sum LP for some vector λ.

Since the reduced cost row of the weighted-sums LP is given by $\lambda^T W$, basis B is efficient if and only if the system

$$\lambda^T W \leq 0$$

$$\lambda > 0$$

is consistent.

Theorem 1.7. *Let $x \in S$ be the extreme point associated with efficient basis B. Then, x if efficient.*

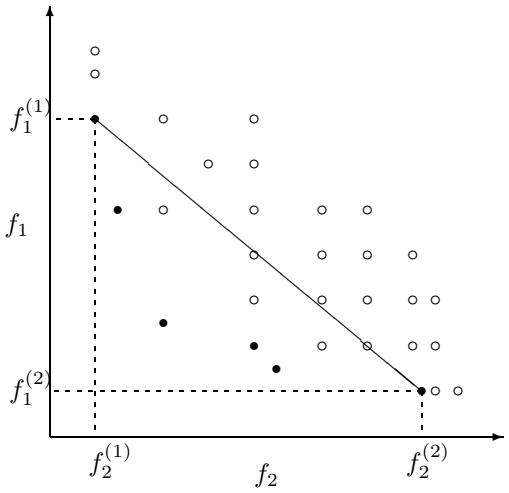


Fig. 1.5 First step of the weighting method.

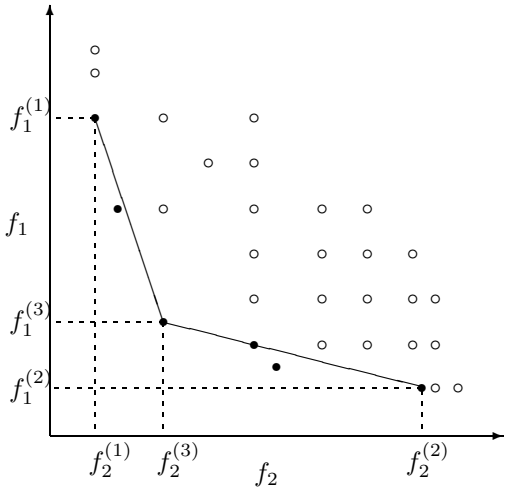


Fig. 1.6 Two new intervals obtained from the first optimization step.

Theorem 1.8. *Let $x \in S$ be an efficient extreme point. Then, there exist an efficient basis B associated with x.*

Definition 1.36. Bases $\bar{B}$ and $\hat{B}$ are *adjacent* if and only if one can be obtained from the other in one pivot.

Definition 1.37. Let B be an efficient basis. Then, x_j is an *efficient non-basic variable* if and only if there exists a vector λ such that

$$\lambda^T W \leq 0$$

$$\lambda^T w^j = 0$$

where w^j is the j^{th} column of W.

Definition 1.38. Let B be an efficient basis and x_j an efficient non-basic entering variable. Then, any feasible pivot from B is an *efficient pivot* operation.

Theorem 1.9. *Let B an efficient basis. Then, any efficient pivot from B yields an adjacent efficient basis $\hat{B}$.*

Theorem 1.10. *Let $\bar{B}$ and $\hat{B}$ be adjacent efficient bases such that one can be obtained from the other by means of an efficient pivot. Let $\bar{x}$ and $\hat{x}$ be the extreme points associated with $\bar{B}$ and $\hat{B}$, respectively. Then, the edge $\gamma(\bar{x}, \hat{x})$ is efficient.*

Definition 1.39. Let $\bar{B}$ and $\hat{B}$ be efficient bases. If one can be obtained from the other by performing only efficient pivots, $\bar{B}$ and $\hat{B}$ are said to be *connected*.

Theorem 1.11. *All efficient bases are connected.*

Definition 1.40. Two efficient extreme points of S are *edge-connected* if they are connected by means of a path of efficient edges of S.

Theorem 1.12. *All efficient extreme points of S are edge-connected.*

1.6 SOME BASIC OPTIMIZATION PROBLEMS

Some of the algorithms described in this thesis are methods to solve subproblems involved in some stages of the resolution of the main problem. Those subproblems are described in this section. The *Assignment Problem* is referred in Chapters 3 and 4 as a subproblem of the asymmetric Traveling Purchaser Problem. The *Knapsack Problem* and *Uncapacitated Facility Location Problem* are mentioned in the polyhedral study carried out in Chapter 4. Finally the *Set Covering Problem* is referred as a relaxation of a subproblem of the Traveling Purchaser Problem during the description of the heuristic approach in Chapter 8.

The Assignment Problem

Let us consider n workers available to carry out n jobs. Each person must be assigned to perform exactly one job. Some workers are better suited to particular jobs than others, so there is an estimated cost c_{ij} if person i is assigned to job j. The problem is to find a minimum cost assignment. Using the binary variables

$$x_{ij} = \begin{cases} 1, & \text{if person } i \text{ does the job } j \\ 0, & \text{otherwise} \end{cases}$$

a mathematical model is

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

s.t.

$$\begin{aligned} \sum_{j=1}^{n} x_{ij} &= 1 \qquad \text{for } i = 1, \ldots, n \\ \sum_{i=1}^{n} x_{ij} &= 1 \qquad \text{for } j = 1, \ldots, n \\ x_{ij} &\in \{0, 1\} \qquad \text{for } i = 1, \ldots, n, \text{ and } j = 1, \ldots, n. \end{aligned}$$

The 0-1 Knapsack Problem

There is a budget b available for investment in projects during the coming year and n projects are under consideration, where a_j is the outlay for project j, and c_j is its expected return. The goal is to choose a set of projects so that the budget is not exceeded and the expected return is maximized. Using the binary variables

$$x_j = \begin{cases} 1, & \text{if project } j \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

a mathematical model is

$$\max \sum_{j=1}^{n} c_j x_j$$

s.t.

$$\begin{aligned} \sum_{j=1}^{n} a_j x_j &\leq b \\ x_j &\in \{0, 1\} \qquad \text{for } j = 1, \ldots, n. \end{aligned}$$

The Set Covering Problem

Given a certain number of regions, the problem is where to install a set of emergency service center. For each possible center the cost of installing a service center, and

which regions it can service are known. The goal is to choose a minimum cost set of service centers so that each region is covered. Let $M = \{1, \ldots, m\}$ be the set of regions, and $N = \{1, \ldots, n\}$ the set of potential centers. Let $S_j \subseteq M$ the regions that can be serviced by a center at $j \in N$, and c_j its installation cost.

To facilitate the description, we first construct a 0-1 incidence matrix A such that $a_{ij} = 1$ if $i \in S_j$, and $a_{ij} = 0$ otherwise. Using the binary variables

$$x_j = \begin{cases} 1, & \text{if center } j \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

a mathematical model is

$$\min \sum_{j \in N} c_j x_j$$

s.t.

$$\begin{aligned} \sum_{j \in N} a_{ij} x_j \geq 1 \qquad & \text{for } i \in M \\ x_j \in \{0, 1\} \qquad & \text{for } j \in N. \end{aligned}$$

The Uncapacitated Facility Location Problem

Given a set of potential depots $N = \{1, \ldots, n\}$ and a set $M = \{1, \ldots, m\}$ of clients, suppose there is a fixed cost f_j associated with the use of depot j, and a transportation cost c_{ij} if all of clients i's order is delivered from depot j. The problem is to decide which depots to open, and which depot serves each client so as to minimize the sum of the fixed and transportation costs. Using the binary variables

$$y_j = \begin{cases} 1, & \text{if depot } j \text{ is used} \\ 0, & \text{otherwise} \end{cases}$$

x_{ij} is the fraction of the demand i satisfied from depot j

a mathematical model is

$$\min \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} + \sum_{j \in N} f_j y_j$$

s.t.

$$\begin{aligned} \sum_{j \in N} x_{ij} = 1 \qquad & \text{for } i \in M \\ x_{ij} \leq y_j \qquad & \text{for } i \in M, \text{ and } j \in N \\ x_{ij} \in \{0, 1\} \qquad & \text{for } i \in M, \text{ and } j \in N \\ y_j \in \{0, 1\} \qquad & \text{for } j \in N. \end{aligned}$$

2

The Traveling Purchaser Problem: An introduction

An introduction on the Traveling Purchaser Problem (TPP) is provided in this chapter. The aim is to give a wide description of this problem; to provide an extensive literature review, taking into account all the previous works on the TPP; to enumerate potential applications to real world problems; and to place the TPP in its scope by describing some related problems; finally a trnasformation of the TPP into the *Generalized Traveling Salesman Problem* is also provided.

This dissertation is concerned with a generalization of the well-known Traveling Salesman Problem (TSP), known as the *Traveling Purchaser Problem* (TPP). The problem can be defined as follows. Let us consider a set of products or items to be purchased and a vehicle originally at a depot. There is a requirement of units for each different product. Let us also consider a set of markets, each selling some units of a certain number of products. The unit price of a product depends on the market where it is available. It is also known the travel cost between each two locations. The TPP asks for selecting a subset of markets and routing the selected markets with a vehicle such that the demand of each product is satisfied and the total purchasing and travel cost is minimized. It is assumed that

i) each product is available in at least one market;

ii) no product is available in the depot;

iii) the required demand can be purchased.

The particular case in which there is not restricted offer of a product at each market is called *unrestricted TPP*. It can be seen as the TPP with one-unit demand for each product.

This problem also arises in the Scheduling context. Let us consider a multipurpose machine. This machine is designed for performing different kinds of tasks or jobs by changing its specific tool, available in a magazine or set of tools. Each one of these tools corresponds to an state of a multipurpose machine. Notice that to change the machine from the state s_i to the state s_j takes a *setup* time, and that, return to state s_i from state s_j takes a different time, since the procedure of installing and uninstalling the tool could be not symmetric (see the state diagram in Fig. 2.1). Let us also consider a set of jobs that can be grouped according to the required tool needed to be processed. Each of this jobs could be processed in one or different states, that is, with one or several tools, but the processing time is closely connected with this tool. The TPP looks for a sequence of a subset of states to perform the set of jobs, starting and finishing in a base state without tool, minimizing the total processing time, which includes both the total setup time and the total processing time. If each state is able to process as many units as necessary then an unrestricted TPP arise. On the other hand, if the processing capacity of a specific state is resource constrained then we are addressing a restricted or general TPP.

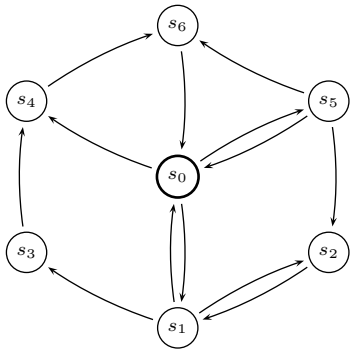


Fig. 2.1 State diagram of a multipurpose machine.

2.1 LITERATURE REVIEW

Most of the works in literature deal with the unrestricted TPP. The name of Traveling Purchaser Problem for the unrestricted TPP was coined by Ramesh [126]. He presents two algorithms, a lexicographic search algorithm and a near neighbour algorithm. The former is an exact algorithm based on lexicographic search. In this approach each solution is represented as a sequence of symbols and search for an optimal solution is analogous to search for a specific word's location in a dictionary. From a partial word, solutions are generated in some hierarchy which reflects an analogous hierarchy in

theirs values. Each partial word defines a block of solutions, and for each block of solutions a lower bound is computed. If this lower bound exceeds the value of a known solution (trial solution), the entered block of words is rejected as it necessarily does not contain solutions of value better than the trial solution value, and the next block of solutions is explored. However, the lower bound depends only on the travel cost from different markets to the depot and it is independent of the purchasing cost. Probably that is the reason for the bad performance of this algorithm.

The near neighbour algorithm also proposed by Ramesh in [126] is based on a heuristic called NEARINSERT (see [134] and [137] for details) for the Traveling Salesman Problem, that starts inserting a single vertex, and iteratively extend a path by inserting all other vertices in a greedy way. The Ramesh's heuristic begins considering the first solution obtained in the lexicographic search as an approximate solution for TPP, and adapts the former algorithm making use of the mentioned heuristic for the TPP.

Computational experience is presented in that article involving instances only up to 12 markets and 10 products, and 8 markets and 22 products.

Prior to Ramesh, Burstall [25] described a real world problem of similar structure in a tube manufacturing firm, but focused on a scheduling approach. The problem of Burstall was to do technical planning, in order to determine for each batch of tubes the range of technically feasible ways in which they could be manufactured. In essence, a set of batches of tubes has to be processed by a multi-state machine. The processing time of each batch as well as technical feasibility depend on the state. On the other hand, the setup time taken by the machine when it change from a state to another is also given. The problem is to process the set of batches minimizing the total time, that includes both the total setup time and the processing time for each processed batch. If additionally, we consider one dummy state in which the machine has to start and stop, it is clear that this problem becomes the TPP, where the states become the markets and each batch of jobs is a product. However, it easy to realize that neither triangle inequality hold nor a symmetric representation is unsuitable in this type of problem.

Burstall developed a heuristic which was further commented on by Lomnicki [107]. He checked his algorithm on a set of instances obtained from the real problem he was addressing, solving problems up to 27 states (markets) and 17 jobs (products). One of these instances is used as illustrative example in Chapter 7 in order to show the behaviour of the proposed algorithm.

Buzacott and Dutta [26] developed an exact procedure based on Dynamic Programming for this sequencing problem. His program is able to solve instances having 12 jobs or less, and he checked it on the Burstall's factory instances other instances involving 10 jobs and 10 states that we presume randomly generated.

Another exact algorithm was presented by Singh and van Oudheusden [141]. They developed a branch-and-bound algorithm. The main idea of this approach is to break up the set of all possible tours into smaller and smaller subsets, and to calculate for each of them a lower bound on the sum of the travel cost and purchasing cost. The lower bound is computed by solving a relaxation of the problem which is similar to the *Uncapacitated Facility Location Problem* (UFLP). The bounds guide the partition

of the subset and identify an optimal solution when a subset is found that contains a single tour, and whose bound is less than or equal to the lower bounds for all other subsets.

Their computer program, called TRAPUR is checked on both asymmetric and symmetric instances. For the asymmetric case, the travel costs are integer randomly generated from a uniform distribution in the range $[15, 30]$. The purchasing cost different from arbitrary large value, M, are also integer generated from a uniform distribution in the range $[a, a + 10]$, where a is an integer whose value does not influence the optimal solution as all the products are to be purchased. The proportion of purchasing cost different from M is around 50%. They generated a total of 65 problems with asymmetric costs, and 10 to 25 markets and 10 to 100 products.

For the symmetric case, the travel cost values are taken from the travel cost date for the 33-city TSP given in Karg and Thompson [90]. Notice that those instances do not satisfy the triangle inequality. The purchasing cost different from M are integer randomly generated from a uniform distribution in the range [0,500]. They generated a total of 40 problems with 10 to 20 markets and 15 to 50 products.

Since the unrestricted TPP is known to be $\mathcal{NP}$-hard in the strong sense (it becomes the TSP when each product can be purchased in only one market) the literature on TPP is mostly directed towards the development of heuristic or near optimal methods. One of these heuristic procedures developed so far is due Golden, Levy and Dahl [73]. They proposed the *Generalized Saving Heuristic*. It starts with an initial tour containing the depot and the markets selling the largest number of products at their lowest available price. Ties are broken by selecting the market with the smallest sum of product prices. At each iteration, the non-visited market producing the largest cost saving is inserted in the current tour. The heuristic stops when no more saving can be achieved. Their heuristic was, later on, modified by Ong [116] who proposed the *Tour Reduction Heuristic* (TRH) It starts with an initial tour containing a subset of markets offering the n products and iteratively drops the markets yielding the largest cost reduction until no further improvement can be obtained. Ong also suggested using a good TSP algorithm to resequence the markets in the intermediate tours. The performance on this approach heavily depends on the initial subset of markets, on the number of times the TSP heuristic is applied, and on the performance of the TSP heuristic.

Pearn and Chien [123] suggested some improvements of the two previous works from Golden, Levy and Dahl [73] and Ong [116]. Two of them were related to the Generalized Saving Heuristic from Golden, Levy and Dahl [73]. The first one called *Parameter Selection Generalized Saving Heuristic*, uses a weighted saving function where a term reflecting the purchasing saving at a given market is multiplied by a weight and added to the travel cost saving. The second version, called the *Tie Selection Generalized Saving Heuristic* is similar to the original heuristic, but the tie-breaking rule selects the market closest to the depot instead of the market offering the smallest sum of product price.

Two improvements were also suggested by Ong [116] for the TRH. They suggested initially selecting the set of markets selling at least one product at its lowest price. They also tested two variants of the TRH. In the first one, called *Adjusted-Cheapest*

Tour-Reduction Heuristic, the initial set of markets contains all markets for which the price of one or more products augmented by their travel cost to the depot is minimal. In the second variant, called *Nearest-Cheapest Tour-Reduction Heuristic* the initial set includes the closest q markets to the domicile. They solved the TPP with five different values of q.

Another heuristic proposed by Pearn and Chien [123] is called *Commodity Adding Heuristic*. This heuristic implicitly assumes that all products are available at all markets. The procedure considers the first product from a list and constructs a least cost solution for this product. At each following iteration, it inserts the next product in the solution in a least cost manner. The authors also propose improving the solution by mean of the Basart and Huguet [17] TSP heuristic, or by market drop or market interchange operations.

Voß [151] presented metaheuristics bases on dynamic tabu search and simulated annealing for the TPP, which used dynamic strategies for managing of tabu list. In this paper he presents two of these dynamic strategies, the *reverse elimination method* and the *cancellation sequence method*, studying their impact on the Traveling Purchaser Problem. In order to compare his dynamic strategies Voß presents numerical results for the TPP computed on three known graphs from the literature with 10, 31 (Clarke and Wright [35]) and 52 (Paessens and Weuthen [122]) markets. Finally, tabu search seems to work better on the TPP.

Two heuristic procedures, *ADD-procedure* and *DROP-procedure*, are also developed for constructing the initial tours. The *ADD-procedure* is a iterative procedure that build a feasible tour by adding new markets to the tour, according to a saving criterion. As soon as a feasible cycle is created, additional markets are added until not improvement in the objective function is achieved.

On the other hand, the *DROP-procedure* is an inversion of the *ADD-procedure*. It starts with a feasible tour containing the whole set of markets. The procedure removes in each step the market which gives the largest reduction in the objective function. If no more reduction is possible, it terminates with a feasible solution. This procedure has been also described as a so called *tour reduction heuristic* by Ong [116]. Afterwards, the initial tour is determined by applying some TSP heuristic.

Based on those two procedures some deterministic exchange procedures have been proposed in [151], the IMP1 and IMP2. The IMP1 procedure compute the best tour after adding a market by the *ADD-procedure* and remove another one with *DROP-procedure*. By changing the order of these procedures it becomes the IMP2. Therefore, IMP1 and IMP2 produce two possible neibourhood definitions.

Another metaheuristic approach is presented by Boctor, Laporte and Renaud [22]. They presented several algorithms based on tabu search for solving approximately TPP instances, both in the unrestricted and general versions, in which the markets are locations in the Euclidean plane. These algorithms are tested on benchmarks up to $m \leq 200$ and $n \leq 200$ comparing with it exacts solutions. They also test their approach on bigger instances, but they compare their results with their own algorithm after running several times, and not with a lower bound. They compare two implementations of the *Commodity Adding Heuristic* (CAH) described by Pearn and Chien [123] called CAH1 and CAH2, three implementation of their *Perturbation*

Heuristics (PH) called UPH1 and UPH2 for the unrestricted TPP and CPH for the general TPP. These PH combine in different ways basic procedures as *market drop*, *market add*, *market exchange*, TSP heuristics, *cheapest insertion*, *double market drop* and *double market exchange*. Table 2.1 summarizes this section, including in addition the contribution of this dissertation.

2.2 APPLICATIONS

The structure of the traveling purchaser problem applies to many real life problems. We discuss here some of its potential applications to job scheduling, warehousing, routing and ring network desing.

Job scheduling

The TPP can be found in a scheduling context as noticed by Burstall [25] and by Buzacott and Dutta [26]. Let us consider a set of m jobs to be performed, and a multi-purpose machine, i.e., a machine that can assume n different configurations. Each job requires a given dedication (i.e. a set of tasks), while each configuration of the machine can execute only part of the dedication of the job. Depending on the configuration, it is known the time to perform a task of a job. The tasks of the jobs can be processed by different configurations and a configuration can be used to perform tasks of different jobs. It is also known the time to changeover from one configuration to another. The machine is initially in a default status (configuration v_0), in which the machine must be after all the jobs are completely executed. The TPP consists in selecting and sequencing a set of configurations to fully execute the jobs minimizing the total processing and changeover time. This application can be seen as the TPP with one-unit demand for each product.

Warehousing

One of the most interesting problems associated with warehousing is the *order-picking* problem. An *order* consists of a subset of required items that are stored in a warehouse. On receiving an order, the warehouse dispatches a vehicle from the picking area to pick the items in the order and transport them back to the shipping area. The objective is to minimize the distance travelled by the vehicle.

For the situation when only one item is stored in a single location area of the warehouse, the order-picking has been recognized as a variant of TSP. However, when an item is stored in more than one location area, the problem has the structure of the unrestricted TPP. Even more, not only one unit of each item but also some demand is required, the item is located at different locations of the warehouse, and in some cases it is not possible to satisfy the demand visiting one location. Then we are dealing with an instance of the TPP.

Table 2.1 Literature review and contributions of this Thesis.

Reference	Problem[a]	Algorithm	Instances	Biggest
Burstall [25]	USTPP	*Ad hoc* heuristic	Real instances	(27×17)
Buzacott and Dutta [26]	USTPP	Dynamic Programming	—	—
Ramesh [126]	USTPP	Lexicographic search *Ad hoc* heuristic	—	(12×10) and (8×22)
Golden, Levy and Dahl [73]	USTPP	*Ad hoc* heuristic	—	—
Ong [116]	USTPP	*Ad hoc* heuristic	[61, 35, 42]	(42×44)
Voß [151]	USTPP	meta-heuristic	[35, 122]	(52×83)
Singh and Oudheusden [141]	USTPP UATPP	Branch and Bound	Random [90]	(20×50) (25×100)
Pearn and Chien [123]	USTPP	*Ad hoc* heuristic	Random	(50×60)
Laporte, Riera and Salazar[b] [102]	USTPP RSTPP	Branch and Cut	Euclidean/[141, 123]	(200×200) (200×100)
Boctor, Laporte and Renaud [22]	USTPP RSTPP	meta-heuristic	[102]	(200×200)
Riera and Salazar[b] [129]	USTPP RSTPP	*Ad hoc* heuristic	[102]	(200×200) (200×100)
Riera and Salazar[b] [130]	Bicr. USTPP	Branch and Cut	[102]	(100×200)
Riera and Salazar[b] [131]	UATPP RATPP	Branch and Cut	Random/[141]	(200×200) (150×200)

[a] USTPP Unrestricted Symmetric Traveling Purchaser Problem
RSTPP Restricted Symmetric Traveling Purchaser Problem
UATPP Unrestricted Asymmetric Traveling Purchaser Problem
Bicr. USTPP Bicriterion Unrestricted Traveling Purchaser Problem

[b] Contributions of this thesis

Routing

One of the routing applications of the TPP is the school bus problem. A school bus has to pass by many points to get children to the school. Given the distance from the child's house to the potential bus stop as well as the cost of driving from point to point the problem is determine a tour of the bus which minimizes the sum of the driving cost and the weighted distances walked by the children from their houses to the nearest points of the tour.

Another related problem is the called shortest covering path (SCPP), introduced by Current, Revelle and Cohon [40], which is a synthesis of the set covering location problem and the shortest path problem. The SCPP determines the shortest path between two given vertices in a graph such that all the demand vertices are covered. A demand vertex is considered covered if either it is directly on the shortest covering path of if it is within a predetermined maximum distance from a vertex on the path. The SCPP can be shown to be an special case of TPP. The design of subway line or rail line may also be of particular interest. The stations may be located at some of the population centers while the neighboring areas will be covered by these stations. Similarly, the same principle may be applied to develop irrigation network in a given region of a road network between two principal cities.

Ring Network Design

Recently, the design of information and communication infrastructure has become a major challenge both within companies and between widespread places, e.g., in major cities where metropolitan area networks are of interest. High-bandwidth fiber optic networks occupy an intermediate position between Local Area Networks (LANs) and Wide Area Networks. Among the various topologies available to the design of such networks, ring networks may be beneficial because they provide some protection against link failures. Now we consider the *General Network Design Problem* (GNDP), which may be described as follow: There is a set of vertices representing routing devices that may be linked to a network. Any two vertices on the ring are able to communicate with each other so one gains a certain revenue. Moreover, there may be revenues for each vertex includes in the ring. On the other hand, constructions costs are incurred for the design of direct links. The basic objective is to maximize the sum of all revenues minus the construction costs.

Corresponding modifications of the GNDP arise from the problem of connecting LAN clients using the ring topology. Apart from n given vertices of the basic problem, there are some secondary vertices that have to be connected to the ring. Consequently, additional costs have to be taken into account. This problem is related to TPP. Here the items correspond to the secondary vertices, and the markets to the vertices of the basic GNDP, respectively.

A related problem is the *Ring Network Design Problem* (RNDP), which has been discussed in Gendreau, Labbé and Laporte [63] who concentrate on the development of efficient heuristics such as greedy construction, as well as greedy add-and-drop exchange based local search. Also, the *Steiner Ring Network Design Problem* (SRNDP),

introduce by Laporte and Norbert [99] is another related problem, which find a subtour of minimum length including a given subset of the vertex set. Gouviea and Pires [76] develop mathematical model for the *Steiner Ring Network Design Problem with Revenues* (SRNDPR) that generalizes the (SRNDP) with respect to revenues and additional constraints. The *Selective Traveling Salesman Problem* (STSP) or *Orienteering Problem* (OP) (Laporte and Martello [97], Fichetti, Salazar and Toth [55]) is to maximize the revenues associated with the vertices included in the ring while there is an upper bound $c(R)$. On the other hand, the *Prize Collecting TSP* (PCTSP) (Balas [8]) is to minimize link costs and penalties due to vertices not include in the ring while there is a lower bound for vertex revenues associated with the vertices included in the ring. Some of these problems will be briefly presented in the following section.

2.3 RELATED PROBLEMS

As already mentioned, the Traveling Purchaser Problem is a generalization of the well known Traveling Salesman Problem. In this section we list some problems from literature which are related to the problem we study. Some of them ask for optimizing a Hamiltonian cycle according to their specific criteria. Others look for a minimum subcycle subject to additional constraints.

The Traveling Salesman Problem

The *Traveling Salesman Problem* (TSP) is one of the most prominent combinatorial optimization problem, and it is the benchmark problem for new algorithmic ideas in this field. The TSP has influenced significantly the development of cutting plane algorithms of polyhedral combinatoric like the Branch-and-Cuts algorithms. The TSP is ease to state: given a finite number of cities along with the cost of travel between each pair of them, find the cheapest way of visiting all the cities and returning to the starting point. Surveys of works on TSP can be found in Bellmore and Nemhauser [19], Lawler, Lenstra, Rinnooy Kan, and Shmoy [103], Reinelt [128], and Jünger, Reinelt and Rinaldi [87] and recently in Guting and Punnen [83].

The TSP can be modelled as follows. We are given a complete undirected graph $G = (V, E)$ with vertex set $V := \{v_1, \ldots, v_n\}$ and edge set $E := \{[i, j] : i, j \in V, i \neq j\}$. In addition, a routing cost c_e is defined for each $e \in E$. Let $x_e = 1$ if edge $e \in E$ is chosen in the optimal solution, and $x_e = 0$ otherwise. The next ILP model formulates the TSP.

$$w^{TSP} := \min \sum_{e \in E} c_e x_e,$$

subject to

$$\sum_{e \in \delta(v_i)} x_e = 2 \qquad \text{for all } v_i \in V \tag{2.1}$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad \text{for all } S \subset V \tag{2.2}$$

$$x_e \in \{0,1\} \qquad \text{for all } e \in E.$$

Constraints (2.1) impose that the number of edges incident with a vertex is 2. In order to avoid subcycles the subtour elimnation constraints (2.2) state that the subset $S \subset V$ must be connected to it complement by at least two edges of the cycle.

The Bottleneck TSP

The *Bottleneck Traveling Salesman problem* (BTSP) is a variation of the classical TSP that differs from the TSP only in the objective function. Instead of min-sum criterion, the min-max criterion is optimized.

The bottleneck TSP was introduced by Gilmore and Gomory [68]. Garfinkel and Gilbert [60] considered the general BTSP model and discussed an application of the problem in the context of machine scheduling. Meaningful interpretations of the BTSP model and its variations can be given in the context of some route planning problems and transportation of goods perishable by time. Garfinkel and Gilbert [60], Carpaneto, Martello and Toth [28], and Sergeev and Chernyshenko [140] developed specialized branch and bounds algorithm to solve BTSP. Computational results based on instances with n less than or equal 200 are reported in [60, 28].

The BTSP can be modelled as follows. Let $G = (V, E)$ be a complete undirected graph with vertex set $V := \{v_1, \ldots, v_n\}$ and edge set $E := \{[i, j] : i, j \in V, i \neq j\}$. Let c_e be the cost associated with each edge $e \in E$. A simple cycle is called *feasible* initial vertex if it goes through each vertex once and go back to the begining. The BTSP consist of finding a feasible cycle minimizing the maximum length of the edges belonging to the cycle. Let $x_e = 1$ if edge $e \in E$ is chosen in the optimal solution, and $x_e = 0$ otherwise. Then BTSP calls for

$$w^{BTSP} := \min \max_{e \in E} c_e x_e,$$

subject to

$$\sum_{e \in \delta(v_i)} x_e = 2 \qquad \text{for all } v_i \in V$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad \text{for all } S \subset V$$

$$x_e \in \{0,1\} \qquad \text{for all } e \in E.$$

The set constraints is identical to the TSP set of constraints.

The Maximum Scatter TSP

The *Maximum Scatter TSP* (MSTSP) is another variant that is based on the objective of finding, in a edge-weighted complete graph $G = (V, E)$, a tour that is most *scattered*.

Specifically, the goal is to maximize the length of a shortest edge in the tour, i.e., to have each point be far from the points that are visited just before of just after it in the tour. This problem is also reffered as the *max-min 1-neighbour TSP*.

The MSTSP arises in some manufacturing processes where it is important to have substantial separation between consecutive operations on a workpiece. The MSTSP also arises in some medical imaging applications. When imaging physiological functions using a Dynamic Spartial Reconstructor (DSR), the radiation sources are placed along the top half of a circular ring, with sensors placed directly opposite, in the bottom half of the ring. The firing sequence determines in which the sources, and their partnered sensors, are activated, usually in a periodic pattern. This motivated the study of firing sequence ordering for some specific geometries of DSR hardware.

To our knowledge few works deal with this problem. Penavic [124] studies the optimal firing for a DSR application in which all sources are equally spaced. Arking, Chiang, Mitchell, Skiena and Yang [5] show that the MSTSP is $\mathcal{NP}$-hard and give some approximation algorithms for the case of Euclidean distances. In a similar way that the TSP, let us define $G = (V, E)$ with vertex set $V := \{v_1, \ldots, v_n\}$ and edge set $E := \{[i, j] : i, j \in V, i \neq j\}$. For each $e \in E$ c_e is the routing cost of the edge e. Let $x_e = 1$ if edge $e \in E$ is chosen in the optimal solution, and $x_e = 0$ otherwise. MSTSP can be modelled as follows.

$$w^{scatterTSP} := \max \min_{e \in E} c_e x_e,$$

subject to

$$\begin{aligned} \sum_{e \in \delta(v_i)} x_e &= 2 && \text{for all } v_i \in V \\ \sum_{e \in \delta(S)} x_e &\geq 2 && \text{for all } S \subset V \\ x_e &\in \{0, 1\} && \text{for all } e \in E. \end{aligned}$$

The TSP with time windows

The *Traveling Salesman Problems with time windows* (ATSP-TW), that can be defined as follows. Consider a undirected graph $G = (V \cup \{v_0\}, E)$ on $n+1$ nodes. Vertex v_0 is the starting vertex (depot) for a salesman. With each edge $e \in E$, an edge duration $c_e > 0$ is associated. Furthermore, assume that for each vertex $v_i \in V$, a processing time $p_i \geq 0$, a release time $r_i \geq 0$, and a due date $d_i \geq r_i$ are given. The release date r_i denotes the earliest possible (and the due date d_i the latest possible) starting time for visiting vertex $v_i \in V$. For the depot vertex v_0 we assume that $r_0 = d_0 = 0$. The processing time p_i represents the elapsed time between the arrival and departure at vertex v_i. The interval $[r_i, d_i]$ is called the *time window* of the vertex v_i. The width of the time windows is given by $d_i - r_i$. The time window for vertex v_i is called *active*, if $r_i > 0$ or $d_i < \infty$. A time window $[0, \infty)$ is called *relaxed*. The problem is to find a sequence of the vertices (starting at the depot vertex v_0 at time 0 and ending

in vertex v_0) with minimal cost such that for every vertex $v_i \in V$ the arrival time t_i at vertex v_i lies within the given time windows $[r_i, d_i]$. It is also assumed that one may arrive at a vertex $v_i \in V$ earlier than r_i and wait until the vertex is released at time r_i. Waiting has no influence on the cost of a solution.

The TSP-TW reduces to the TSP if $p_i = 0$, $r_i = 0$, and $d_i = +\infty$, for every v_i. Therefore the TSP-TW with general time windows is $\mathcal{NP}$-hard. Tsitsiklis [148] showed that the TSP-TW with general time windows is strongly $\mathcal{NP}$-complete, even if the underlying graph G is a path and all processing times are equal 0.

A dynamic programming algorithm for TWP-TW was presented by Dumas, Desrisuers, Gelinas and Solomon [47], able to solve problems up to 200 nodes. Balas and Simonetti [14] presented a new dynamic programming algorithm that can be applied to a wide class of restricted TSP's. This approach yields good results on the asymmetric TSP-TW. Bianco, Mingozzi and Ricciardelli [21] presented a dynamic programming algorithm for TSP-TW with precedent constraints and presented computational results for instances up to 120 nodes. Finally, Ascheuer, Fischetti and Grötschel [6] developed a Branch-and-Cut algorithm for the asymmetric TSP-TW. For surveys in time constrained routing and scheduling problems see [45, 46], among others.

The following model is defined on binary edge variables.Let $x_e = 1$ if edge $e \in E$ is chosen in the optimal solution, and $x_e = 0$ otherwise. Then the TSP-TW can be modelled as

$$w^{TSP-TW} := \min \sum_{e \in E} c_e x_e,$$

subject to

$$\sum_{e \in \delta(v_i)} x_e = 2 \qquad \text{for all } v_i \in V \cup \{v_0\} \tag{2.3}$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad \text{for all } S \subseteq V \tag{2.4}$$

$$\sum_{e \in P} x_e \leq |P| - 1 \qquad \text{for all infeasible path } P \tag{2.5}$$

$$x_e \in \{0, 1\} \qquad \text{for all } e \in E. \tag{2.6}$$

Inequalities (2.5) forbid infeasible path, i.e., path violating the given time windows. Therefore, each solution x of (2.3)–(2.6) is the incidence vector of a feasible Hamiltonian tour, and vice versa. Constraints (2.3) and (2.4) are exactly the same that the degree constraints and sub tour elimination constraints in the TSP.

The Cycle Problem

The *Cycle Problem* is the problem of finding a minimum weight circuit (i.e. a simple cycle) in an undirected (directed) graph $G = (V, E)$ ($G = (V, A)$) with cost c_e (c_a) associated with each edge $e \in E$ (arc $a \in A$). This problem is in general $\mathcal{NP}$-hard, since the TSP can be reduce to it by substracting a large positive constant from each edge.

However, polynomial solvable cases can be obtained if restriction on the graphs or the cost vector are considered. If we have nonnegative costs or if the costs of the edges of each circuit sum up to a nonnegative number, the weighted problem is solvable in polynomial time (see Coullard and Pulleyblank [37]).

The undirected and Directed Cycle Polytope has been studied by Bauer [18] and Balas and Oosten [13] respectively. In addition, a polyhedral study for the Cycle Polytope with loop variables can be found in Salazar [136] for the undirected case, and in Balas [8] for the directed case (referred as the P_0 polytope).

A model for the directed cycle problem described in Balas [8] is described below. The following model is defined on binary edge variables and binary loop variables.Let $x_e = 1$ if edge $e \in E$ is chosen in a feasible solution of the cycle problem, and $x_e = 0$ otherwise. The variable y_i is equal to 1 if the vertex v_i belong to the solution and equal to 0 otherwise.

$$\begin{aligned}
\sum_{a \in \delta^+(v_i)} x_a = y_i \qquad & \text{for all } v_i \in V \\
\sum_{a \in \delta^-(v_i)} x_a = y_i \qquad & \text{for all } v_i \in V \\
\sum_{a \in \delta^+(S)} x_a \geq y_i \qquad & \text{for all } S \subset V \\
x_a \in \{0,1\} \qquad & \text{for all } a \in A \\
y_i \in \{0,1\} \qquad & \text{for all } v_i \in V.
\end{aligned}$$

The Generalized TSP

The *Generalized Traveling Salesman Problem* (GTSP) can be defined as follows. We are given a complete undirected graph $G = (V, E)$ with vertex set $V := \{v_1, \ldots, v_n\}$ and edge set $E := \{[i, j] : i, j \in V, i \neq j\}$. In addition, a proper partition $M_1, \ldots, M_m$ of V is given in which each vertex subset is called *cluster*. Let c_e be the cost associated with each edge $e \in E$. A simple cycle is called *feasible* if it goes through each cluster at least once. GSTP consists in finding a feasible cycle $T \subset E$ whose global cost $\sum_{e \in T} c_e$ is minimum. The problem involves two related decisions:

(i) choosing a vertex subset $S \subseteq V$, such that $|S \cap M_k| \geq 1$ for $k = 1, \ldots, m$;

(ii) finding a minimum cost Hamiltonian cycle in the subgraph of G induced by S.

A different version of the problem, called E-GTSP (where E stands for Equality), arises when imposing the additional constraint that exactly one vertex of each cluster must be visited. Notice that GTSP and E-GTSP are equivalent when the costs satisfy the triangle inequality, i.e., $c_{ij} \leq c_{ik} + c_{kj}$ for all vertex triples (v_i, v_j, v_k).

Both GTSP and E-GTSP are clearly $\mathcal{NP}$-hard, as they reduce to TSP when $m = n$, i.e., $|M_h| = 1$ for all h. They have been studied, among others, by Laporte and

Norbert [100], Salazar [135], Fischetti, Salazar and Toth [54] and [53], and Sepehri [139]. Their asymmetric counterparts have been investigated in Laporte, Mercure and Nobert [98] and Noon and Bean [115].

An integer linear programming model for GTSP is as follow. Let $x_e = 1$ if edge $e \in E$ is chosen in the optimal solution, and $x_e = 0$ otherwise. In addition, let $y_i = 1$ if the vertex $v_i \in V$ is visited, and $y_i = 0$ otherwise. GTSP then calls for

$$w^{GTSP} := \min \sum_{e \in E} c_e x_e,$$

subject to

$$\sum_{e \in \delta(v_i)} x_e = 2y_i \qquad \text{for all } v_i \in V \tag{2.7}$$

$$\sum_{v_i \in M_h} y_i \geq 1 \qquad \text{for all } h := 1, \ldots, m \tag{2.8}$$

$$\sum_{e \in \delta(S)} x_e \geq 2(y_i + y_j - 1) \qquad \text{for all } S \subset V, 2 \leq |S| \leq n-2, \; v_i \in S, v_j \in V \setminus S \tag{2.9}$$

$$x_e \in \{0, 1\} \qquad \text{for all } e \in E \tag{2.10}$$

$$y_i \in \{0, 1\} \qquad \text{for all } v_i \in V. \tag{2.11}$$

Constraints (2.7) impose that the number of edges incident with a vertex is either 2 (if v_i is visited) or 0 (otherwise). Constraints (2.8) force at least one vertex in each cluster to be visited. Inequalities (2.9) are connectivity constraints saying that each cut separating two visited vertices (v_i and v_j) must be crossed at least twice by the cycle.

The Orienteering Problem

The *Orienteering Problem*(OP) can be defined as follows. Given a set of n cities, each having an associated nonnegative prize, and a vehicle stationed in a depot located in city v_1. Let $c_{ij} = c_{ji}$ be the time spent for routing cities v_i and v_j in sequence. The OP is to find a route for a vehicle, visiting each city at most once, requiring a total time not exceeding a given bound c_0, and collecting a maximum total prize. This problem is $\mathcal{NP}$-hard, and arises in several routing and scheduling applications, see e.g., Golden, Levy, and Vohra [74].

Heuristic Algorithms for OP and some generalizations have been proposed by Tsiligires [147], Golden, Levy and Vohra [74], Golden, Wang, and Liu [75] and Chao, Golden, and Wasil [29]. Exact enumerative methods have been proposed by Laporte and Martello [97], and by Ramesh, Yoon, and Karwan [127]. Leifer and Rosenwein [105] have discussed an LP-based bounding procedure. Gendreau, Laporte, and Semet [64] proposed a branch-and-cut aproach. Finally Fichetti, Salazar and Toth [55] introduced a new family of cuts and developed a branch-and-cut for this problem.

We consider a complete graph $G = (V, E)$ with $n := |V|$ nodes. Vertex v_1 represent the depot. Let p_i denote the nonnegative prize associated with each $v_i \in V$ (with $p_1 = 0$), c_e be the nonnegative travel time associated with any $e \in E$, and c_0 be the maximum total travel time allowed for the vehicle.

We assume throughout that all values p_i, c_e, and c_0 are integer.

OP can then be formulated as the 0–1 Integer Linear Programming model:

$$w^{OP} := \max \sum_{v_i \in V} p_i y_i,$$

subject to

$$\sum_{e \in \delta(v_i)} t_e x_e \leq t_0 \tag{2.12}$$

$$\sum_{e \in \delta(v_i)} x_e = 2y_i \qquad \text{for all } v_i \in V \tag{2.13}$$

$$\sum_{e \in \delta(S)} x_e \geq 2y_i \qquad \text{for all } S \subset V, \quad v_i \in S, \quad v_1 \in V \setminus S \tag{2.14}$$

$$y_1 = 1 \tag{2.15}$$

$$x_e \in \{0, 1\} \qquad \text{for all } e \in E \tag{2.16}$$

$$y_i \in \{0, 1\} \qquad \text{for all } v_i \in V \setminus \{v_1\}. \tag{2.17}$$

Constraints (2.12) impose the bound t_0 on the total travel time. The degree equations (2.13) stipulate that a feasible solution has to go exactly once through each visited node. The Generalized subtour Elimination Constraints (2.14) force each visited vertex $v_i \in V \setminus \{v_1\}$ to be reachable from vertex v_1 by mean two edge-disjoint paths. Finally, (2.15) imposes that vertex v_1 must be visited, and (2.16)–(2.17) require that all variables are 0–1 valued.

The Vehicle Routing Problem

Vehicle Routing Problems VRP deal with the optimal use of a fleet of vehicles to transport (pick up or deliver) goods between a central depot and a set of clients. Several interesting examples arise in scheduling school buses, mail collection from the mail-boxes, delivery of laundry, garbage collection, etc. Because of this enormous number of practical applications several particular versions have been studied in the literature. For surveys on the subject see, e.g., Christofides, Mingozzi and Toth [32], Christofides [31], Laporte and Nobert [101], Golden and Assad [72] and Laporte [96]. We consider now the *Capacitated Vehicle Routing Problem* (CVRP), introduced by Dantzig and Ramser [43]. In this problem a quantity d_i of a single commodity is to be delivered to each customer $v_i \in V$, from a central depot v_0 using k independent delivery vehicles of identical capacity Q. Delivery is to be accomplished at minimum total cost, with $c_{ij} \geq 0$ denoting the transit cost from v_i to v_j, for $v_i, v_j \in V$. The cost structure is assumed to be symmetric, i.e., $c_{ij} = c_{ji}$ and $c_{ii} = 0$.

Combinatorially, a solution for this problem consists of a partition $\{R_1, \ldots, R_k\}$ of V into k routes, each satisfying $\sum_{v_j \in R_i} d_j \leq Q$ for $i = 1, \ldots, k$, and a corresponding permutation σ_i of each route specifying the service ordering. This problem is naturally associated with the complete undirected graph $G = (V \cup \{v_0\}, E)$, and the edge-traversal costs c_e for all $e \in E$. In this graph, a solution is the union of k cycles whose only intersection is the depot node. Each cycle corresponds to the route serviced by one of the k vehicles. By associating a binary variable with each edge in the graph, we obtain the following ILP formulation.

$$w^{CVRP} := \min \sum_{e \in E} c_e x_e,$$

subject to

$$\sum_{e \in \delta(v_0)} x_e = 2k \tag{2.18}$$

$$\sum_{e \in \delta(v_i)} x_e = 2 \qquad \text{for all } v_i \in V \tag{2.19}$$

$$\sum_{e \in \delta(S)} x_e \geq 2b(S) \qquad \text{for all } S \subset V, \quad |S| > 1 \tag{2.20}$$

$$x_e \in \{0, 1\} \qquad \text{for all } e \in E \setminus \{\delta(v_0)\} \tag{2.21}$$

$$x_e \in \{0, 1, 2\} \qquad \text{for all } e \in \delta(v_0). \tag{2.22}$$

We define $b(S) = \lceil \left(\sum_{v_i \in S} d_i\right) / Q \rceil$, an obvious lower bound on the number of vehicles needed to service the customers in set S. Contraints (2.18) and (2.19) are the degree constraints. Constraints (2.20) can be viewed as a generalization of the subtour elimination constraints from the TSP and enforce the connectivity of the solution as well as to ensure that no route has total demand exceeding the capacity Q.

The Median Cycle Problem

The *Median Cycle Problem* (MCP) can be defined as follow. Let $G = (V, E \cup A)$ be a complete mixed graph where $V = \{v_1, \ldots, v_n\}$ is the vertex set, $E = \{[v_i, v_j] : v_i, v_j \in V, i < j\}$ is the edge set, and $A = \{(v_i, v_j) : v_i, v_j \in V\}$ is the arc set (loops (v_i, v_i) are included in A). Vertex v_1 is referred to as the depot. With each edge $[v_i, v_j] \in E$ is associated a non-negative routing cost c_{ij}, and with each arc $(v_i, v_j) \in A$ is associated a non-negative assignment cost d_{ij}. A solution to the MCP is a simple cycle through a subset V' of V including v_1 and at least two other vertices. The routing cost of a solution is the sum of the routing cost of all edges on the cycle. The assignment cost of a solution is defined as $\sum_{v_i \in V \setminus V'} \min_{v_j \in V'} d_{ij}$. Two versions of MCP have been investigated. In the first version, called MCP1 or *Ring Star Problem*, the aim is to determine a solution so as to minimize the sum of the routing cost and the assignment cost. In the second version, called MCP2, we seek a solution of least routing cost, subject to an upper bound d_0 on the assignment cost.

A deep treatment of MCP, including both exact and heuristics algorithms, is made in the PhD disertation of Rodríguez [133]. Lee, Chiu and Sanchez [104] defined a very closely related problem to MCP1 by considering an additional set of vertices W and setting the assignment cost to $\sum_{v_i \in W} \min_{v_j \in V'} d_{ij}$. They developed a branch-and-cut solving intances with $|V| \leq 50$, $|W| \leq 90$ and $|V| + |W| \leq 100$. Xu, Chiu and Glover [153] proposed a tabu search aproach for this problem. Recently Labbé, Laporte, Rodríguez and Salazar [95] and [94] developed a branch-and-cut approach for MCP1 and MCP2 involving instances up to $|V| \leq 300$ and up to $|V| \leq 150$ respectively.

MCP1 can be formulate as an ILP model as follows. For each edge $[v_i, v_j] \in E$, let x_{ij} be a binary variable equal to 1 if and only if edge $[v_i, v_j]$ appears on the cycle. For each arc $(v_i, v_j) \in A$, let y_{ij} be a binary variable equal to 1 if and only if vertex v_i is assigned to vertex v_j on the cycle. Notice that if a vertex v_i is on the cycle, it is then assigned to itself, i.e., $y_{ii} = 1$. The formulation is then:

$$w^{MCP1} := \min \sum_{[v_i,v_j] \in E} c_{ij} x_{ij} + \sum_{(v_i,v_j) \in A} d_{ij} y_{ij}$$

subject to

$$\sum_{e \in \delta(v_i)} x_e = 2y_{ii} \qquad \text{for all } v_i \in V \tag{2.23}$$

$$\sum_{v_j \in V} y_{ij} = 1 \qquad \text{for all } v_i \in V \setminus \{v_1\} \tag{2.24}$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \sum_{v_j \in S} y_{ij} \qquad \text{for all } S \subset V, \quad v_1 \notin S, \quad v_i \in S \tag{2.25}$$

$$x_{ij} \in \{0, 1\} \qquad \text{for all } [v_i, v_j] \in E \tag{2.26}$$

$$y_{ij} \geq 0 \qquad \text{for all } (v_i, v_j) \in A \tag{2.27}$$

$$y_{11} = 1 \tag{2.28}$$

$$y_{1j} = 0 \qquad \text{for all } v_j \in V \setminus \{v_1\} \tag{2.29}$$

$$y_{jj} \text{ integer} \qquad \text{for all } v_j \in V \setminus \{v_1\}. \tag{2.30}$$

In this formulation, Contraints (2.23) are degree constraints. They ensure that the degree of a vertex v_i is 2 if and only if it belongs to the cycle (i.e., $y_{ii} = 1$). Constraints (2.24) state that either v_i is a vertex on the cycle (in that case y_{ii}), or v_i is assigned to a vertex v_j on the cycle (in that case $y_{ij} = 1$). Constraints (2.25) are connectivity constraints since they state that S must be connected to it complement by at least two edges of the cycle whenever at least one vertex v_i is assigned to $v_j \in S$.

The ILP formulation of MCP2 is identical to that of MCP1 except that the objective becomes

$$w^{MCP2} := \min \sum_{[v_i,v_j] \in E} c_{ij} x_{ij}$$

and the constraint

$$\sum_{\substack{(v_i, v_j) \in A \\ v_i \neq v_j}} d_{ij} y_{ij} \leq d_0$$

is introduced.

The Prize Collecting TSP

The *Prize Collecting TSP* (PCTSP) was introduced by Balas [8] and can be described as follows. Let us consider a depot at which a vehicle is stationed, and a set of cities, each one having a non-negative prize w_i for each city v_i when visited, and a penalty c_i to every city v_i when no visited. With a traveling cost of c_a for all arcs $a = (v_i, v_j)$ joining cities v_i and v_j. This problem looks for a tour that minimizes the travel cost and penalties, subject to a lower bound w_0 on the amount of prize money it collects. A branch and bound method for the exact solution of PCTSP was developed by Fischetti and Toth [57]. Polyhedral results was obtained by Balas [8, 9].

Let y_i be 1 if city v_i is included into the tour and 0 otherwise, and let x be the incidence vector of the arcs in the tour, then PCTSP can be formulated on a complete directed graph $G' = (V, A)$

$$w^{PC'} := \min \sum_{a \in A} c_a x_a + \sum_{v_i \in V} c_i y_i$$

subject to

$$\begin{aligned}
\sum_{a \in \delta^+(v_i)} x_a = y_i & \qquad \text{for all } v_i \in V \\
\sum_{a \in \delta^-(v_i)} x_a = y_i & \qquad \text{for all } v_i \in V \\
\sum_{v_i \in V} w_i y_i \geq w_0 & \\
x_a \in \{0,1\} & \qquad \text{for all } a \in A \\
y_i \in \{0,1\} & \qquad \text{for all } v_i \in V \\
G'(y,x) \text{ is a cycle.} &
\end{aligned}$$

Here $G'(y, x)$ is the subgraph G' whose vertices and arcs are those defined by y and x, respectively. As also happens in the definition of the *Median Cycle Problem*, it is convenient to complement the variables y_i, $v_i \in V$, i.e. introduce n new variables $x_{ii} = 1 - y_i$, $v_i \in V$, representing loops of a graph $G = (V, A \cup O)$ obtained from G' by endowing every vertex with a loop. Now, the incidence vector $(y, x) \in \{0,1\}^{n^2}$ of vertices and arcs of G' is replaced by the incidence vector $x \in \{0,1\}^{n^2}$ of loops and arcs of G. If we define $c_{ii} := p_i$, $v_i \in V$, and $U := \sum_{v_i \in V} w_i - w_0$, the problem can be restated as follow

$$w^{PC} := \min \sum_{a \in A \cup O} c_a x_a$$

subject to

$$\sum_{a \in \delta(v_i)^+} x_a + x_{ii} = 1 \qquad \text{for all } v_i \in V$$

$$\sum_{a \in \delta(v_i)^-} x_a + x_{ii} = 1 \qquad \text{for all } v_i \in V$$

$$\sum_{v_i \in V} w_i x_{ii} \leq U$$

$$x_a \in \{0, 1\} \qquad \text{for all } a \in A \cup O$$

$G(x)$ is a cycle of length ≥ 2.

The Covering Tour Problem

The *Covering Tour Problem* (CTP) is defined as follows. Let $G = (V \cup W, E)$ be an undirected graph, where $V \cup W$ is the vertex set, $V = \{v_1, \ldots, v_n\}$ and $E = \{[v_i, v_j] : v_i, v_j \in V \cup W, i < j\}$ is the edge set. Vertex v_1 is a depot, V is a set of vertices that can be visited, $T \subseteq V$ is a set of vertices that must be visited ($v_1 \in T$), and W is a set of vertices that must be covered. For each edge $[v_i, v_j] \in E$ a distance c_{ij} is defined. The CTP consists in determining a minimum lenght tour of Hamiltonian cycle over a subset of V in such a way that the tour contains all vertices T, and every vertex of W is covered by the tour. Such tour may not always exist. CTP was introduced by Current [39]. It is formulated in Current and Schilling [41]. In [41] a two-objective version of the problem is also considered. The authors propose a heuristic to generate a set of efficient solutions. Recently Gendreau, Laporte and Semet [65] developed an exact branch-and-cut algorithm for this problem.

The CTP can be formulated as an ILP model as follows. For each $v_k \in V$, let y_k be a binary variable equal to 1 if and only if the vertex v_k belongs to the tour. If $v_k \in T$, then y_k is neccesary equal to 1. For $v_i, v_j \in V$ and $i < j$, let x_{ij} a binary variable equal to 1 if and only if edge $[v_i, v_j]$ belongs to the tour. Also define binary coefficients δ_{lj} equal to 1 if and only if $v_l \in W$ can be covered by $v_k \in V$, and let $S_l = \{v_k \in V | \delta_{lk} = 1\}$ for every $v_l \in W$. Then the CTP can be stated as:

$$w^{CTP} := \min \sum_{e \in E} c_e x_e$$

subject to

$$\sum_{v_i \in S_l} y_i \geq 1 \qquad \text{for all } v_l \in W \tag{2.31}$$

$$\sum_{e \in \delta(v_i)} x_e = 2y_i \qquad \text{for all } v_i \in V \tag{2.32}$$

$$\sum_{e \in \delta(S)} x_e \geq 2y_i \qquad S \subset V, 2 \geq |S| \geq n-2, \; T \setminus S \neq \varnothing, v_i \in S \tag{2.33}$$

$$x_e \in \{0,1\} \qquad \text{for all } e \in E \tag{2.34}$$

$$y_i = 1 \qquad \text{for all } v_i \in T \tag{2.35}$$

$$y_i \in \{0,1\} \qquad \text{for all } v_i \in V \setminus T. \tag{2.36}$$

In this formulation, Constraints (2.31) ensure that every vertex of W is covered by the tour, while Constraints (2.32) are the degree contraints. Constraints (2.33) are connectivity constraints. They force the presence of at least two edges between any set S and $V \setminus S$, for every proper subset $S \subseteq V$ such that $T \setminus S \neq \varnothing$ and S contains a vertex v_t belinging to the tour. Finally, constraints (2.34)–(2.36) set the integrality requirements.

The Pickup-and-Delivery TSP

There are different versions of the *Pickup-and-Delivery Traveling Salesman Problem* in the literature. They all concerns with the collection and delivery of some products a each customer, and differes in the number of product to be transported, the existence of precence constraints, the existence of time windows, the existence of capacities, diffeent vehicles, etc. The *one-commodity Pickup-and-Delivery Traveling Salesman Problem* (1PDTSP) is a generalization of the TSP in which a special city is considered as the *depot*, and the other cities as *customers* partitioned into two groups according to the type of the required service. Each *derivery customer* requires a given non-zero amount of the product, while each *pickup customer* provides a given non-zero amount of the product. The product collected in a pickup customer can be served to a delivery customer, as it is assumed non-deterioration for the use of the product. It is also considered a vehicle with a fixed upper-limit *capacity* that starts and ends the route at the depot, and the classical travel distances between each pair of locations. Then the 1PDTSP calls for a minimum distance tour for the vehicle visiting each customer once and satisfying the customer requirements without ever violating the vehicle capacity.

On the other hand, there is another variant known in the literature as the *Traveling Salesman Problem with Pickups-and-Delivery* (TSPPD). As in the 1-PDTSP, there are two types of customers, each one with a given demand, and a vehicle with a given capacity originally in the depot. Also travel distances are given. The main difference between the two problems is that in the TSPPD the product collected from pickup customer is different than the product served to delivery customers. Therefore, the total amount of product collected from pickup customers must be delivered only at the depot, and there is another different product going from the depot to the delivery customers.

The TSPPD was introduced by Mosheiov [111], who proposed applications and heuristic approaches. Anily and Mosheiov [3] and Gendreau, Laporte and Vigo [66] present approximation algorithms for the TSPPD. Anily and Hassin [2] introduce the *Swapping Problem*, the particular case of 1-PDTSP in which the customer demands

and vehicle capacity are all identical numbers, and propose an approximation algorithm. Finally, Hernández and Salazar [85] propose a 0-1 integer linear programming model, as well as a Branch-and-Cut approach for solving the 1-PDTSP and TSPPD for both the symmetric case.

We now present an ILP model for the 1-PDTSP. Let $G = (V, E, A)$ an undirected (directed) graph. Let $V := \{v_1, v_2, \ldots, v_n\}$ a the set of vertices and let E be the edge set, in which each edge $e = [v_i, v_j] \in E$ joins the vertex v_i and v_j; and let A be the arc set, consisting of the arcs $(v_i, v_j) \in A$ joining the vertices v_i and v_j. The depot is denoted by the vertex v_1, and the customers by the remaining vertices. For each customer $v_i \in V \setminus \{v_1\}$ the demand q_i is given, inducing a *delivery customer* if $q_i < 0$ and a *pickup customer* if $q_i > 0$. The capacity of the vehicle is represented by Q, and it is assumed to be a positive number. For each pair of vertices $v_i, v_j, \in V$ a travel distance c_{ij} is also given. Let us consider the symmetric case, in which $c_{ij} = c_{ji}$.

Without lost of generality, the depot can be considered a customer by defining $q_1 := -\sum_{v_i \in V \setminus \{v_1\}} q_i$.

Let us consider the for each edge $e \in E$ the edge-decision variable x_e that is equal to 1 if and only if edge e belongs to the cycle and 0 otherwise. Let us also consider the continuous variable g_a certificating the existence of a load of the vehicle going through arc a. Then, the symmetric case can be formulated as follows.

$$w^{1PDTSP} := \min \sum_{e \in E} c_e x_e$$

subject to

$$\sum_{e \in \delta(v_i)} x_e = 2 \qquad \text{for all } v_i \in V \tag{2.37}$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad S \subset V \tag{2.38}$$

$$x_e \in \{0, 1\} \qquad \text{for all } e \in E \tag{2.39}$$

$$\sum_{a_\in \delta^+(v_i)} g_a - \sum_{a_\in \delta^-(v_i)} g_a = q_i \qquad \text{for all } v_i \in V \tag{2.40}$$

$$0 \leq g_{(i,j)} \leq \frac{Q}{2} x_{ij} \qquad \text{for all } (i, j) \in A. \tag{2.41}$$

Constraints (2.37) impose that each customer must visited once, and Constraints (2.38) force the 2-connectivity between customers. Constraints (2.40) and (2.41) guarantee the existence of a certificate $[g_a : a \in A]$ proving that a vector $[x_e : e \in E]$ defines a feasible 1-PDTSP cycle.

The Black and White TSP

The *Black and White Traveling Salesman Problem* (BWTSP) is first analyzed in Ghiani, Laporte and Semet [67] and Bourgeois, Laporte and Semet [24]. A BWTSP is defined on a graph $G = (V, E)$ where vertex set V is partitioned into black nodes,

denoted by $\mathcal{B}$, and white nodes, denoted by $\mathcal{W} = V \setminus \mathcal{B}$. A solution of a BWTSP is a tour in G, satisfying some additional constraints. $\mathcal{W}$ defines a "black-to-black" path to be a sequence of vertices $(v_{i_1}, \ldots, v_{i_k})$ where $v_{i_1}, v_{i_k} \in \mathcal{B}$, and $v_{i_2}, \ldots, v_{i_{k-1}} \in \mathcal{W}$. A feasible BWTSP tour must comply the following conditions. A feasible tour cannot use more than $Q \in \mathbb{Z}^+$ white vertices in any black-to-black path, which we refer to as cardinality constraints; the total arc cost of any black-to-black path between two black vertices cannot exceed a fixed value, denoted by $L \in \mathbb{R}^+$, which refer to as cost constraints.

Ghiani, Laporte and Semet [67] present a branch-and-cut for finding exact solutions for the undirected BWTSP. They also introduce several classes of valid cuts, which can be classified into two groups: cardinality constraints and cost constraints. This branch-and-cut algorithm was tested on a large number of randomly generated test problems with different characteristics with respect to Q, $|\mathcal{B}|$ and L. The method seems to perform well on problem with no restriction cost, that is with $L = \infty$, and with cardinality loosely constrained. For this type of problems, the largest instances solved have 100 nodes. However, for problems with tightly constrained cardinality and cost, the largest problems successfully solved are some 20 vertices instances.

Bourgeois, Laporte and Semet [24] present heuristic approaches for obtaining feasible solutions and thus upper bounds to undirected BWTSP's. The heuristic approaches introduced by them can be described as follows. First, a TSP feasible solution is obtained using the GENIUS heuristic described by Gendreau, Hertz and Laporte [62]. If the solution is infeasible to the BWTSP, then some swapping heuristics and insertion heuristics are applied in an attempt to remove infeasibilities (they refer to these heuristics as feasibility heuristics). Bourgeois, Laporte and Semet [24] point out that these heuristics do not guarantee feasible solutions. Then, if a feasible solution is obtained, a somewhat restricted 2-opt procedure that only allows moves within some neighbourhood of feasible solutions is applied in order to obtain better feasible solutions.

2.4 TRANSFORMATION OF THE STPP INTO THE GTSP

A transformation to solve the TPP is presented in this section. This transformation allow us to solve the STPP by mean a GTSP solver (see Fischetti, Salazar and Toth [53] for an exact algorithm for the symmetric GTSP).

The unlimited-supply TPP can readily be formulated as a *generalized TSP* with equality constrains (E-GTSP) consisting in designing in an auxiliary graph $G' = (V', E')$ a minimum-cost simple cycle visiting exactly one vertex of each given cluster (see, e.g., Fischetti, Salazar and Toth [53]). More specifically, the auxiliary graph G' is obtained as follows. Let v_{ki} represent a *stall* at which product p_k can be purchased at market v_i. Then $V' := \{v_0\} \cup \{v_{ki} : \ p_k \in K, \ v_i \in M_k\}$ is the vertex set and $E' := \{[v_{ki}, v_{hj}] : \ v_{ki}, v_{hj} \in V', k < h\}$ is the edge set. The cost of using edge $[v_{ki}, v_{hj}] \in E'$ is zero if $i = j$, and c_{ij} otherwise; the cost of visiting v_0 is zero and the cost of visiting v_{ki} is b_{ki} (Fig. 2.2 and 2.3 illustrate this transformation for both the symmetric and asymmetric case). The corresponding E-GTSP is then

defined with the clusters $C_0 := \{v_0\}$ and $C_k := \{v_{ki} : v_i \in M_k\}$ for all $p_k \in K$. Clearly an optimal solution of this E-GTSP gives a sequence of stalls to purchase each product in one stall of one market with minimum pricing and routing costs, i.e., an optimal solution of the unlimited-supplied TPP. In a similar way, it could be possible to transform a general TPP in a GTSP visiting *at least* one node for each cluster. Since in general $|V'|$ is much larger than $|V|$, using this transformation for solving TPP is likely to be rather inefficient.

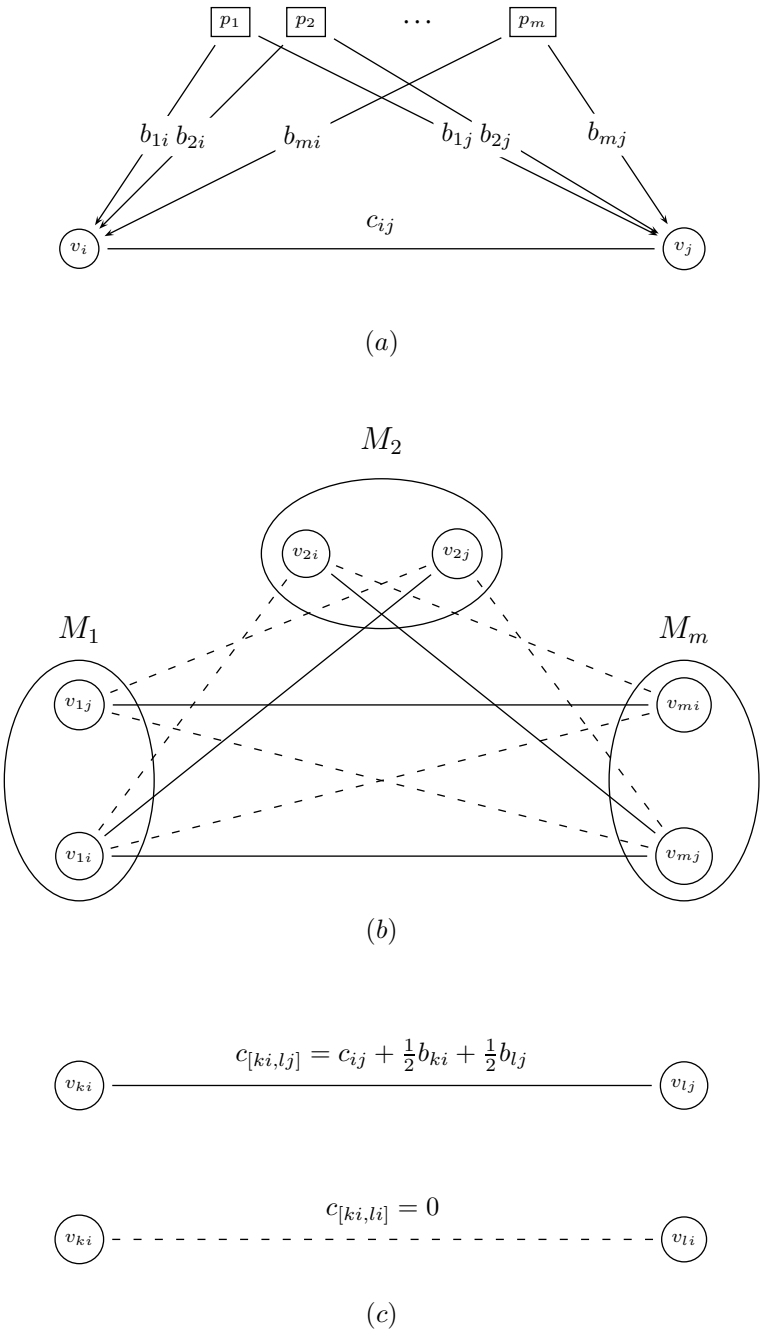
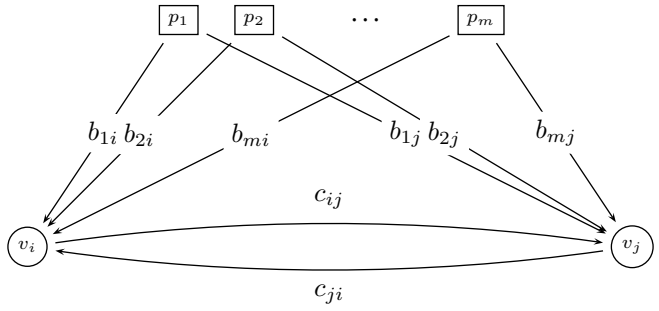
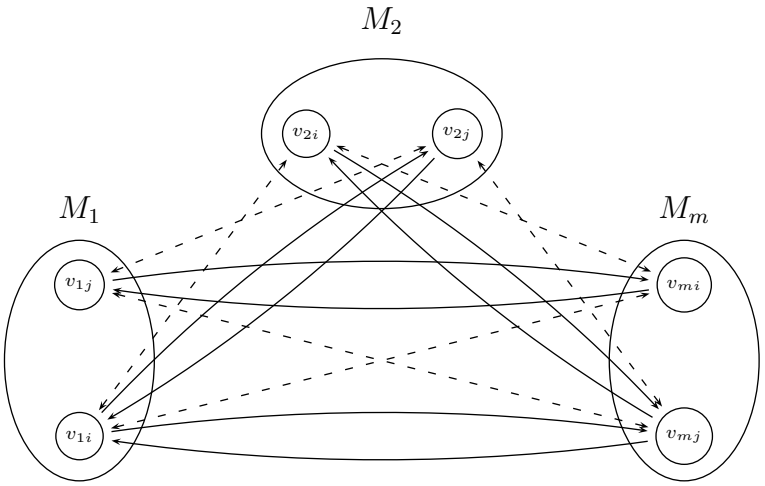


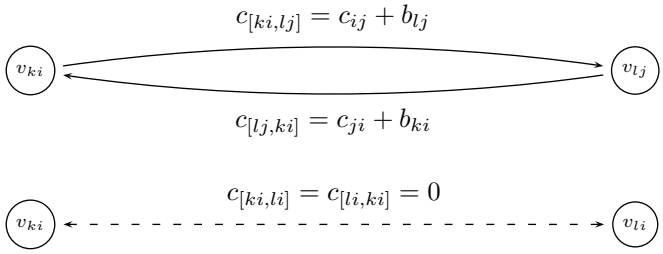
Fig. 2.2 Transformation of the STPP into GTSP.



(a)



(b)



(c)

Fig. 2.3 Transformation of the ATPP into the GTSP.

3

Mathematical Models

The purpose of this chapter is to provide a model for both versions of the TPP, the symmetric and asymmetric, and to introduce the notation used throughout the chapter. Sections 3.1 and 3.3 describe the Integer Programming model proposed for the symmetric and asymmetric cases respectively. Sections 3.2 and 3.4 provide additional inequalities which strength the LP-relaxation of the proposed models.

The aim in the Traveling Purchaser Problem is to determine a route among a set of selected markets as well as an assignment of the product satisfying the demand constrains. The given set of potential *markets* $M := \{v_1, \ldots, v_n\}$ with the *depot* v_0, and the possible connections between each pair of them can be represented formally by a undirected (directed) graph $G = (V, E)$ ($G = (V, A)$), where $V := \{v_0\} \cup M$ is the vertex set and $E := \{[v_i, v_j] : v_i, v_j \in V, i < j\}$ ($A := \{[v_i, v_j] : v_i, v_j \in V\}$) is the edge (arc) set, representing all the possible pair of nodes with direct connection.

Each *product* $p_k \in K := \{p_1, \ldots, p_m\}$ is available at a subset $M_k \subseteq M$ of markets. We will assume for convenience that $n \geq 4$ and $m \geq 1$. Let us denote by d_k the number of units of product p_k that must be purchased, and let q_{ki} be the number of units of p_k that are available at the market v_i. We assume q_{ki} and d_k satisfy $0 < q_{ki} \leq d_k$ and $\sum_{v_j \in M_k} q_{kj} \geq d_k$ for all $p_k \in K$ and $v_i \in M_k$. Let b_{ki} be the price of the product p_k at the market v_i and the let us denote by c_e (c_a) the travel cost between v_i and v_j, where $e = [v_i, v_j]$ ($a = (v_i, v_j)$). The TPP consists in determining a simple cycle in G passing through the depot and a subset of markets so that all products are purchased at a total minimum cost obtained by adding the routing cost and the purchase price. This definition of the TPP generalizes the classical case with unlimited supplies, i.e., $q_{ki} = d_k$ for all i, k.

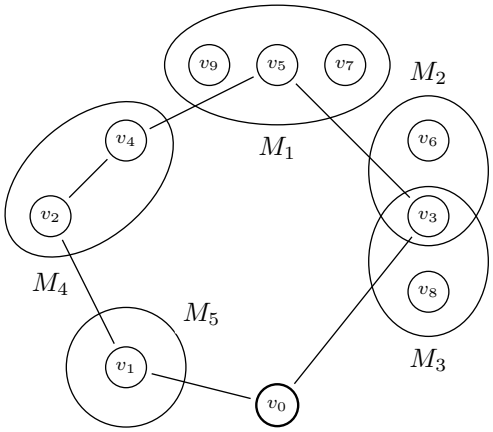


Fig. 3.1 Feasible solution of TPP.

Fig. 3.1 shows an instance of TPP containing 9 markets $M := \{v_1, \ldots, v_9\}$ and 5 type of different products $K := \{p_1, \ldots, p_5\}$. Notice that, both the products p_2 and p_3 are available at the market v_3, since $M_2 \cap M_3 = \{v_3\}$. Moreover, in order to satisfy the demand of product p_4 it is mandatory to visit markets v_4 and v_2. This also happen with product p_5 since it is only available in market v_1. In this particular case the markets are locations in the plane, so the travel costs between two markets are given by the Euclidean distance.

The TPP is $\mathcal{NP}$-hard since it reduces to the *Traveling Salesman Problem* (TSP) when $m = n$ and $|M_k| = 1$ for all $p_k \in K$. The TPP also reduces to the *Uncapacitated Facility Location Problem* (UFLP) (see section 1.6) when $M_k = M$ for all k, $q_{ki} = d_k$ for all i, k, and $c_{ij} = (f_i + f_j)/2$ for all $[v_i, v_j] \in E$, where f_i is the cost of opening facility v_i ($f_0 := 0$) and b_{ki} is the cost of serving customer p_k from facility v_i.

3.1 ILP FORMULATION FOR THE STPP

An integer linear programming formulation for the undirected or symmetric TPP (STPP) is presented in this section. This formulation is based on the notation introduced in the previous section as well as in Section 1.1 of Chapter 1, which was related to graph theory. In addition to this, let us define

$$M^* := \{v_0\} \cup \left\{ v_i \in M : \text{ there exists } p_k \in K \text{ such that } \sum_{v_j \in M_k \setminus \{v_i\}} q_{kj} < d_k \right\},$$

as the set of vertices that must necessarily be part of any feasible TPP solution.

To model the STPP we use three types of decision variables. A first family of variables associated to edges is

$$x_e := \begin{cases} 1 & \text{if edge } e \text{ belongs to the solution} \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } e \in E.$$

A second family is defined by the variables y as follows

$$y_i := \begin{cases} 1 & \text{if vertex } v_i \text{ belongs to the solution} \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } v_i \in V.$$

It should be noticed that this variable has just the opposite sense that the loop variables defined for the *Prize Collecting TSP* by Balas in [8, 9, 10], and Balas and Oosten in [13]. In those articles the variable y is related to a graph $G(V, E, L)$, where L is the set of loops, that is, edges from a vertex to itself. And the variables y_i take the value 1 if the vertex v_i is not visited,i.e., there exist a loop in the vertex v_i. Nevertheless, other authors, as Fischetti, Salazar and Toth [53, 54, 55, 56], Rodríguez [133], Gendreau, Laporte and Semet [65], etc., make use of the former notation.

In addition, the third family z has the following meaning,

$$z_{ki} := \begin{array}{l} \text{the amount of product } p_k \text{ is purchased} \\ \qquad \text{at market } v_i, \text{ for all } p_k \in K \text{ and all } v_i \in M_k. \end{array}$$

The TPP formulation for the STPP is as follows

$$w^{OPT} := \min \sum_{e \in E} c_e x_e + \sum_{p_k \in K} \sum_{v_i \in M_k} b_{ki} z_{ki} \tag{3.1}$$

subject to

$$\sum_{e \in \delta(\{v_i\})} x_e = 2y_i \qquad \text{for all } v_i \in V \tag{3.2}$$

$$\sum_{e \in \delta(S)} x_e \geq 2y_i \qquad \text{for all } S \subseteq M \text{ and all } v_i \in S \tag{3.3}$$

$$\sum_{v_i \in M_k} z_{ki} = d_k \qquad \text{for all } p_k \in K \tag{3.4}$$

$$z_{ki} \leq q_{ki} y_i \qquad \text{for all } p_k \in K \text{ and all } v_i \in M_k \tag{3.5}$$

$$x_e \in \{0, 1\} \qquad \text{for all } e \in E \tag{3.6}$$

$$y_i \in \{0, 1\} \qquad \text{for all } v_i \in M \setminus M^* \tag{3.7}$$

$$y_i = 1 \qquad \text{for all } v_i \in M^* \tag{3.8}$$

$$z_{ki} \geq 0 \qquad \text{for all } p_k \in K \text{ and all } v_i \in M_k. \tag{3.9}$$

The objective function is to minimize the sum of both the total routing cost, given for the sum of the cost associated to the edges belonging to the solution, and the total purchasing cost, that is the sum of the selected product by their unit price.

Constrains (3.2), that are the *degree constrains*, ensure that the degree of a market v_i is 2 if the markets is visited ($y_i = 1$), i.e., each selected market is visited just once.

However, this would produce a set of subtours the following family of inequalities. Constrains (3.3), refereed as *YSEC*, and can be equivalently re-written as

$$\sum_{e \in E(S)} x_e \leq \sum_{v_j \in S \setminus \{v_i\}} y_j \qquad \text{for all } S \subseteq M \text{ and all } v_i \in S, \tag{3.10}$$

which ensure that a subset $S \subseteq M$ of visited markets must be connected to the remaining vertices by at least two edges, in order to avoid unconnected subcycles. It should be noticed that the number of *YSEC* inequalities is $O(2^n)$, that is, exponential in the size of the problem. This may seem to be a serious difficulty. However, this drawback has been overcame, since an efficient algorithm to separate this family of inequalities, based on the efficient algorithm from Padberg and Rinaldi [117] for the minimum capacity cut, has been found. It is described in section 5.4.

Inequalities (3.4) guarantee that the exact amount of product p_k is purchased. Notice that, for the particular case of the TPP, this equation could be replaced by a greater or equal inequality, since the objective function guarantee that no more than the strictly necessary amount of product would be purchased. Inequalities (3.5) mean that is not possible to purchase a product p_k in a market v_i if is not visited and that it is not possible to purchase more than it offers q_{ki}. Constrains (3.6) to (3.9) impose bounds and integrality conditions on the variables.

3.2 VALID INEQUALITIES FOR THE STPP

The linear relaxation of model (3.1)–(3.9) can be strengthened by using valid inequalities for two of its subproblems.

First observe that constrains

$$\begin{aligned}
\sum_{e \in \delta(\{v_i\})} x_e &= 2y_i \qquad \text{for all } v_i \in V \\
\sum_{e \in \delta(S)} x_e &\geq 2y_i \qquad \text{for all } S \subseteq M \text{ and all } v_i \in S \\
x_e &\in \{0,1\} \qquad \text{for all } e \in E \\
y_i &\in \{0,1\} \qquad \text{for all } v_i \in M,
\end{aligned}$$

are those of a *Cycle Problem* (see Coullard and Pulleyblank [37], Bauer [18], and Salazar [136]), a generalization of the TSP in which only a subset of vertices must be in the cycle. Thus, for example, one can impose the trivial inequalities

$$x_{[v_0,v_j]} \leq y_j \qquad \text{for all } v_j \in V. \tag{3.11}$$

Notice that (3.11) in not included in (3.10) since the $S \subseteq M$, and $v_0 \notin M$.

Other valid inequalities for the cycle part of this problem are the *2-matching inequalities* for the TSP from Edmons [48], which are a particular case of the Chvátal combs (Chvátal [34]) and of the comb inequalities (introduced and studied by Grötschel

and Padberg [79, 80]).

$$\sum_{e \in T} x_e - \sum_{e \in \delta(H) \setminus T} x_e \leq |T| - 1 \tag{3.12}$$

for all $H \subset V$ and $T \subset \delta(H)$ satisfying

(i) $\{v_i, v_j\} \cap \{v_s, v_t\} = \emptyset$ for $[v_i, v_j], [v_s, v_t] \in T$ and $[v_i, v_j] \neq [v_s, v_t]$,

(ii) $|T| \geq 3$ and odd.

These inequalities are obtained by summing up the degree equations (3.2) for all $v_i \in H$ and the bound restrictions $x_e \leq 1$ for all $e \in T$, dividing all by 2, and rounding down all coefficientes to the closest integer. See Bauer [18] for others valid inequalities for the *Cycle* polytope.

A second subproblem defined by

$$\begin{aligned} \sum_{v_i \in M_k} z_{ki} &= d_k && \text{for all } p_k \in K \\ z_{ki} &\leq q_{ki} y_i && \text{for all } p_k \in K \text{ and all } v_i \in M_k \\ y_i &\in \{0, 1\} && \text{for all } v_i \in M \\ z_{ki} &\geq 0 && \text{for all } p_k \in K \text{ and all } v_i \in M_k, \end{aligned}$$

correspond to a generalization of the UFLP with upper bounds on the customer-facility variables. Valid inequalities for the subproblem when $q_{ki} = d_k$ can be obtained from the *Set Covering Problem* polytope (see e.g., Balas and Ng [12]). Consider a subset of products $L \subseteq S$ with $3 \leq |L| \leq |K| - 1$, let $M'(L) := \cap_{p_k \in L} M_k$ be the set of markets each one selling all products in L and $M''(L) := \cup_{p_k \in L} M_k$, the set of markets each one selling at least one products in L. We can then impose the constraint

$$2 \sum_{v_i \in M'(L)} y_i + \sum_{v_i \in M''(L) \setminus M'(L)} y_i \geq 2, \tag{3.13}$$

which stipulates that at least two markets in $M''(L)$ must be visited if no market in $M'(L)$ is visited. Decomposing this second subproblem for each product yields the family of cover inequalities:

$$\sum_{v_i \in S} y_i \geq 1 \quad \text{for all } S \subseteq M_k \text{ such that } \sum_{v_i \in M_k \setminus S} q_{ki} < d_k, \tag{3.14}$$

for all $p_k \in K$. These constrains state that a market in S must be visited if markets in $M_k \setminus S$ are not enough to provide the required demand d_k of product p_k.

If we sum up (3.2) for all $v_i \in S$, we obtain

$$\sum_{e \in \delta(S)} x_e + 2 \sum_{e \in E(S)} x_e = 2 \sum_{v_i \in S} y_i,$$

$$\sum_{e \in \delta(S)} x_e \leq 2 \sum_{v_i \in S} y_i.$$

Then, a simple strengthening of constrains (3.14) is given by

$$\sum_{e \in \delta(S)} x_e \geq 2 \quad \text{for all } S \subseteq M \text{ such that } \sum_{v_i \in M_k \setminus S} q_{ki} < d_k, \tag{3.15}$$

for all $p_k \in K$.

Finally, some valid constrains particular to the STPP can be derived. For example, the constraint

$$\sum_{e \in \delta(S)} x_e \geq \frac{2}{d_k} \sum_{v_i \in S \cap M_k} z_{ki} \quad \text{for all } S \subseteq M \text{ and } p_k \in K \tag{3.16}$$

states that at least two edges must be incident to S whenever some amount of any product p_k is purchased in a market of $S \cap M_k$. Constrains (3.16) coincide with inequalities (3.5) when $S = \{v_i\}$. Clearly, if $\sum_{v_i \in M_k \setminus S} q_{ki} < d_k$ for some product p_k, then constrains (3.16) are dominated by constrains (3.15). Again, constrains (3.16) can be strengthened by

$$\sum_{e \in \delta(S)} x_e \geq \frac{2 \sum_{v_i \in S \cap M_k} z_{ki}}{\min\{d_k, \sum_{v_i \in S \cap M_k} q_{ki}\}} \quad \text{for all } S \subseteq M \text{ and } p_k \in K,$$

where $\min\{d_k, \sum_{v_i \in S \cap M_k} q_{ki}\}$ is a stronger upper bound on the quantity of product p_k that can be purchased in S.

Additional inequalities involving more than one product and based on subset of markets can also be derived. The advantage of constraints (3.16) is that they allow the generation of violated inequalities in polinomial time (see Section 5.4 in Chapter 5), which has proved useful in our computational experiments (see Section 5.7 in Chapter 5).

3.3 ILP FORMULATION FOR THE ATPP

Now we present an Integer Programming model for the asymmetric version of the TPP (ATPP). We also use three types of decision variables that are identical to the undirected case but the variables related to the arcs. In this case, since both extremities of the arc play different roles, it is necessary taking into account a different variable for each case:

$$x_a := \begin{cases} 1 & \text{if arc } a \text{ belongs to the solution} \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } a \in A;$$

$$y_i := \begin{cases} 1 & \text{if vertex } v_i \text{ belongs to the solution} \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } v_i \in V;$$

$$z_{ki} := \text{the amount of product } p_k \text{ is purchased at market } v_i, \text{ for all } p_k \in K \text{ and all } v_i \in M_k.$$

The ATPP formulation is the following

$$w^{OPT} := \min \sum_{a \in A} c_a x_a + \sum_{p_k \in K} \sum_{v_i \in M_k} b_{ki} z_{ki} \tag{3.17}$$

subject to

$$\sum_{a \in \delta^+(\{v_i\})} x_a = y_i \qquad \text{for all } v_i \in V \tag{3.18}$$

$$\sum_{a \in \delta^-(\{v_i\})} x_a = y_i \qquad \text{for all } v_i \in V \tag{3.19}$$

$$\sum_{a \in \delta^+(S)} x_a \geq y_i \qquad \text{for all } S \subseteq M \text{ and all } v_i \in S \tag{3.20}$$

$$\sum_{v_i \in M_k} z_{ki} = d_k \qquad \text{for all } p_k \in K \tag{3.21}$$

$$z_{ki} \leq q_{ki} y_i \qquad \text{for all } p_k \in K \text{ and all } v_i \in M_k \tag{3.22}$$

$$x_a \in \{0, 1\} \qquad \text{for all } a \in A \tag{3.23}$$

$$y_i \in \{0, 1\} \qquad \text{for all } v_i \in M \setminus M^* \tag{3.24}$$

$$y_i = 1 \qquad \text{for all } v_i \in M^* \tag{3.25}$$

$$z_{ki} \geq 0 \qquad \text{for all } p_k \in K \text{ and all } v_i \in M_k. \tag{3.26}$$

The objective function is defined in the same sense that was in its symmetric counterpart.

Constrains (3.18) and (3.19) impose the in-degree and out-degree of each visited vertex be equal to one, respectively. Constrains (3.20), named *YSEC*$^+$ impose strong connectivity. Because of (3.18) and (3.19), inequality (3.20) can be equivalently re-written as

$$\sum_{a \in A(S)} x_a \leq \sum_{v_j \in S \setminus \{v_i\}} y_j \qquad \text{for all } S \subseteq M \text{ and all } v_i \in S.$$

These constraints ensure that a visited market in a subset $S \subseteq M$ must be connected to the depot through a path. By also using equations (3.18) and (3.19), these constrains are also equivalent to:

$$\sum_{a \in \delta^-(S)} x_a \geq y_i \qquad \text{for all } S \subset M \text{ and all } v_i \in S.$$

As in the directed case, inequalities (3.21) guarantee that the exact amount of product p_k is purchased. Inequalities (3.22) mean that it is not possible to purchase a product p_k in a market v_i if it is not visited, and that it is not possible to purchase more than its offer q_{ki} if it is visited. Constraints (3.23)–(3.26) impose bounds and integrality conditions on the variables.

3.4 VALID INEQUALITIES FOR THE ATPP

When using model (3.17)–(3.26) in a cutting-plane approach for the exact solution of the ATPP, one observe that the basic linear relaxation need additional valid inequalities to close the integrality gap. To this end this section shows some considerations leading to a tighter linear relaxation.

A first observation is based on the fact that constraints (3.20) can be trivially strengthened if there is a product p_k that cannot be totally purchased outside markets in S. More precisely, if $S \subseteq M$ such that $\sum_{v_i \in M_k \setminus S} q_{ki} < d_k$ for a product $p_k \in K$, then the inequality:

$$\sum_{a \in \delta^+(S)} x_a \geq 1 \tag{3.27}$$

is valid for all ATTP solutions. This inequality imposes the requirement that the purchaser must visit a market in S.

As in the undirected version some problems underlying in this model. A first subproblem of this family is the one defined by the assignment problem, i.e., all spanning unions of directed cycles. Another subproblem is the *Directed Cycle Problem* with the y variables, which is obtained from assignment problem restricting the set of assignments to those having exactly one cycle of length greater than one, i.e.,

$$\begin{aligned}
\sum_{a \in \delta^+(\{v_i\})} x_a &= y_i \qquad \text{for all } v_i \in V \\
\sum_{a \in \delta^-(\{v_i\})} x_a &= y_i \qquad \text{for all } v_i \in V \\
\sum_{a \in \delta^+(S)} x_a &\geq y_i \qquad \text{for all } S \subseteq M \text{ and all } v_i \in S \\
x_a &\in \{0,1\} \qquad \text{for all } a \in A \\
y_i &\in \{0,1\} \qquad \text{for all } v_i \in M.
\end{aligned}$$

It is referred in Balas [8, 9, 10] and in Balas and Oostend [13] as the P_0 polytope. Thus valid inequalities from P_0 are also valid for the ATPP. For example,

$$\sum_{(i,j) \in A(S)} x_{ij} \leq \sum_{v_h \in S \setminus \{v_l\}} y_h + (1 - y_p),$$

for all $S \subset V$, $2 \leq |S| \leq |V| - 1$, and $v_l \in S$, $v_p \notin S$, which Balas [8, 10] proved that were valid for P_0 and facet defining when $S \leq |V| - 2$. As consequence, the inequalities

$$\sum_{(i,j) \in \delta^+(S)} x_{ij} \geq y_l + y_p - 1,$$

for all $S \subset V$, $2 \leq |S| \leq |V| - 1$, and $v_l \in S$, $v_p \notin S$, are also valid for P_0.

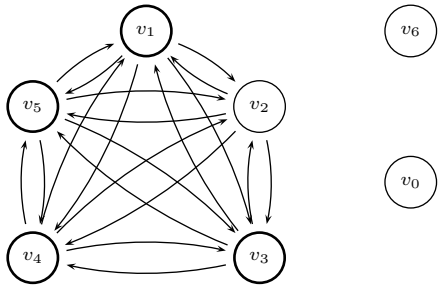


Fig. 3.2 Inequalities (3.28) for the ATPP.

Proposition 3.1. *Inequalities*

$$\sum_{(i,j)\in A(S)} x_{ij} \leq \sum_{v_h \in S\setminus\{v_l\}} y_h, \tag{3.28}$$

are valid for the ATPP if $\sum_{v_i \in M_k \setminus S} q_{ki} \geq d_k$ *for all products* $p_k \in K$ *and for* $S \subseteq M$.

Proof. To prove the validity of (3.28) let us consider that the point (x^*, y^*, z^*) violates (3.28), then the cycle in G has its nodes in S, but, since $S \subseteq M$ and $v_0 \notin M$, and then (x^*, y^*, z^*) would violate (3.25) and it would not belong to the ATPP. □

Proposition 3.2. *Inequalities*

$$\sum_{(i,j)\in A(S)} x_{ij} \leq \sum_{v_h \in S} y_h - 1, \tag{3.29}$$

for $S \subset M$ *are valid for the ATPP if*

$$\sum_{v_i \in M_k \setminus S} q_{ki} < d_k \tag{3.30}$$

for a product $p_k \in K$.

Proof. If (x^*, y^*, z^*) violates (3.29) then the cycle in G has all its nodes either in S, or in $V \setminus S$. It is clear that at least one vertex in $V \setminus S$ has to be visited since $v_0 \notin S$. In addition, if (3.30) holds then at least one vertex in S has to be visited in order to satisfy (3.21), then (x^*, y^*, z^*) does not belong to the ATPP. □

On the other hand, the lifted cycle D_k^+ inequalities

$$x_{i_l i_1} + \sum_{h=1}^{l-1} x_{i_h i_{h+1}} + 2\sum_{h=3}^{l} x_{i_1 i_h} + \sum_{j=4}^{l}\sum_{h=3}^{j-1} x_{i_j i_h} \leq y_{i_1} + \sum_{h=3}^{l} y_{i_h} + (1 - y_{i_p}),$$

for any $l \in \{3, \ldots, n-2\}$ and any $p \in \{l+1, \ldots, n\}$, which are facet for the P_0 polytope (see Balas [9, 10] for details), are also valid for the ATPP.

These inequalities come from

$$x_{i_l i_1} + \sum_{h=1}^{l-1} x_{i_h i_{h+1}} + 2\sum_{h=3}^{l} x_{i_1 i_h} + \sum_{j=4}^{l}\sum_{h=3}^{j-1} x_{i_j i_h} \leq l - 1,$$

which have been proposed by Grötschel and Padberg [81] for the Asymmetric TSP, Fischeti [51] proved that they are facets for the asymmetric TSP, and a separation algorithm was proposed by Fischetti and Toth [58].

Proposition 3.3. *For any $k \in \{3, \ldots, n-2\}$ and any $l \in \{k+1, \ldots, n\}$, the lifted cycle inequalities*

$$x_{i_k i_1} + \sum_{h=1}^{k-1} x_{i_h i_{h+1}} + 2\sum_{h=3}^{k} x_{i_1 i_h} + \sum_{j=4}^{k}\sum_{h=3}^{j-1} x_{i_j i_h} \leq y_{i_1} + \sum_{h=3}^{k} y_{i_h} + (1 - y_{i_l}), \quad (3.31)$$

are valid for the ATPP.

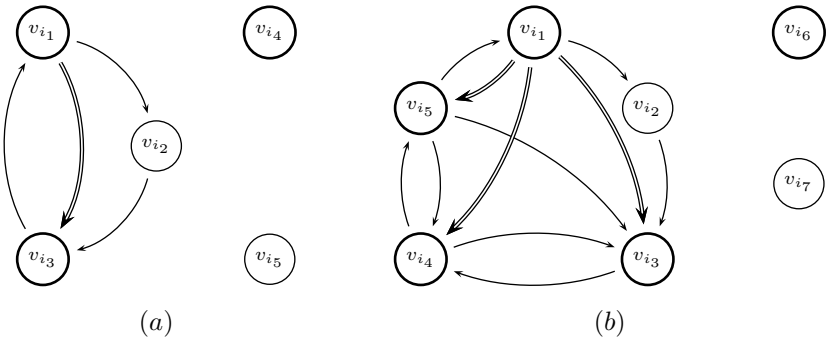


Fig. 3.3 Primitive lifted cycle inequalities for the ATPP for 5 and 7 vertices.

Proof. Let us suppose that the claim is false. Therefore the point (x^*, y^*, z^*) for the ATPP violates (3.31). Let us consider the following two cases.

1. The node v_{i_l} does not belong to the cycle, so the cycle C associated is a subset of $A(\{i_1, \ldots, i_k\})$. Then $y_{i_l} = 0$, and since (3.31) is violated the sum of the remaining components of (x^*, y^*, z^*) associated to the arcs is at least the sum of the components associated to the vertices plus one. But it is not possible because a simple cycle contains at most one arc with coefficient 2 and the remaining arcs with coefficient 1.

2. The node v_{i_l} belongs to the cycle. Then $y^*_{i_l} = 1$ and the sum of the component of (x^*, y^*, z^*) corresponding to arcs is at least the sum of the component

corresponding to vertices but v_{i_2}, if (3.31) is violated. For this to be true, if p is the number of nodes belonging to the cycle of the set $\{v_{i_1}, \ldots, v_{i_k}\}$, the cycle would have to contain $k - p$ arcs with coefficients 1 of the subgraph induced by the $k - p$ remaining nodes of the set $\{v_{i_1}, \ldots, v_{i_k}\}$, or $k - p - 2$ arcs with coefficient 1 and one arc with coefficient 2 (notice that if node v_{i_2} belongs to the cycle no coefficient 2 arcs are allowed), of the same subgraph. Both possibilities are excluded since the cycle contains a node v_{i_l}.

□

The lifted inequalities (3.31) can be strengthened as follows.

Proposition 3.4. *Inequalities*

$$x_{i_l i_1} + \sum_{h=1}^{l-1} x_{i_h i_{h+1}} + 2\sum_{h=3}^{l} x_{i_1 i_h} + \sum_{j=4}^{l}\sum_{h=3}^{j-1} x_{i_j i_h} \leq \sum_{h=1}^{l} y_{i_h} - y_2, \qquad (3.32)$$

are valid for the ATPP, for any $l \in \{3, \ldots, |M| - 2\}$ *if* $\sum_{v_i \in M_k \setminus S} q_{ki} \geq d_k$ *for all products* p_k.

Proof. Let us suppose that (x^*, y^*, z^*) belonging to the ATPP violates (3.32), then the cycle is entirely contained in the subgraph induced by the node set $\{v_{i_1}, \ldots, v_{i_k}\}$, and $y_i = 0$ for all $v_i \in \{v_{k+1}, \ldots, v_n\}$. But there must be a product p_k such that its demand is not satisfied, and this contradicts the assumption that (x^*, y^*, z^*) is in the ATPP. □

Proposition 3.5. *Inequalities*

$$x_{i_l i_1} + \sum_{h=1}^{l-1} x_{i_h i_{h+1}} + 2\sum_{h=3}^{l} x_{i_1 i_h} + \sum_{j=4}^{l}\sum_{h=3}^{j-1} x_{i_j i_h} \leq \sum_{h=1}^{l} y_{i_h} - 1, \qquad (3.33)$$

for $l := \{3, \ldots, |M|-2\}$, *are valid for the ATPP if* $\sum_{v_i \in M_k \setminus S} q_{ki} < d_k$ *for a product* $p_k \in K$ *for* $S = \{v_{i_1}, \ldots, v_{i_l}\} \subset M$.

As in the symmetric case, a second subproblem defined by

$$\begin{aligned}
\sum_{v_i \in M_k} z_{ki} &= d_k && \text{for all } p_k \in K \\
z_{ki} &\leq q_{ki} y_i && \text{for all } p_k \in K \text{ and all } v_i \in M_k \\
y_i &\in \{0, 1\} && \text{for all } v_i \in M \\
z_{ki} &\geq 0 && \text{for all } p_k \in K \text{ and all } v_i \in M_k,
\end{aligned}$$

corresponds to a generalization of the UFLP with upper bounds on the customer-facility variables. Valid inequalities for this subproblem when $q_{ki} = d_k$ can be obtained from the *Set Covering Problem* polytope (see e.g., Balas and Ng [12]).

Decomposing this second subproblem for each product yields the family of cover inequalities:

$$\sum_{v_i \in S} y_i \geq 1 \quad \text{for all } S \subseteq M_k \text{ such that } \sum_{v_i \in M_k \setminus S} q_{ki} < d_k, \tag{3.34}$$

for all $p_k \in K$. These constraints state that a market in S must be visited if markets in $M_k \setminus S$ are not enough to provide the required demand d_k of product p_k.

If we sum up (3.18) for all $v_i \in S$, we obtain

$$\sum_{a \in \delta(S)^+} x_a + \sum_{a \in A(S)} x_a = \sum_{v_i \in S} y_i,$$

$$\sum_{a \in \delta(S)^+} x_a \leq \sum_{v_i \in S} y_i.$$

Therefore, a simple strengthening of constraints (3.34) is given by

$$\sum_{(i,j) \in \delta^+(S)} x_{ij} \geq 1 \tag{3.35}$$

for all $S \subseteq M$ such that $\sum_{v_i \in M_k \setminus S} q_{ki} < d_k$, for all $p_k \in K$.

Some valid constraints particular to the ATPP can be derived. For example, the constraint

$$\sum_{(i,j) \in \delta^+(S)} x_{ij} \geq \frac{1}{d_k} \sum_{v_i \in S \cap M_k} z_{ki} \quad \text{for all } S \subseteq M \text{ and } p_k \in K \tag{3.36}$$

states that at least two edges must be incident to S whenever some amount of any product p_k is purchased in a market of $S \cap M_k$. Constraints (3.36) coincide with inequalities (3.22) when $S = \{v_i\}$. Clearly, if $\sum_{v_i \in M_k \setminus S} q_{ki} < d_k$ for some product p_k, then constraints (3.36) are dominated by constraints (3.35). Again, constraints (3.36) can be strengthened by

$$\sum_{(i,j) \in \delta^+(S)} x_{ij} \geq \frac{\sum_{v_i \in S \cap M_k} z_{ki}}{\min\{d_k, \sum_{v_i \in S \cap M_k} q_{ki}\}} \quad \text{for all } S \subseteq M \text{ and } p_k \in K,$$

where $\min\{d_k, \sum_{v_i \in S \cap M_k} q_{ki}\}$ is a stronger upper bound on the quantity of product p_k that can be purchased in S.

4

Polyhedral Analysis

This chapter focuses on the polyhedral aspects of the Traveling Purchaser Problem from a theoretical point of view. It is assumed that the reader has a deep knowledge on Linear Algebra and Polyhedral theory. Nevertheless, section 1.3 provides an introduction to the basic concepts on those topics. The first two sections of this chapter are devoted to the dimension of the TPP polytope, in both the symmetric and asymmetric versions. A general procedure to obtain facet defining inequalities is described in section 4.3. Finally, facets for the symmetric and asymmetric TPP (STPP and ATPP) are described in sections 4.4 and 4.5 respectively.

4.1 DIMENSION OF THE STPP POLYTOPE

Let $\mathcal{X}$ be the set of all vectors (x, y, z) satisfying constraints

$$\sum_{e \in \delta(\{v_i\})} x_e = 2y_i \qquad \text{for all } v_i \in V$$

$$\sum_{e \in \delta(S)} x_e \geq 2y_i \qquad \text{for all } S \subseteq M \text{ and all } v_i \in S$$

$$\sum_{v_i \in M_k} z_{ki} = d_k \qquad \text{for all } p_k \in K$$

$$z_{ki} \leq q_{ki} y_i \qquad \text{for all } p_k \in K \text{ and all } v_i \in M_k$$

$$x_e \in \{0, 1\} \qquad \text{for all } e \in E$$

$$y_i \in \{0, 1\} \qquad \text{for all } v_i \in M \setminus M^*$$

$$\begin{aligned} y_i &= 1 \qquad \text{for all } v_i \in M^* \\ z_{ki} &\geq 0 \qquad \text{for all } p_k \in K \text{ and all } v_i \in M_k, \end{aligned}$$

i.e., the set of all feasible STPP solutions. Let $Q^{TPP} := \text{conv}\{\mathcal{X}\}$ be the STPP polytope. Let also $Q := \{(x, y, z) \in Q^{TPP} : y_i = 1 \text{ for all } v_i \in V \setminus M^*\}$, i.e. the STPP in which all the vertices have to be visited. Let Q_x be the projection of Q onto the affine space of the x variables, and Q_z the projection of Q onto the affine space of the z variables. Then Q_x is the symmetric TSP polytope on $G = (V, E)$ and Q_z is the polytope of a generalization of the *Assignment Problem* defined by

$$\begin{aligned} \sum_{v_i \in M_k} z_{ki} &= d_k \qquad \text{for all } p_k \in K \\ 0 \leq z_{ik} &\leq q_{ik}. \end{aligned}$$

The following well known graph theoretical lemma is going to be useful proving the dimension of Q_x.

Lemma 4.1. *(Grötschel and Padberg [81]) Let $K_n = (V, E)$ be the complete graph on n vertices, and let k denote any integer.*

(i) if $|V| = 2k+1$, then there exist k edge-disjoint tours T_i such that $E = \bigcup_{i=1}^{K} T_i$.

(ii) if $|V| = 2k$, then there exist k edge-disjoint tours T_i and a perfect 1-matching M edge-disjoint from any T_i such that $E = M \cup \bigcup_{i=1}^{k-1} T_i$.

Theorem 4.1. *(Naddef [112]) The dimension of Q_x is $|E| - |V|$.*

Proof. (sketch) For $n = 3$ there is only one tour, so the theorem is true in that case. The equations

$$\sum_{e \in \delta(\{v_i\})} x_e = 2y_i \qquad \text{for all } v_i \in V$$

are linearly independent (See Figure 4.1 for an example of these equations in the undirected graph K_4) so the dimension can not be more than what is announced in the theorem. It is therefore enough to exhibit $|E|-|V|+1$ affinely independent tours. For this we use Lemma 4.1. If $n = 2k$, then the edges of K_{n-1} can be partitioned into $k - 1$ edge disjoint Hamiltonian cycles. If $n = 2k + 1$, then the edges of K_{n-1} can be partitioned into $k - 1$ edges disjoint Hamiltonian cycles and a perfect matching. In the first case, for each Hamiltonian cycle of K_{n-1} and each edge on that cycle, we create a one Hamiltonian cycle of K_n by inserting node n between the endnodes of that edge, i.e. if $e = (i, j)$, we remove edge e and add the two edges (i, n) and (j, n). In the second case, for the Hamiltonian cycle we do the same thing. We also complete the perfect matching to a Hamiltonian cycle arbitrarily, and do as previously only with the edges of the perfect matching. Now it only remains to check that in both cases we have the right number of cycles and that the corresponding vectors are affinely independent. □

Theorem 4.2. *For the particular case in which both d_k and q_{ki} are equal to* 1*, for each $p_k \in K$ and for each market $v_i \in M$, the dimension of Q_z is $\sum_{p_k \in K \setminus K^*}(|M_k| - 1)$.*

Proof. For each product $p_k \in K$ the number of variables is $|M_k|$, and there is just one equation, so the dimension of Q_z is no more than $|M_k| - 1$ for each product $p_k \in K$. It is hence enough to exhibit $\sum_{p_k \in K \setminus K^*}(|M_k| - 1) + 1$ affinely independent points $\{z_1, \ldots, z_{\sum_{p_k \in K \setminus K^*}(|M_k|-1)+1}\}$ belonging to Q_z. This set of points is described in Table 4.1, and is constructed as follows.

For each product $p_k \in K$ there is a variable z_{ki} for each market $v_i \in M_k$. Let us construct the sequence $v_{i_{k1}}, \ldots, v_{i_{k|M_k|}}$ according to an arbitrary order of the set M_k for each $p_k \in K$. For each product $p_k \in K$ each point in Q_z has exactly a variable of the set $z_{ki_{kj}}$ for $j := 1 \ldots, |M_k|$ with value 1. The first group of $|M_1|$ affinely independent points is constructed fixing to 1 the variables $z_{ki_{kj}}$ for $j := 1, \ldots, |M_k|$ and $k := 2, \ldots, |K|$, and varying $z_{1i_{1j}} := 1$ for $j := 1, \ldots, |M_1|$.

The remaining $|K| - 1$ groups of points of size $|M_k| - 1$ for $k := 2, \ldots, |K|$ are created by fixing $z_{hi_{h1}}$ to 1 for $h \neq k$ and $k := 2, \ldots, |K|$, and varying $z_{ki_{kj}} := 1$ for $j := 1, \ldots, |M_k|$.

It is clear that the points $\{z_2 - z_1, \ldots, z_{\sum_{p_k \in K \setminus K^*}(|M_k|-1)+1} - z_1\}$ are linearly independent, since the matrix described in Table 4.1 is non-singular, and they are in Q_z. □

Since $Q := Q_x \times \{y : y_i = 1 \text{ for all } i \in V\} \times Q_z$, the following result follows.

Lemma 4.2. $\dim(Q) = |E| - |V| + \sum_{p_k \in K \setminus K^*}(|M_k| - 1)$*, and the facets of Q are the facets of the TSP polytope Q_x and the facets of Q_z.*

Since the polyhedral structure of Q is widely known (see, e.g., Naddef [112]), our first aim is to extend results from Q onto Q^{TPP}. To this end we introduce the following intermediate polytopes:

$$Q(F) := \{(x, y, z) \in Q : y_i = 1 \text{ for all } v_i \in V \setminus (M^* \cup F)\},$$

for all $F \subseteq V \setminus M^*$. Moreover, the integer STPP solutions in $Q(F)$ also must contain vertices in $V \setminus (M^* \cup F)$. Hence, $Q(\emptyset) = Q$ and $Q(V \setminus M^*) = Q^{TPP}$.

We first compute the dimension of $Q(F)$ for any given F.

Theorem 4.3. *For all $F \subseteq V \setminus M^*$,* $\dim(Q(F)) = |E| - |V| + \sum_{p_k \in K \setminus K^*}(|M_k| - 1) + |F|$.

Proof. In the space of all of the $|E| + |V| + \sum_{p_k \in K} |M_k|$ variables (x, y, z), a vector in $Q(F)$ satisfies the $|V|$ equations

$$\sum_{e \in \delta(\{v_i\})} x_e = 2y_i \qquad \text{for all } v_i \in V,$$

the $|K|$ equations

$$\sum_{v_i \in M_k} z_{ki} = d_k \qquad \text{for all } p_k \in K,$$

Table 4.1 Set of $\sum_{p_k \in K}(|M_k|-1)+1$ affinely independent points in Q_z.

	$z_{1i_{11}}$	$z_{1i_{12}}$	$\cdots$	$z_{1i_{1\|M_1\|}}$	$z_{2i_{21}}$	$z_{2i_{22}}$	$\cdots$	$z_{2i_{2\|M_2\|}}$	$\cdots$	$z_{ki_{k1}}$	$z_{ki_{k2}}$	$\cdots$	$z_{ki_{k\|M_k\|}}$
1	1	0		0	1	0		0		1	0		0
	0	1	$\cdots$	0	1	0	$\cdots$	0		1	0	$\cdots$	0
$\|M_1\|-1$	$\vdots$		$\ddots$		$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$		$\vdots$
	0	0		1	1	0		0		1	0		0
	1	0	$\cdots$	0	0	1	$\cdots$	0		1	0	$\cdots$	0
$\|M_2\|-1$	$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\ddots$			$\vdots$	$\vdots$		$\vdots$
	1	0		0	0	0		1		1	0		0
	1	0	$\cdots$	0	1	0	$\cdots$	0		1	0	$\cdots$	0
$\sum_{k=3}^{\|K\|-1}(\|M_k\|-1)$	$\vdots$												
	1	0		0	1	0		0		1	0		0
	1	0	$\cdots$	0	1	0	$\cdots$	0		0	1	$\cdots$	0
$\|M_{\|K\|}\|-1$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\ddots$	
	1	0	$\cdots$	0	1	0	$\cdots$	0		0	0	$\cdots$	1

the $|V| - |F|$ equations

$$y_i = 1 \quad \text{for all } v_i \in V \setminus F,$$

and the $\sum_{p_k \in K^*}(|M_k| - 1)$ equations

$$z_{ki} = q_{ki} \quad \text{for all } p_k \in K^* \text{ and } v_i \in M_k.$$

Since all the equations are linearly independent,

$$\dim(Q(F)) \leq (|E|+|V|+\sum_{p_k \in K} |M_k|)-(|V|+|K|+|V|-|F|+\sum_{p_k \in K^*}(|M_k|-1)).$$

The other direction of the inequality (and hence the thesis) follows by induction on $|F|$. Indeed, for $|F| = 0$ then $F = \emptyset$ and $Q(F) = Q$, and we are done. Suppose now that the thesis is true for a set $F \subset V \setminus M^*$, and let us prove it for $F' = F \cup \{v_i\}$ with $v_i \notin F \cup M^*$. By the induction hypothesis there are $|E| - |V| + \sum_{p_k \in K \setminus K^*}(|M_k| - 1) + |F| + 1$ affinely independent STPP solutions with $y_i = 1$; because $v_i \notin M^*$ there exists also a STPP solution with $y_i = 0$ (e.g., a Hamiltonian cycle in the subgraph induced by $V \setminus \{v_i\}$). Therefore, there exist at least $|E| - |V| + \sum_{p_k \in K \setminus K^*}(|M_k| - 1) + |F'| + 1$ affinely independent STPP solutions. □

A first trivial consequence of Theorem 4.3 is the dimension of the STPP polytope.

Corollary 4.1. *The dimension of Q^{TPP} is $|E| - |M^*| + \sum_{p_k \in K \setminus K^*}(|M_k| - 1)$.*

4.2 DIMENSION OF THE ATPP POLYTOPE

Let $\mathcal{Y}$ be the set of all vectors (x, y, z) satisfying constraints

$$\sum_{a \in \delta^+(\{v_i\})} x_a = y_i \quad \text{for all } v_i \in V \tag{4.1}$$

$$\sum_{a \in \delta^-(\{v_i\})} x_a = y_i \quad \text{for all } v_i \in V \tag{4.2}$$

$$\sum_{a \in \delta^+(S)} x_a \geq y_i \quad \text{for all } S \subseteq M \text{ and all } v_i \in S \tag{4.3}$$

$$\sum_{v_i \in M_k} z_{ki} = d_k \quad \text{for all } p_k \in K \tag{4.4}$$

$$z_{ki} \leq q_{ki} y_i \quad \text{for all } p_k \in K \text{ and all } v_i \in M_k \tag{4.5}$$

$$x_a \in \{0, 1\} \quad \text{for all } a \in A \tag{4.6}$$

$$y_i \in \{0, 1\} \quad \text{for all } v_i \in M \setminus M^* \tag{4.7}$$

$$y_i = 1 \quad \text{for all } v_i \in M^* \tag{4.8}$$

$$z_{ki} \geq 0 \quad \text{for all } p_k \in K \text{ and all } v_i \in M_k. \tag{4.9}$$

Let $P^{TPP} := \text{conv}\{\mathcal{Y}\}$ be the ATPP polytope. Let also $P := \{(x, y, z) \in P^{TPP} : y_i = 1 \text{ for all } v_i \in V \setminus M^*\}$, P_x be the projection of P onto the affine space of the x variables, let

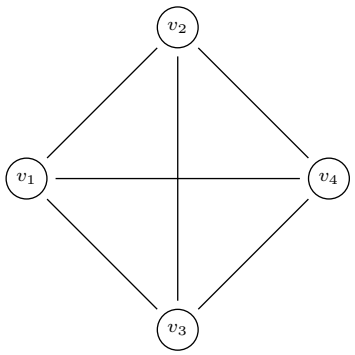
$$AP := \text{conv}\{(x, y) \in \mathbb{R}^A \times \mathbb{R}^V : (x, y) \text{ satisfies } (4.1), (4.2) \text{ and } (4.6)\text{–}(4.8)\}$$

be the convex hull of incidence vectors of all spanning unions of directed cycles, and let

$$P_0 := \text{conv}\{(x, y) \in \mathbb{R}^A \times \mathbb{R}^V : (x, y) \in AP \text{ and satisfies } (4.3)\}$$

be the restriction of the set of assignments to those having exactly one cycle of length greater than one.

Let us also define P_z as the projection of P onto the affine space of the z variables. Then P_x is the asymmetric TSP on $G = (V, A)$ and $P_z = Q_z$ is the polytope of a generalization of the *Assignment Problem* defined as the in the symmetric case.



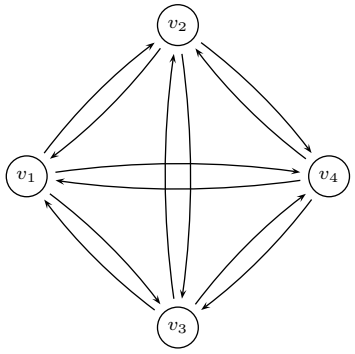
x_{12}	x_{13}	x_{14}	x_{23}	x_{24}	x_{34}
1	1	1			
1			1	1	
	1		1		1
		1		1	1

Fig. 4.1 An illustration of the degree equations for the STSP.

Theorem 4.4. *(Grötschel and Padberg [81] and Fischetti [51]) The dimension of P_x is $|A| - 2|V| + 1$.*

Notice that the rank of

$$\sum_{a \in \delta^+(\{v_i\})} x_a = y_i \qquad \text{for all } v_i \in V$$



x_{12}	x_{13}	x_{14}	x_{21}	x_{23}	x_{24}	x_{31}	x_{32}	x_{34}	x_{41}	x_{42}	x_{43}
1	1	1									
			1	1	1						
						1	1	1			
									1	1	1
			1			1			1		
1							1			1	
	1			1							1
		1			1			1			

Fig. 4.2 An illustration of the assignment equations for the ATSP.

$$\sum_{a \in \delta^-(\{v_i\})} x_a = y_i \qquad \text{for all } v_i \in V$$

is $2|V| - 1$, so dim(P_x)$\leq |A| - 2|V| + 1$. Figure 4.2 shows an example of these equations for the directed graph D_4. It is easy realize each row is linearly dependent from the remaining rows.

A direct proof of this theorem has been given in Grötschel and Padberg [81]. However, it is not too difficult to give a proof of this theorem paralleling the proof of theorem 4.1. From the $|E| + |V| + 1$ undirected tour constructed in the proof of the theorem 4.1 we can obtain $|A| - 2|V| + 2$ directed tours by taking the two possible orientations of each undirected tour. In order to complete this approach we must show that the incidence vectors of these directed tours are linearly independent. Another interesting proof of this theorem is given in Fischetti [51].

Lemma 4.3. $\dim(P) = |A| - 2|V| + 1 + \sum_{p_k \in K \setminus K^*}(|M_k| - 1)$, *and the facets of* P *are the facets of the TSP polytope* P_x *and the facets of* P_z.

Since it is quite important the sequence in which the vertices are taken during the sequential lifting procedure described in the next section, the intermediate polytopes

are restated considering this sequence. Let $P(\{v_1, \ldots, v_k\})$ be the intermediate polytopes such as

$$P(\{v_1, \ldots, v_k\}) := \{(x, y, z) \in P : y_i = 1 \text{ for all } v_i \in V \setminus (M^* \cup \{v_1, \ldots, v_k\})\}$$

for a given sequence $\{v_1, \ldots, v_k\}$ of vertices belonging to $V \setminus M^*$.

Theorem 4.5. *For all* $F \subseteq V \setminus M^*$, $\dim(P(F)) = |A| - |V| + 1 + \sum_{p_k \in K \setminus K^*}(|M_k| - 1) + |F|$.

Proof. In the space of all of the $|A| + |V| + \sum_{p_k \in K} |M_k|$ variables (x, y, z), a vector in $P(F)$ satisfies the $2|V|$ equations in

$$\sum_{a \in \delta^+(\{v_i\})} x_a = y_i \qquad \text{for all } v_i \in V$$

$$\sum_{a \in \delta^-(\{v_i\})} x_a = y_i \qquad \text{for all } v_i \in V$$

with rank $2|V| - 1$, the $|K|$ equations

$$\sum_{v_i \in M_k} z_{ki} = d_k \qquad \text{for all } p_k \in K,$$

the $|V| - |F|$ equations

$$y_i = 1 \qquad \text{for all } v_i \in V \setminus F,$$

and the $\sum_{p_k \in K^*}(|M_k| - 1)$ equations

$$z_{ki} = q_{ki} \qquad \text{for all } p_k \in K^* \text{ and } v_i \in M_k.$$

Therefore,

$$\dim(P(F)) \leq (|A| + |V| + \sum_{p_k \in K} |M_k|) - (2|V| - 1 + |K| + |V| - |F| + \sum_{p_k \in K^*} (|M_k| - 1)).$$

The other direction of the inequality (and hence the thesis) follows by induction on $|F|$. Indeed, for $|F| = 0$ then $F = \emptyset$ and $P(F) = P$, and we are done. Suppose now that the thesis is true for a set $F \subset V \setminus M^*$, and let us prove it for $F' = F \cup \{v_i\}$ with $v_i \notin F \cup M^*$. By the induction hypothesis there are $|A| - 2|V| + 1 + \sum_{p_k \in K \setminus K^*}(|M_k| - 1) + |F| + 1$ affinely independent TPP solutions with $y_i = 1$; because $v_i \notin M^*$ there exists also a TPP solution with $y_i = 0$ (e.g., a Hamiltonian cycle in the subgraph induced by $V \setminus \{v_i\}$). Therefore, there exist at least $|A| - 2|V| + 1 + \sum_{p_k \in K \setminus K^*}(|M_k| - 1) + |F'| + 1$ affinely independent TPP solutions. □

Corollary 4.2. *The dimension of* P_0 *is* $|A| - |V| + 1 - |M^*|$.

Corollary 4.3. *The dimension of P^{TPP} is* $|A|-|V|+1-|M^*|+\sum_{p_k\in K\setminus K^*}(|M_k|-1)$.

A second immediate consequence is that $Q(F)$ and $P(F)$ with $|F| = |V| - 1$ are facet-defining for P, as stated in the following result.

Corollary 4.4. *The inequality $y_i \leq 1$ defines a facet of Q^{TPP} and P^{TPP} if and only if $v_i \in V \setminus M^*$.*

A more important consequence of Theorem 4.3 is that, given a subset of markets $F \subset V \setminus M^*$, adding a new market $v_i \in V \setminus (F \cup M^*)$ to F increases the dimension of $Q(F)$ and $P(F)$ by exactly one unit.

4.3 THE LIFTING THEOREM

We use standard sequential lifting (see section 1.3 for details on this procedure) to calculate the coefficients of the y variables. Since in the previous basic theorem on sequential lifting is stated for full-dimensional polyhedra, and neither Q^{TPP} nor P^{TPP} are full-dimensional, we restate it here in the appropriated form. The following lemma based on the well-known sequential lifting theorem described, e.g., in Balas [10].

Suppose that we *introduce* the variables y into Q (P), one by one in some arbitrary sequence $v_1, \ldots, v_n$. Notice that to *introduce* a y_j variable for a vertex v_j belonging to a sequence $v_1, \ldots, v_{j-1}, v_j, \ldots, v_k$ into Q (P), is equivalent to obtain the polytope $Q(\{v_1, \ldots, v_{j-1}\} \cup v_j)$ ($P(\{v_1, \ldots, v_{j-1}\} \cup v_j)$).

Lemma 4.4. *Let $v_1, \ldots, v_k$ a sequence of vertices belonging to $V \setminus M^*$. Let*

$$\sum_{e\in E} \alpha_e x_e + \sum_{p_k\in K} \sum_{v_i\in M_k} \gamma_{ki} z_{ki} \geq \eta \tag{4.10}$$

be a facet-defining inequality for Q (P). Then the lifted inequality

$$\sum_{e\in E} \alpha_e x_e + \sum_{j=1}^{k} \beta_j (1 - y_j) + \sum_{p_k\in K} \sum_{v_i\in M_k} \gamma_{ki} z_{ki} \geq \eta \tag{4.11}$$

is valid and facet-defining for $Q(\{v_1, \ldots, v_k\})$ ($P(\{v_1, \ldots, v_k\})$), where

$$\beta_j := \eta - \min \Bigg\{ \sum_{e\in E} \alpha_e x_e + \sum_{i=1}^{j-1} \beta_i (1 - y_i) + \sum_{p_k\in K} \sum_{v_i\in M_k} \gamma_{ki} z_{ki} \ : \\ (x, y, z) \in Q(\{v_1, \ldots, v_{j-1}\}) \text{ and } y_j = 0 \Bigg\}. \tag{4.12}$$

Proof. By induction on k. For $k = 0$, $Q(\emptyset) = Q$, and (4.11) becomes (4.10), which is a valid facet defining for Q. Suppose that the hypothesis is true for $k =$

$0, 1, \ldots, q-1$, and let $k = q$. From the definition of the coefficients β_j, $j = 1, \ldots, q$, (4.11) with $k = q$ is valid for $Q(\{v_1, \ldots, v_q\})$. To see that it is facet defining, note that be the induction hypothesis (4.11) with $k = q - 1$ is a facet defining for $Q(\{v_1, \ldots, v_{q-1}\})$; hence, there exist dim($Q(\{v_1, \ldots, v_{q-1}\})$) affinely independent points $(x^i, y^i, z^i) \in Q(\{v_1, \ldots, v_{q-1}\})$ satisfying (4.11) at equality. To each such point $(x^i, y^i, z^i) \in Q(\{v_1, \ldots, v_{q-1}\}$, there corresponds a point in $Q(\{v_1, \ldots, v_q\})$ of the form $(x^i, y^i, 1, z^i)$, in which 1 is the value of the component y^i_j. It is easy to see that dim ($Q(\{v_1, \ldots, v_q\})$= dim ($Q(\{v_1, \ldots, v_{q-1}\})$+1) = dim +1), since $Q(\{v_1, \ldots, v_q\})$ has one more y variable than $Q(\{v_1, \ldots, v_{q-1}\})$ has, and the rank of the equalities system of $Q(\{v_1, \ldots, v_q\})$ is equal to $Q(\{v_1, \ldots, v_{q-1}\})$. So, we need one additional point in $Q(\{v_1, \ldots, v_q\})$. For this purpose, we use the vector $(x^*, y^*, 0, z^*) \in Q(\{v_1, \ldots, v_q\})$ that minimizes the expression defining (4.12). By construction, this point satisfies (4.11) at equality. Further, since $y^*_q = 0$, whereas $y^i_q = 1$ for all other points, $(x^*, y^*, 0, z^*)$ and the remaining dim ($Q(\{v_1, \ldots, v_{q-1}\})$) (dim) points $(x^*, y^*, 1, z^*)$ form a set of dim ($Q(\{v_1, \ldots, v_q\})$) (dim) affinely independent points in $Q(\{v_1, \ldots, v_q\})$. This completes the induction. □

As a consequence, any facet-defining inequality for $Q(F)$ ($P(F)$) can be lifted in a simple way so as to be facet-defining for $Q(F \cup \{v_i\})$ ($P(F \cup \{v_i\})$) as well. The idea is to choose a *lifting sequence* of the vertices in $V \setminus M^*$, say $\{v^1, \ldots, v^s\}$, and iteratively derive a facet of $Q(\{v^1, \ldots, v^{t-1}, v^t\})$ ($P(\{v^1, \ldots, v^{t-1}, v^t\})$) from a facet of $Q(\{v^1, \ldots, v^{t-1}\})$ ($P(\{v^1, \ldots, v^{t-1}\})$) for $t = 1, \ldots, s$.

We now use this technique for the trivial inequalities and the connectivity constraints.

4.4 FACETS OF THE STPP POLYTOPE

Theorem 4.6. *The inequality $x_e \geq 0$ defines a facet of Q^{TPP} for every $e \in E$ ($x_a \geq 0$ defines a facet of P^{TPP} for every $a \in A$).*

Proof. This is a direct consequence of Lemmas 4.2 and 4.4 (4.3 and 4.4), since $x_e \geq 0$ ($x_a \geq 0$) defines a facet of the symmetric (asymmetric) TSP polytope and every lifting sequence produces $\tilde{\beta}(v^t) = 0$ for all $v^t \in V \setminus M^*$. □

Theorem 4.7. *Let $S \subset M$ with $2 \leq |S| \leq |M| - 1$. Then the inequality*

$$\sum_{e \in \delta(S)} x_e \geq \begin{cases} 2 & \text{if there exists a product } p_k \text{ such that} \\ & \sum_{v_i \in M_k \setminus S} q_{ki} < d_k \text{ or } |S| = |M| - 1 \\ 2y_i & \text{for any } v_i \in S, \text{ otherwise,} \end{cases}$$

is facet-defining for Q^{TPP}.

Proof. If $F = \emptyset$, the inequality $\sum_{e \in \delta(S)} x_e \geq 2$ is known to be facet-defining for the TSP polytope, and therefore of Q (Lemma 4.2). To apply Lemma 4.4, let us consider any sequence of the markets in $V \setminus M^*$. Clearly, when there exists a product p_k such

that $\sum_{v_i \in M_k \setminus S} q_{ki} < d_k$ or $|S| = |M| - 1$, then $\tilde{\beta}(v^t) = 0$ for any sequence of the markets in $V \setminus M^*$, since is mandatory to visit at least a market in $|S|$. Otherwise, considering any $v_i \in S$ and a sequence $v^1, \ldots, v^s$ with $v^s = v_i$ yields $\tilde{\beta}(v^t) = 0$ for all $t = 1, \ldots, s-1$ and $\tilde{\beta}(v^s) = 2$. □

For a particular subset of markets S with $|S| = |M| - 1$, constraints

$$x_{[v_0, v_j]} \leq y_j \quad \text{for all } v_j \in V.$$

define facets of Q^{TPP}. Moreover, for the particular case $|S| = 2$ we obtain the following general result.

Corollary 4.5. *Let $e = [v_i, v_j] \in E$ be a given edge. The upper-bound inequality on x_e is the following:*

- *if $v_i \in M^*$ and $v_j \in M^*$, then $x_e \leq 1$;*
- *if $v_i \notin M^*$, $v_j \notin M^*$ and there exists $p_k \in K : \sum_{v_s \in M_k \setminus \{v_i, v_j\}} q_{ks} < d_k$; then $x_e \leq y_i + y_j - 1$;*
- *otherwise, $x_e \leq y_i$ when $v_i \notin M^*$ and $x_e \leq y_j$ when $v_j \notin M^*$.*

It defines a facet of Q^{TPP}.

In the same spirit, using Lemma 4.4, it is possible to derive others facet-defining inequalities for Q^{TPP} coming from facet-defining inequalities of the TSP polytope Q_x (like comb inequalities). It is also possible to lift facet-defining inequalities from Q_z as the following theorem shows.

Theorem 4.8. *If $d_k < \sum_{v_j \in M_k \setminus \{v_i\}} q_{kj}$ then the inequality $z_{ki} \geq 0$ defines a facet of Q^{TPP} if $p_k \in K \setminus K^*$ satisfies $|M_k| \geq 3$ and $v_i \in M_k$.*

Proof. If $F = \emptyset$, then $z_{ki} \geq 0$ is facet-defining for $Q(F)$ since it defines a facet of the assignment polytope Q_z when $d_k < \sum_{v_j \in M_k \setminus \{v_i\}} q_{kj}$. Indeed, this trivially follows by observing that Q_z is the Cartesian product of simplexes, each one in $|M_k| - 1$ dimensions when $p_k \in K \setminus K^*$. Moreover, for any sequence of $V \setminus M^*$ one computes $\tilde{\beta}(v^t) = 0$ in Lemma 4.4 for all $v^t \in M_k$ since there is a market in $M_k \setminus \{v_i, v^t\}$. □

When $d_k \geq \sum_{v_j \in M_k \setminus \{v_i\}} q_{kj}$ or when $M_k = \{v_i, v_j\}$, then $z_{ki} \geq 0$ is dominated by $z_{kj} \leq q_{kj} y_j$. We now analyze this upper-bound constraint on the z_{ki} variables.

Theorem 4.9. *Let $p_k \in K \setminus K^*$ and $v_i \in M_k$. If $v_i \notin M^*$ then the inequality $z_{ki} \leq q_{ki} y_i$ defines a facet of Q^{TPP}. Otherwise, $z_{ki} \leq q_{ki}$ defines a facet when $q_{ki} < d_k$.*

Proof. If $q_{ki} < d_k$, then $-z_{ki} \geq -q_{ki}$ defines a facet of $Q(\emptyset)$, so Lemma 4.4 can be applied to obtain a facet-defining inequality of $Q(V \setminus M^*)$. Indeed, if $v_i \in M^*$, then for any sequence of markets in $V \setminus M^*$ one computes $\tilde{\beta}(v^t) = 0$; otherwise, considering a sequence $v^1, \ldots, v^{t-1}, v^t$ with with $v^t = v_i$, one obtains $\tilde{\beta}(v^s) = 0$ for $s = 1, \ldots, t-1$ and $\tilde{\beta}(v_i) = -q_{ki}$.

When $q_{ki} = d_k$, Lemma 4.4 cannot be used since $z_{ki} \leq q_{ki}$ is dominated by (3.4). We are going to exhibit $\dim(Q^{TPP})$ affinely independent solutions which will prove the claim. Consider $p_k \in K \setminus K^*$ and $v_i \in M_k \setminus M^*$, and denote by

$$K_i := \{p_h \in K : v_i \in M_h\}$$

the set of products available at market v_i, and by

$$M_i^* := \left\{ v_j \notin M^* : \text{ there exists } p_h \text{ such that } \sum_{v_s \in M_h \setminus \{v_i, v_j\}} q_{hs} < d_h \right\},$$

the set of new markets that must be visited when v_i is not. Since $v_i \notin M^*$ we have $K_i \cap K^* = \emptyset$. Also because of Corollary 4.3, there are $\dim(Q^{TPP}) - (|\delta(\{v_i\})| + |K_i| + |M_i^*|) + 1$ affinely independent TPP solutions with $y_i = 0$, and therefore satisfying $z_{ki} = y_i$. We need another $|\delta(\{v_i\})|+|K_i|+|M_i^*|-1$ affinely independent TPP solutions with $y_i = 1$ and $z_{ki} = q_{ki}$. Indeed, $|\delta(\{v_i\})|$ can be constructed as follows: for an arbitrary but fixed $e \in \delta(\{v_i\})$, consider a Hamiltonian cycle in G using e and e' for each $e' \in \delta(\{v_i\}) \setminus \{e\}$, plus another Hamiltonian cycle not using e; in all such solutions, products can be purchased in such a way that $z_{ki} = q_{ki}$ and $z_{hi} = 0$ for all $p_h \in K_i \setminus \{p_k\}$. Moreover, for each $p_h \in K_i \setminus \{p_k\}$ consider a Hamiltonian cycle with $z_{ki} = q_{ki}$, $z_{hi} = q_{hi}$ and $z_{li} = 0$ for all $p_l \in K_i \setminus \{p_k, p_h\}$. Finally, for each $v_j \in M_i^*$ consider a Hamiltonian cycle on the subgraph induced by $V \setminus \{v_j\}$, with $z_{ki} = q_{ki}$. □

4.5 FACETS OF THE ATPP POLYTOPE

The following facet proofs are related to lifted inequalities from the cycle inequalities

$$x(C) \leq |C| - 1 \tag{4.13}$$

for $C \subseteq E$ the arc set of a directed cycle, $2 \leq |C| \leq |V| - 1$ for the TSP polytope P^{TSP}. In particular, to the Subtour Elimination Constraints

$$\sum_{(i,j) \in A(S)} x_{ij} \leq |S| - 1,$$

and the D^+ and D^-

$$x_{i_l i_1} + \sum_{h=1}^{l-1} x_{i_h i_{h+1}} + 2 \sum_{h=3}^{l} x_{i_1 i_h} + \sum_{j=4}^{l} \sum_{h=3}^{j-1} x_{i_j i_h} \leq l - 1,$$

$$x_{i_l i_1} + \sum_{h=1}^{l-1} x_{i_h i_{h+1}} + 2 \sum_{h=2}^{l-1} x_{i_h i_1} + \sum_{j=3}^{l-1} \sum_{h=2}^{j-1} x_{i_j i_h} \leq l - 1,$$

for any $l \in \{3, \ldots, |V| - 2\}$. However the incoming results can be extended to all those inequalities obtained from the lifted cycle inequalities. We now give an outline of a technique to prove that a certain face for P^{TPP} obtained by lifting the y variables from a lifted cycle facet of P^{TSP} is facet defining. Let F be a inequality facet defining

$$\sum_{e \in E} \alpha_e x_e \geq \eta,$$

for P^{TSP} obtained by sequential lifting of (4.13) on the x variables. And let F' the lifted valid inequality for P^{TPP}

$$\sum_{e \in E} \alpha_e x_e + \sum_{v_i \in V} \beta_i y_i \geq \eta',$$

obtained by sequential lifting of F on the y variables. A set X of $|A| - |V| + 1 - |M^*| + \sum_{p_k \in K \setminus K^*}(|M_k| - 1)$ (dim (P^{TPP})) affinely independent points satisfying with equality F' have to be found. Those point are obtained in three stages.

First, a set X^{TSP} of $|A| - 2|V| + 1$ (dim (P^{TSP})) point is built as follows. Let us consider the polyhedron $P := \{(x, y, z) \in P^{TPP} : y_i = 1, \text{ for all } v_i \in V \setminus M^*\}$, and let $P_x = P^{TSP}$ be the affine projection of P onto the x variables. Therefore, dim (P^{TSP}) affinely independent points in P^{TPP} are obtained by considering the dim (P^{TSP}) affinely independent tours for TSP satisfying F with equality, setting $y_i := 1$ for all $v_i \in V$ and setting the z variables with a feasible assignment, let us say, the first row of the matrix in Table 4.1. Obviously, for Lemma 4.4, and since F is facet defining for P^{TSP} those dim (P^{TSP}) points satisfy F' with equality and are affinely independent.

In the second stage a set $X^z$of $\sum_{p_k \in K \setminus K^*}(|M_k| - 1)$ (dim (P_z)) points is built by setting the x, y components as a feasible cycle visiting all vertices in V. The z components take the value of the dim (P_z) assignment values of Table 4.1. Because of how Table 4.1 is constructed these assignments are affinely independent.

Two sets X^s and X^t are constructed from the partition on the vertex set V induced by the vertices in C of (4.13), let us say $V(C)$ and $V \setminus (V(C) \cup M^*)$, in the third stage. X^s is constructed with $|V(C)| - 1$ affinely independent points $(x, y, z)^s \in P^{TPP}$, one for each $v \in V(C) \setminus \{v_p\}$, such that $y^s_s = 0$,and $y^s_i = 1$ for all $v_i \in V \setminus \{v_s\}$. That is, all the remaining vertices but v_s belong to the cycle. The z components are constructed according to a feasible assignment, and the x components are constructed such that a simple cycle pass throughout $V \setminus \{v_s\}$. In a similar way the set X^t is constructed, but taking into account the set $V \setminus (V(C) \cup M^*)$. Therefore, $|V| - |M*| - 2$ affinely independent cycles are constructed in this third stage.

The remaining two points are more specific, and depend on the right hand side. Let us see now four examples.

Theorem 4.10. *Inequalities*

$$\sum_{(i,j) \in A(S)} x_{ij} \leq \sum_{v_h \in S \setminus \{v_l\}} y_h, \tag{4.14}$$

are valid for P^{TPP}, for all $S \subset M \backslash M^*$, and $v_l \in S$, and define facet if $\sum_{v_i \in M_k \backslash S} q_{ki} \geq d_k$ for all products $p_k \in K$.

Proof.

We prove this by constructing a set X of $|A|-|V|+1-|M^*|+\sum_{p_k \in K \backslash K^*}(|M_k|-1)$ affinely independent points $(x, y, z) \in P^{TPP}$ satisfying (4.14) with equality.

Let us consider the polytope $P := \{P^{TPP} \cap y_i = 1, \text{ for all } v_i \in V \setminus M^*\}$, and let P_x be the affine projection of P on the x variables, already defined above. The inequality obtained from (4.14) by setting $y_i = 1$ for all $v_i \in V \setminus M^*$, that is, the subtour elimination constraint inequality associated with S is known to be facet defining for P_x (see Grötschel [77]) if $2 \leq |S| \leq |V| - 1$. Since dim(P_x) is $|A| - 2|V| + 1$ there is a set X^r with $|A| - 2|V| + 1$ affinely independent points $(x, 1, z)^r \in P^{TPP}$ with $y_i = 1$ for all $v_i \in V$, satisfying (4.14) with equality. For all those points, the z components have to contain a feasible assignment, for example, the first row of Table 4.1.

An additional set X^z with $\sum_{p_k \in K \backslash K^*}(|M_k| - 1)$ points $(x, 1, z)^z \in P^{TPP}$ are added to the set X. These new points are constructed by setting the x, y components as the first of the $|A| - 2|V| + 1$ points described above, and then satisfy (4.14) with equality. The z components take the value of the $\sum_{p_k \in K \backslash K^*}(|M_k| - 1)$ remaining values of Table 4.1. Because of how Table 4.1 is constructed these new points are affinely independent.

Let us suppose now that $\sum_{v_i \in M_k \backslash S} q_{ki} \geq d_k$ for all products $p_k \in K$. Then we can construct a set X^s of $|S| - 1$ affinely independent points $(x, y, z)^s \in P^{TPP}$, one for each $v \in S \setminus \{v_p\}$, such that $y^s_s = 0$, and $y^s_i = 1$ for all $v_i \in V \setminus \{v_s\}$. That is the remaining all the vertices but v_s belong to the cycle. The z components are constructed according to a feasible configuration, and the x components are constructed such that a simple cycle pass throughout $V \setminus \{v_s\}$. Because of the values taken by y these $|S|-1$ points are affinely independent (see Table 4.2) and satisfy (4.14) with equality.

Next we put into X a set X^t of $|V| - |S| - |M^*| - 1$ affinely independent points $(x, y, z)^t$ belonging to P^{TPP}. In this case, we follow a similar scheme to the one followed in the construction of the set X^s. For each vertex $v_t \in V \setminus (\{v_l\} \cup S \cup M^*)$, $y^t_{v_t} = 0$,and $y^t_v = 1$ for all $v \in V \setminus \{v_t\}$. The components x and z are constructed as in the previos set. Again, these points clearly exists, are affinely independent (see Table 4.2), belong to P^{TPP} and satisfy (4.14) with equality.

We need two additional points, with coefficient equal to 1 for the nodes v_k and v_l respectively. Let $(x, y, z)^k$ be a point such $y_i := 1$ for all vertices $v_i \in V \setminus S$ and $y_i = 0$ for $v_i \in S$. In addition, the x components are constructed such that this point be a simple cycle passing throughout the vertices in $V \setminus S$, and the z according to a feasible solution. The second point $(x, y, z)^l$ is constructed as follows. Let $(x, y, z)^l$ be a point such $y_i := 1$ for all vertices $v_i \in V \setminus S$ and $v_i \in S \setminus \{v_l\}$ and $y_l := 0$.

□

Table 4.2 Set X^s y X^t and points x^k, x^l and $x^{\bar{k}}$ affinely independent points.

	$y_{M^*,1}$	$\cdots$	$y_{M^*,\lvert M^*\rvert}$	$y_{S,1}$	$\cdots$	$y_{S,\lvert S\rvert-1}$	$y_{S,\lvert S\rvert}$	$y_{V\setminus S,1}$	$\cdots$	$y_{V\setminus S,\lvert V\setminus S\rvert-1}$	$y_{V\setminus S,\lvert V\setminus S\rvert}$
	1	$\cdots$	1	0		1	1	1	$\cdots$	1	1
X^s	$\vdots$		$\vdots$		$\ddots$		$\vdots$	$\vdots$		$\vdots$	$\vdots$
	1	$\cdots$	1	1		0	1	1	$\cdots$	1	1
	1	$\cdots$	1	1	$\cdots$	1	1	0		1	1
X^t	$\vdots$		$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\ddots$		$\vdots$
	1	$\cdots$	1	1	$\cdots$	1	1	1		0	1
x^k	1		1	0	$\cdots$	0	0	1		1	1
x^l	1		1	1	$\cdots$	1	0	1		1	1
$x^{\bar{k}}$	1		1	1	$\cdots$	1	1	1		1	0

Theorem 4.11. *Inequalities*

$$\sum_{(i,j)\in A(S)} x_{ij} \leq \sum_{v_h \in S} y_h - 1, \tag{4.15}$$

for $S \subset M$ are valid and define facet if

$$\sum_{v_i \in M_k \setminus S} q_{ki} < d_k \tag{4.16}$$

for a product $p_k \in K$.

Proof.

To prove that these inequalities are facet defining we constructs a set X of $|A| - |V|+1-|M^*|+\sum_{p_k \in K\setminus K^*}(|M_k|-1)$ affinely independent points $(x, y, z) \in P^{TPP}$ satisfying (4.14) with equality.

The proof is analogous to the previous one. In fact, it is identical until the two last points since the point x^k does not satisfy (4.16) with equality. Therefore, a new point let us say $x^{\bar{k}}$ has to be added to the set $X \setminus \{x^k\}$. This point is constructed by setting $y_i = 1$ for all $v_i \in V \setminus S$ and $v_i \in S \setminus \{v_k\}$, and $y_k = 0$. □

Let us focus now on the lifted cycle D_k^+ inequalities coming from

$$x_{i_l i_1} + \sum_{h=1}^{l-1} x_{i_h i_{h+1}} + 2\sum_{h=3}^{l} x_{i_1 i_h} + \sum_{j=4}^{l}\sum_{h=3}^{j-1} x_{i_j i_h} \leq l - 1, \tag{4.17}$$

already described above.

Theorem 4.12. *Inequalities*

$$x_{i_l i_1} + \sum_{h=1}^{l-1} x_{i_h i_{h+1}} + 2\sum_{h=3}^{l} x_{i_1 i_h} + \sum_{j=4}^{l}\sum_{h=3}^{j-1} x_{i_j i_h} \leq \sum_{h=1}^{l} y_{i_h} - y_2, \tag{4.18}$$

are valid and define facet for the P^{TPP}, for any $l \in \{3, \ldots, |M| - 2\}$ if for all products $p_k \sum_{v_i \in M_k \setminus S} q_{ki} \geq d_k$ for $S = \{v_{i_1}, \ldots, v_{i_l}\} \subset M$.

Proof. The following proof follows the same scheme in the previous f We prove this by constructing a set X of $|A| - |V| + 1$ affinely independent points $(x, y, z) \in P^{TPP}$ satisfying (4.18) with equality. The inequality (4.17) obtained by removing the coefficients corresponding to the vertices is facet defining for the ATSP polytope P_x (see Fischetti [51]); hence , there exists a set of $|A| - 2|V| + 1$ (dim(P_x)) affinely independent points $x \in P$ satisfying (4.17) with equality.

We initialize X by putting in it, for each of these $x \in P$, the point (x, y, z) defined by $y_i = 1$, $v_i \in V$. The variables z takes the value of the first point of Table 4.1, for each of those $|A| - 2|V| + 1$ points. Clearly, each point defined this way is in P^{TPP}, since (3.22) holds, satisfies (4.18) with equality, and they are affinely independent.

Additional $\sum_{p_k \in K}(|M_k|-1)$ points are added to our construction X. This new set of point is constructed by fixing one of the solutions described above with respect to the variables x and y, and varying the z components with the remaining points of Table 4.1, i.e., all points but the first one.

Let us suppose that $\sum_{v_i \in \{v_{i_{l+1}},\ldots,v_{i_n}\} \cap M_k} q_{ki} \geq d_k$ for $S = \{v_{i_1},\ldots,v_{i_l}\} \subset M$. Then we can construct, a set of $l-1$ affinely independent points $(x,y,z) \in P^{TPP}$ that satisfy (4.18) with equality. We put into $l-1$ points $(x,y,z)^s \in P^{TPP}$, one for each $v_{i_s} \in \{v_{i_1},\ldots,v_{i_l}\} \setminus \{v_{i_2}\}$, such that $y^s_{i_s} = 0$, $y^s_i = 1$ for all $v_i \neq v_{i_s}$, and $x^s_{i_j i_{j+1}} = 1$ for all $v_{i_j} \in \{v_{i_1},\ldots,v_{i_{s-2}},v_{i_{s+1}},\ldots,v_{i_l}\}$. In other words the cycle induced by x^s contains all of the nodes other than i_s, and all but two arcs, of the cycle $(v_{i_1},\ldots,v_{i_l},v_{i_1})$. Clearly, such $(x,y,z)^s$ exists, is in P^{TPP} and, and satisfies (4.18) with equality. Further, such $(x,y,z)^s$ has a unique component $y^s_{i_s}$ equal to 0, which is equal to 1 for all other points of X.

Next we put into X $n-k-1-|M^*|$ points $(x,y,z)^t \in P^{TPP}$ such that for each $v_{i_t} \in \{v_{i_{l+1}},\ldots,v_n\} \setminus (\{v_{i_p}\} \cup M^*)$, $y^t_{i_t} = 0$ $y^t_i = 1$ for all $i \neq i_t$, and $x^t_{i_j i_{j+1}} = 1$ for $j = 1,\ldots,l-1$. That is, the induced cycle contains all of the nodes, and all but one of the arcs, of the cycle $\{v_{i_1},\ldots,v_{i_l},v_{i_1}\}$. Again, these points clearly exist,are in P^{TPP}, and satisfy (4.18) with equality. Furthermore, they are affinely independent from each other and the remaining points of X. Notice that, the z components in both the points $(x,y,z)^s$ and the points $(x,y,z)^t$ contains a feasible assignment.

We need two additional points, with coefficient equal to 0 for the nodes v_{i_2} and v_{i_p} respectively. Let $(x,y,z)^{i_2}$ be such that $y^{i_2}_i = 0$ for all $v_i \in \{v_{i_1},\ldots,v_{i_l}\}$, $y^{i_2}_i = 1$ for $v_i \in \{v_{i_{l+1}},\ldots,v_{i_n}\}$, and the cycle induced by x^{i_2} has a node set $\{v_{i_{l+1}},\ldots,v_{i_n}\}$. A last point $(x,y,z)^{i_p}$ such that $y^{i_p}_{i_p} = 0$ and $y^{i_p}_i = 1$ for all $i \neq i_p$, and $x^{i_p}_{i_j i_{j+1}} = 1$ for $j = 1,\ldots,l-1$. Then, $(x,y,z)^{i_p}$ is in P^{TPP} and satisfy (4.18) with equality. Also, the points in X are affinely independent; thus (4.18) defines facet of P^{TPP}. □

Theorem 4.13. *Inequalities*

$$x_{i_l i_1} + \sum_{h=1}^{l-1} x_{i_h i_{h+1}} + 2\sum_{h=3}^{l} x_{i_1 i_h} + \sum_{j=4}^{l}\sum_{h=3}^{j-1} x_{i_j i_h} \leq \sum_{h=1}^{l} y_{i_h} - 1, \tag{4.19}$$

for $l := \{3,\ldots,|M|-2\}$, are valid and facet defining for P^{TPP} if $\sum_{v_i \in M_k \setminus S} q_{ki} < d_k$ for a product $p_k \in K$ or $M^* \cap S \neq \emptyset$ for $S = \{v_{i_1},\ldots,v_{i_l}\} \subset M$.

Proof. The proof of (4.19) is analogous to the previous proof, except for the point $(x,y,z)^{i_2}$. In this particular case, a new point $(x,y,z)^{\tilde{i_2}}$ replaces $(x,y,z)^{i_2}$, and it is defined as follows. $y^{\tilde{i_s}}_{i_s} = 0$ and $y^{\tilde{i_s}}_i = 1$ for all $i \neq i_p$, and $x^{i_p}_{i_j i_{j+1}} = 1$ for $j = 1,\ldots,l-1$. □

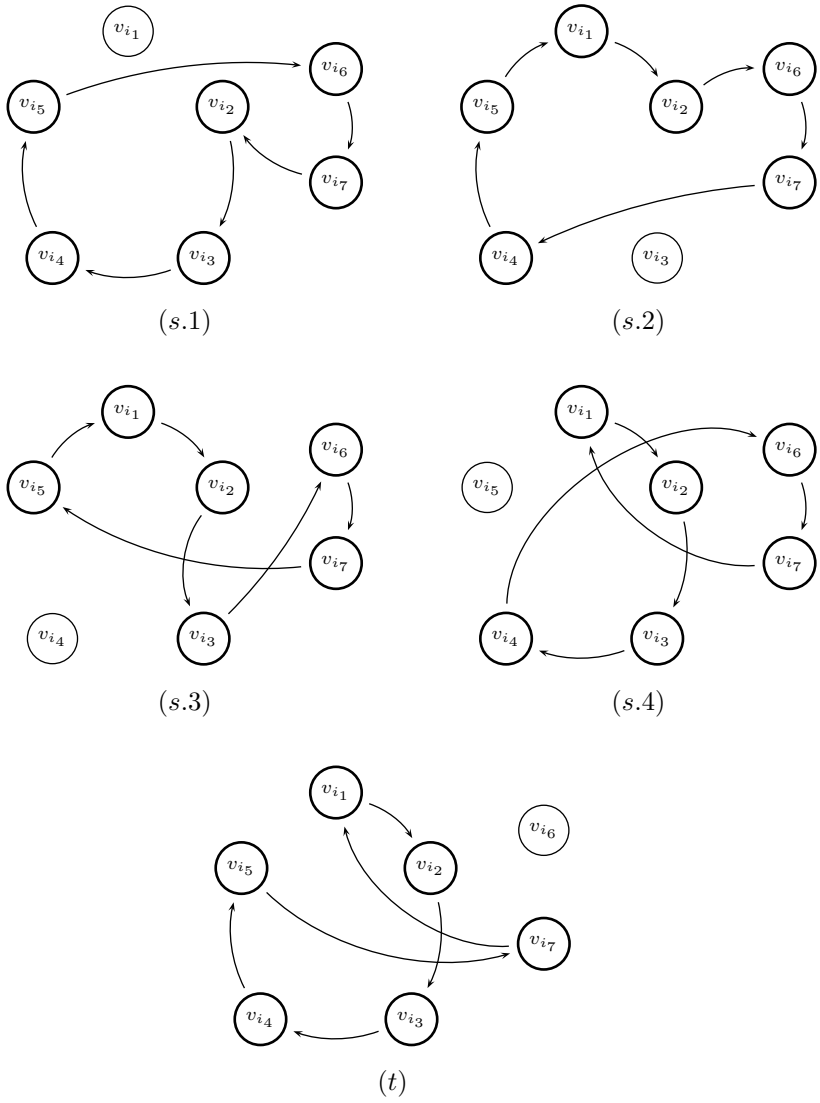


Fig. 4.3 Sets X^s and X^t of affinely independent points for facets from the cycle lifted inequalities.

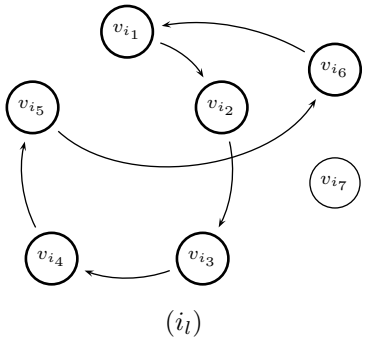


Fig. 4.4 Additional affinely independent point i_l for inequalities (4.18) and (4.19).

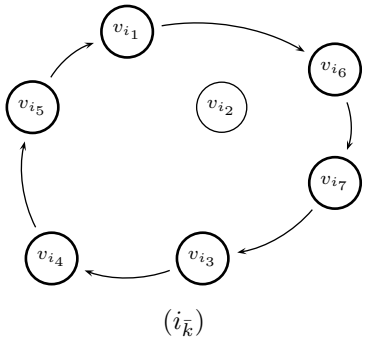


Fig. 4.5 Additional affinely independent point $i_{\bar{k}}$ for inequalities (4.18).

5

A Branch-and-Cut Algorithm for the STPP

The Branch-and-Cut technique applied to both the symmetric and asymmetric Traveling Purchaser Problem is described in the following two chapters. These algorithms are based on the models described in Chapter 3 and on the polyhedral analysis carried out in Chapter 4. In this chapter the implementation for the symmetric case is studied. An overview containing the main steps of this method is given in Section 5.1. This scheme is also used in the next chapter for the asymmetric case. We go deeply into the initial step and the initial heuristic in Sections 5.2 and 5.3 respectively. Section 5.4 studies the separation procedures of some of the valid inequalities described in the previous chapter. Column generation and different strategies in the branching phase are treated in Sections 5.5 and 5.6 respectively. Finally, our computational experience is shown in Sections 5.7.

5.1 THE BRANCH-AND-CUT SCHEME

In this section we give a short overview of the Branch-and-Cut technique which is also used in the next chapter applied to the asymmetric version of the TPP. For a extensive and comprehensive description of this method we refer to the reader to Padberg and Rinaldi [119], Jünger and Thienel [89], Jünger, Reinelt and Thienel [88], Thienel [145], and Caprara and Fischetti [27].

The *linear relaxation* of an integer linear program (IP) is the linear program obtained from IP by dropping the constraints that all variables have to be integer. For example, the linear relaxation of STPP and ATPP are obtained by dropping the integrality constraints. Therefore, the optimal value w^{LP} of the relaxation (in the

minimization case) is a lower bound to the optimal value w^{IP} of the integer linear program, i.e., $w^{LP} \leq w^{IP}$.

Branch-and-Cut is a solution technique to solve integer linear programs with an exponential, or at least very large, number of constraints. These constraints are generally only known implicitly and not explicitly. The Branch-and-Cut method is illustrated by pseudocode in Figures 5.1, 5.2 and 5.3.

```
Input:  Π
Output:  z, ω^UB, status
  PRE-PROCESING( Π, status );
  if ( status = infeas )  return;
  INITIAL-HEURISTIC( Π, z, ω^UB );
  L^P := L^P ∪ Π;
  while ( L^P ≠ ∅ )
    extract Π from L^P;
    OPTIMIZE-SUBPROBLEM ( Π, z, ω^UB, status );
    if ( status = Branching ) BRANCHING ( Π, L^P );
```

Fig. 5.1 The Branch-and-Cut method.

```
Input:  Π, z, ω^UB
Output:  Π, z, ω^UB, status
   status := Optimize;
  while ( status = Optimize )
    SOLVE-LP ( Π);
    if ( ( Π.status = Infeas ) or ( Π.ω^LB ≥ ω^UB ))
      status := Fathom;
    else
      if ( FEASIBLE ( Π, z, ω^UB)) then
        status := Fathom;
      else
         PRIMAL-HEURISTIC ( Π, z, ω^UB);
         SEPARATE-ELIMINATE ( Π, #cuts );
          if ( #cuts = 0 )
            status := Branching;
```

Fig. 5.2 Optimizing node. Branch-and-Cut.

A generic optimization problem is referred as Π in this pseudocode. The problem Π is a data structure that consists of the list of active constraints $\mathcal{L}^C$, the list of active variables $\mathcal{L}^V$, the optimization vector w, the solution vector of the current linear relaxation z^*, the objective value of the current linear relaxation ω^{LB} according to w, and the status of linear relaxation (status).

According to Fig. 5.1, in a basic sketch of a Branch-and-Cut method for a minimization problem, a list $\mathcal{L}^P$ of subproblems is initialized with the linear relaxation of the problem Π. The value of best solution z found so far is stored in the global upper bound ω^{UB}. Each major iteration step (referred as procedure OPTIMIZE-SUBPROBLEM in Figures 5.2 and 5.3), proceeds by selecting a subproblem from the list. A local lower bound is computed for this subproblem by solving the linear

```
Input:  Π, z, ω^UB
Output: Π, z, ω^UB, status
   status := Optimize;
  while ( status = Optimize )
    repeat
      SOLVE-LP ( Π, status );
      if ((( Π.status = Infeas )  or ( Π.ω^LB ≥ ω^UB ))
        PRICING ( Π, #var );
        status := Fathom;
    until ((Π.status ≠ Infeas) and (ω^LB < ω^UB)) or (#var ≠ 0));
    if ( FEASIBLE (( Π, z, ω^UB)) and ( status ≠ Fathom ))
      status := Fathom;
    else
      PRIMAL-HEURISTIC ( Π, z, ω^UB);
      repeat
        SEPARATE-ELIMINATE ( Π, #cuts );
        if ( #cuts = 0 )
          PRICING ( Π, #var );
          status := Branching;
      until ( #cuts + #var ≠ 0 );
```

Fig. 5.3 Optimizing node. Branch-and-Cut-and-Price.

relaxation with the current set of active constraints and active variables (SOLVE-LP), and the algorithm tries to improve the global upper bound by mean a heuristic based on solution obtained in the current linear relaxation (PRIMAL-HEURISTIC). A subproblem is fathomed from the list if either:

- a integer feasible solution for the original problem is obtained from de linear relaxation,
- the linear relaxation is infeasible,
- the local lower bound associated with the subproblem is greater than the global upper bound.

If the current linear relaxation does not become a feasible solution of the original problem (FEASIBLE) then an attempt is made to generate valid inequalities violated and drop non-violated inequalities by the current solution, using separation procedures (SEPARATE-ELIMINATE). New inequalities are added to the list $\mathcal{L}^C$ (#cuts $\neq 0$) as long as they are generated, and, in order to avoid a huge list, they are ranked according to how violated their are. This procedure (OPTIMIZATION-NODE) is repeated until either no more violated constraints are found, or the subproblem is fathomed. If not violated constraints are found then the branching phase (BRANCHING) is launched. The current subproblem is then divided into disjoint subproblems whose union of feasible solutions contains all feasible solution of the current subproblem. As soon the list of subproblems is empty the algorithm stops, and then the global upper bound can be output as the optimum solution.

Nevertheless, since the number of variables is also very large due partly to the presence of z_{ki} variables. For example, an instance with 200 markets and 200 products

could have 40000 z_{ki} variables, a large number of columns for the LP relaxation in a cutting-plane approach. We have therefore devised a solution method using variable generation and pricing, on top of the usual constraint relaxation scheme. To operationalize this mechanism, and based on our computational experiments, the algorithm works with dynamically augmented subsets or "pools" of variables and constraints previously mentioned and called $\mathcal{L}^V$ and $\mathcal{L}^C$, respectively. The procedure OPTIMIZATION-SUBPROBLEM described in Fig. 5.2 shows a variation of the original optimization procedure. Notice that new variables are added to the pool $\mathcal{L}^V$ if

- linear relaxation becomes infeasible,
- the lower bound computed by the linear relaxation is greater than the global upper bound,
- no new cuts are generated by the separation procedure.

We next address the main steps of our algorithm. A deeper explanation of the procedures involved in the Branch-and-Cut is provided in the following sections, taking into account preprocessing, separation, heuristics, and branching scheme. The tactical choices for the parameters were fixed from our computational experiences solving the instances described in Sections 5.7 and 6.4.

5.2 PREPROCESSING

The preprocessing phase tries to build the initial model from the input data. This step involves checking feasibility, initializing the variables and constraints pool, and fixing some variables in order to reduce the size of the system.

1. Check feasibility

   This steps checks whether $\sum_{v_i \in M_k} q_{ik} \geq d_k$ for all $p_k \in K$. An $O(|V||K|)$ time complexity algorithm checks for each product $p_k \in K$ whether the offer satisfies the total demand d_k. If it does not, the program exits with an error message.

2. Initial pool of variables

   This step defines the initial pool of variables $\mathcal{L}^V$, by including all variables x_e for all $e \in E$ associated with ten least cost edges incident to each vertex, and variables z_{ki} for all $p_k \in K$ and $v_i \in M_k$ corresponding to the ten cheapest markets for each product.

3. Initial pool of constraints

   Define $\mathcal{L}^C$ by including constraints (3.5), (3.11), all constraints (3.15) with $S := M_k$ for all p_k, as well as the lower and upper bounds on the variables.

4. Initial heuristic

In order to obtain a initial value of ω^{UB}, a feasible solution is computed by an simple heuristic described in section 5.3. In addition, the initial feasible solution will provide to the variables pool as well as to the initial linear programming the variables associated to the set of arcs and vertices belonging to this solution, which guarantee the feasibility of the initial linear programming.

5. Initial Linear programming

   The initial linear programming consist of

   (a) the set of all variables y_i for all $v_i \in V$ associated to the set of markets, all variables x_e with edges belonging to the heuristic solution and the five least cost edges incident to each vertex, and all variables z_{ki} corresponding to the assignments made in the heuristic solution and to the five cheapest markets for each product;

   (b) the set of constraints consisting of equalities (3.2) and (3.4);

   (c) and the lower and upper bounds on the variables.

This initial linear programming initializes the list of subproblems $\mathcal{L}^P$.

5.3 HEURISTICS

Two heuristics procedures have been developed in this Branch-and-Cut algorithm for obtaining feasible solutions. The *initial heuristic* builds a feasible solution from the input data, and the *primal heuristic* tries to improve the upper bound ω^{UB} from the current LP-relaxation.

Initial Heuristic

The heuristic used to construct a first incumbent solution works along the lines of the methods described in Ong [116] and Pearn and Chien [123].

An initial solution containing all vertices is built by the well-known nearest-neighbor TSP heuristic, and it is then improved by the Lin-Kernighan procedure (see [106] for details).

The method gradually reduces the initial cycle by dropping at each step a new market v_i^*. For each vertex v_i belonging to current cycle, the tour reduction $\mathcal{T}(v_i)$, as well as the increasing in the purchasing cost $\mathcal{P}(v_i)$ after removing v_i is computed. If dropping the vertex v_i could cause infeasibility, then v_i will be removed from the list of potential candidates. The vertex v_i^* that maximizes $\mathcal{T}(v_i) + \mathcal{P}(v_i)$ will be selected for be dropped.

If no further reduction is possible, then an insertion procedure is performed. For each vertex v_i not belonging to the current cycle the tour increasing $\hat{\mathcal{T}}(v_i)$, as well as the purchasing reduction $\hat{\mathcal{P}}(v_i)$ is computed. If $\hat{\mathcal{P}}(v_i) > \hat{\mathcal{T}}(v_i)$ then v_i is added to the cycle. This procedure is repeated until no further improvement is possible.

After each removal/insertion a post-optimization procedure is carried out by this algorithm. In particular, we use a simple 2-opt mechanism in order to improve the tour. With respect to the purchasing cost, for a given set of markets is easy to compute the optimal assignment of products. Finally, a cleaning procedure dropping all those markets without influence in the purchasing cost is performed.

LP-Based Heuristic

The primal heuristic is applied from a fractional LP solution (x^*, y^*, z^*) in order to construct a new feasible solution improving the best upper bound computed so far. Since the primal heuristic is frequently applied, emphasis has been put on speed. The heuristic, that consist of two phases, proceed as follow.

In the first phase, the set of candidate markets has to be established. Accordingly, an initial set of market $\hat{M}$ is built from those markets $v_i \in V$ such as $y_i = \rho$, where ρ is a threshold initially set to 1. If not enough markets are selected for making a feasible solution, then ρ is decreased in a small amount. This procedure is repeated until feasibility is restored.

The second phase tries to build a Hamiltonian cycle passing through the vertices of set $\hat{M}$. This procedure follows the same scheme described in the previous phase. That is, the threshold ρ selects those edges belonging to the solution. If is not possible to close the cycle then the nearest-neighbour method is performed over the no connected markets.

After these two phases, a post-optimization procedure is carried out applying a 2-opt algorithm if triangle inequality hold for the current instance.

5.4 SEPARATION PROCEDURES

As already mentioned in section 1.4 of chapter 1, it is not necessary an explicit list of constraints defining the original problem. It is only required a method for identifying inequalities that are valid for the original problem, but violated by the current linear relaxation, i.e., given a fractional solution obtained from the linear relaxation (x^*, y^*, z^*), a separation procedure consists of determining a member $\alpha x + \beta y + \gamma z \geq \eta$ of a given family of valid TPP inequalities such that $\alpha x^* + \beta y^* + \gamma z^* < \eta$.

The aim of this section is describing the separation method for families (3.3), (3.16), (3.15) and (3.12). These separation problems are quite standard (see, e.g., Jünger, Reinelt and Rinaldi [87]) and are solved exactly in polynomial time for the first two cases, while heuristic approaches are used for the last two.

Let us define the *support graph* for a fractional solution (x^*, y^*, z^*) as the weighted graph $(G^*, x^*, y^*, z^*) = (V^*, E^*, x^*, y^*, z^*)$ where $V^* := \{v \in V | y_v^* > 0\}$,$E^* := \{e \in E | x_e^* > 0\}$.

Separation of the INCOMPA Inequalities

This section describes the separation algorithm for the family of constraints (3.5) (INCOMPA). Despite of be a polynomial number of inequalities ($|M||K|$), we prefer perform a dynamic separation. Given a fractional solution (x^*, y^*, z^*), separating a INCOMPA constraints is to find a market $v_i \in M$ and a product $p_k \in K$ such as $z^*_{ki}/q_k > y^*_i$. An exhaustive algorithm for checking if this inequality is violated is carried out by checking for each market and each product available in this market the inequality is violated. Notice that, only those markets v_i such $y^*_i > 0$ should be taken into account. Fig. 5.4 describes this simple algorithm.

```
Input:  (G*, y*, z*),  L^C
Output: L^C
  for all v_i ∈ M
    if ( y*_i > 0 )
      for all p_k ∈ K
        if (z_ki/q_k > y_i)
          L^C := L^C ∪ z_ki/q_k ≤ y_i
```

Fig. 5.4 Separation of the INCOMPA inequalities.

Separation of YSEC Inequalities

Proposition 5.1. *Given a market v_i with $y^*_i > 0$, a most violated YSEC constraint (3.3) corresponds to a minimum-capacity cut (min-cut) $(S, V^* \setminus S)$ with $v_i \in S$ and $v_0 \notin S$ in the support graph G^* by imposing a capacity x^*_e on each edge $e \in E^*$.*

In practice, two cases are taken into account in order to compute the separation.

1. The solution is not connected (see separation algorithm in Fig. 5.5). In this case for each connected component $S \subseteq V \setminus \{v_0\}$ the inequalities

$$\sum_{e \in \delta(S)} x_e \geq 2y_i, \qquad \text{for all } v_i \in S,$$

   are violated. The procedure (COMPONENTS) computing each connected component takes a time complexity (see Mehlhorn [110]) of $O(|V| + |E|)$.

2. The solution is connected, or is not connected but a connected component $S \subseteq V$ containing the depot is being studied. For this other case, a min-cut algorithm (MINCUT) based on an implementation described in Goldberg and Tarjan [71] is performed between every pair (v_0, v_i) for each $v_i \in S \subseteq \setminus\{v_0\}$, in order to obtain the most violated inequalities. This process takes a time complexity of $O(|V|^4)$.

In order to reduce the number of non-zero when a new YSEC cut is added to the LP, the constraint is introduced in the following way.

$$\sum_{e \in E(S)} x_e - \sum_{v_j \in S \setminus \{v_i\}} y_j \leq 0, \tag{5.1}$$

for a violated set of vertices $S \subseteq V \setminus \{v_0\}$ and a vertex $v_i \in S$. Notice that (3.3) imply (5.1), since adding (3.2) for all vertex $v_i \in S$ we obtain

$$2 \sum_{e \in E(S)} x_e + \sum_{e \in \delta(S)} x_e = 2 \sum_{v_i \in S} y_i. \tag{5.2}$$

Replacing (5.2) in (3.3) we obtain (5.1).

Fig. 5.6 shows a fractional point violating the following (3.3).

$$\sum_{e \in \delta(S)} x_e \geq 2y_6$$

With $S := \{v_1, v_3, v_4, v_5, v_6, v_9\}$, since $\sum_{e \in \delta(S)} x_e = 0.5$ and $y_6 = 1$, and then $0.5 \not\geq 2y_6$.

Input: (G^*, x^*, y^*), $\mathcal{L}^C$
Output: $\mathcal{L}^C$
COMPONENTS($G^*, \mathcal{C}$)
for all $S \in \mathcal{C}$
if ($v_0 \notin S$)
for all $v_i \in S$
$\mathcal{L}^C := \mathcal{L}^C \cup \sum_{e \in \delta(S)} x_e \geq 2y_i$
else
for all $v_i \in S \setminus \{v_0\}$
cut-value := MINCUT(G^*, v_0, v_i, S');
if (cut-value $< 2y_i$)
$\mathcal{L}^C := \mathcal{L}^C \cup \sum_{e \in E(S)} x_e - \sum_{v_j \in S \setminus \{v_i\}} y_j \leq 0$

Fig. 5.5 Separation of the YSEC inequalities.

Separation of ZSEC Inequalities

Proposition 5.2. *Given a product p_k, determining a most violated ZSEC constraint (3.16) is equivalent to finding a subset S with minimum value of*

$$\sum_{e \in \delta(S)} x_e^* + \sum_{v_i \in M_k \setminus S} 2\frac{z_{ki}^*}{d_k}. \tag{5.3}$$

Observe that equations (3.4) imply (5.3), since

$$\sum_{e \in \delta(S)} x_e \geq 2 \sum_{v_i \in M_k \cap S} \frac{z_{ki}}{d_k},$$

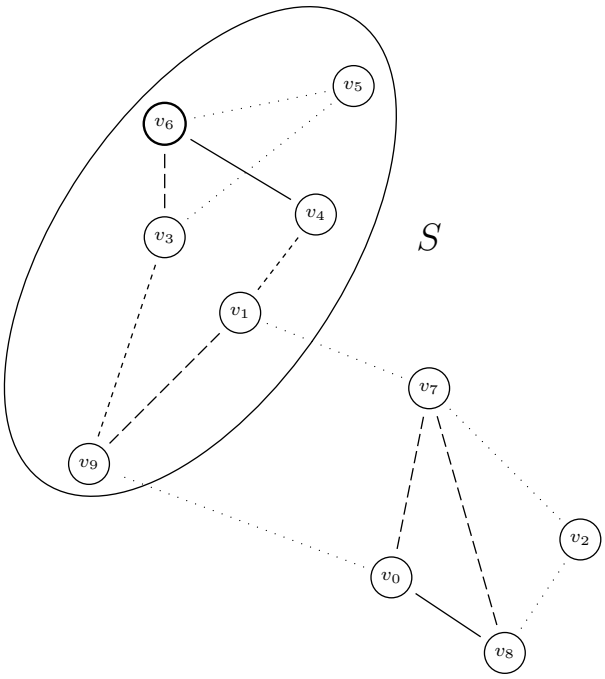


Fig. 5.6 Fractional point violating a YSEC inequality.

$$\sum_{e\in\delta(S)} x_e + 2\sum_{v_i\in M_k\setminus S}\frac{z_{ki}}{d_k} \geq 2\sum_{v_i\in M_k\cap S}\frac{z_{ki}}{d_k} + 2\sum_{v_i\in M_k\setminus S}\frac{z_{ki}}{d_k},$$

$$\sum_{e\in\delta(S)} x_e + 2\sum_{v_i\in M_k\setminus S}\frac{z_{ki}}{d_k} \geq 2\sum_{v_i\in M_k}\frac{z_{ki}}{d_k}.$$

But by (3.4) $\sum_{v_i\in M_k} z_{ki}/d_k = 1$.

This reduces the separation problem to a maximum-flow problem defined on the following weighted graph. Consider a dummy market $\hat{v}$, and let $\hat{G} := (\hat{V}, \hat{E})$, where $\hat{V} := V^* \cup \{\hat{v}\}$ and $\hat{E} := E \cup \{[v_i, \hat{v}] : v_i \in M_k\}$. The capacity of edge $e \in E$ is x^*_e, and the capacity of each new edge $[v_i, \hat{v}]$ is equal to $2z^*_{ki}/d_k$. Let $(S', \hat{V} \setminus S')$ be a minimum-capacity cut in $\hat{G}$ separating v_0 and $\hat{v}$, with $\hat{v} \in S'$. If the capacity of this cut is at least 2, then (x^*, y^*, z^*) satisfies all constraints (3.16) associated with p_k. Otherwise, $S := S' \setminus \{\hat{v}\}$ yields a most violated constraint (3.16). This algorithm is described by pseudocode in Fig. 5.7 and its time complexity is $O(|K||V|^3)$.

As in the previous separation algorithm, in order to reduce the non-zero elements the ZSEC cuts are introduced in the LP as follows.

$$\sum_{e\in E(S)} x_e - \sum_{v_i\in S} y_i + \sum_{v_u\in S\cap M_k}\frac{z_{ki}}{d_k} \leq 0.$$

This inequality is equivalent to (3.16) if we replace (5.2) in (3.16).

Fig. 5.8 shows a fractional solution violating (3.16). In this particular case $0.75/d_3$ and $0.25/d_3$ units of the product p_3 are offered at markets v_1 and v_4 respectively. The subset S' consist of the vertices $S' := \{\hat{v}, v_1, v_3\}$ and according to Fig. 5.8 it follows $\sum_{e\in\delta(S'\setminus\hat{v})} x_e^* = 1$ and $\sum_{v_i\in M_3} z_{i3}/d_3 = 1$. Therefore inequality

$$\sum_{e\in\delta(S)} x_e \geq \frac{2}{d_k} \sum_{v_i\in S\cap M_3} z_{3i}, \qquad\qquad S := \{v_1, v_4\}$$

is violated.

Input: (G^*, x^*, y^*, z^*), $\mathcal{L}^C$
Output: $\mathcal{L}^C$
  for all $p_k \in K$
    $\hat{V} := V^* \cup \hat{v}$
    for all $v_i \in M_k$
      $\hat{E} := E^* \cup e = [\hat{v}, v_i]$
      $x_e^* = 2\frac{z_{ik}^*}{d_k}$
    $\hat{G} = (\hat{V}, \hat{E})$
    `cut-value := MINCUT(` $\hat{G}$`,`v_0`,`$\hat{v}$`,` S' `);`
    if `( cut-value` < 2 `)`
      $S := S' \setminus \{\hat{v}\}$
      $\mathcal{L}^C := \mathcal{L}^C \cup \sum_{e\in\delta(S)} x_e^* \geq 2\sum_{v_i\in M_k\cap S} \frac{z_{ki}}{d_k}$

Fig. 5.7 Separation of the ZSEC inequalities.

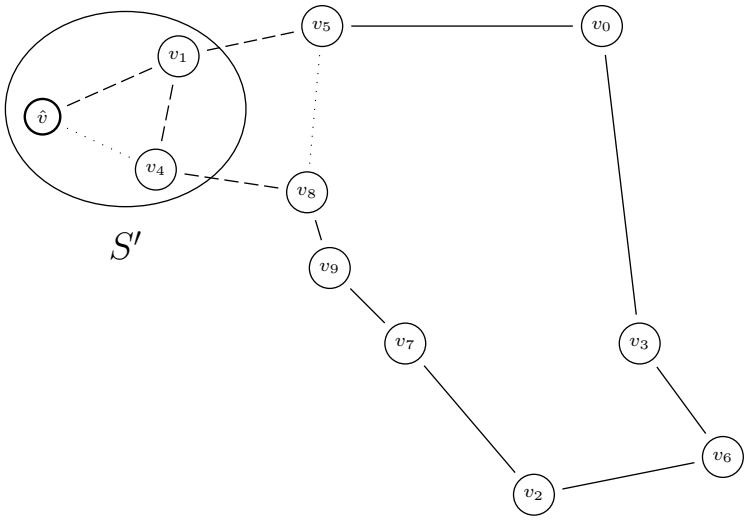


Fig. 5.8 Fractional point violating a ZSEC inequality.

Separation of 2SEC Inequalities

Two heuristics methods have been developed in order to separate constraints 2SEC (3.15).

1. The first heuristic proceed as follows. Let us consider a product p_k and determine a set $M'_k \subseteq M_k$ such that $\sum_{v_i \in M_k \setminus M'_k} q_{ki} < d_k$, starting with $M'_k := M_k$ and cumulating markets v_i in M_k as long as $\sum_{v_i \in M_k \setminus M'_k} q_{ki}$ does not exceed d_k. Then construct a graph $\hat{G} = (\hat{V}, \hat{E})$, where $\hat{V} := V \cup \{\hat{v}\}$, $\hat{v}$ is a dummy market, and $\hat{E} := E \cup \{[v_i, \hat{v}] : v_i \in M'_k\}$. The capacity x^*_e of edge $e \in E$ is x^*_e, and the capacity $x^*_{[v_i,\hat{v}]}$ of each new edge $[v_i, \hat{v}]$ is equal to 2. As above, let us determine a minimum-capacity cut $(S', \hat{V} \setminus S')$ in $\hat{G}$ with $\hat{v} \in S'$ and separating v_0 and $\hat{v}$. If the capacity of this cut is less than 2, a violated constraint (3.15) with $S := S' \setminus \{\hat{v}\}$ has been identified. Otherwise, M'_k is modified by means of a vertex interchange mechanism and the procedure is repeated. In total, at most $|M_k|$ candidate sets M'_k are considered. Note that in the case of the TPP with unlimited supplies, $M'_k = M_k$ and this separation procedure is exact and runs in $O(|K||V|^3)$ time.

Input: (G^*, x^*, y^*, z^*), $\mathcal{L}^C$
Output: $\mathcal{L}^C$
  for all $p_k \in K$
    $\hat{V} := V^* \cup \hat{v}$
    for all $v_i \in M_k$
      $\hat{E} := E^* \cup e = [\hat{v}, v_i]$
      $x^*_e = 2$
    $\hat{G} = (\hat{V}, \hat{E})$
    `cut-value := MINCUT(` $\hat{G}$`,`v_0`,`$\hat{v}$`,` S' `);`
    if `(` `cut-value` $\leq 2y_i$ `)`
      $S := S' \setminus \{\hat{v}\}$
      $\mathcal{L}^C := \mathcal{L}^C \cup \sum_{e \in \delta(S)} x^*_e \geq 2 \sum_{v_i \in M_k \cap S} \frac{z_{ki}}{d_k}$

Fig. 5.9 Separation of the 2SEC inequalities: Heuristic 1.

2. In the second heuristic, we first attempt to identify a set S yielding a violation of the weaker cover inequalities (3.13). If this is successful then constraint (3.15) associated to S is also violated. Otherwise, we still check whether a violation of (3.15) has been identified. As in Crowder, Johnson and Padberg [38], constraints (3.13) can easily be separated by solving the 0-1 *Knapsack Problem* (KP)

$$\sigma^* := \max\Big\{ \sum_{v_i \in M_k} y^*_i u_i : \sum_{v_i \in V_k} q_{ik} u_i \leq d_k - \epsilon, u_i \in \{0,1\} \text{ for all } v_i \in M_k \Big\},$$

where ϵ is a small positive value (if all q_{ik} and d_k are integer numbers, then $\epsilon := 1$). Indeed, S is defined by the set of markets v_i with $u^*_i = 0$ in the optimal KP solution. If $\sigma^* > \sum_{v_i \in M_k} y^*_i - 1$, then constraints (3.13) associated with

S is violated. Otherwise all constraints (3.13) associated with p_k are satisfied. The KP is relatively easy to solve (see, e.g., Martello and Toth [109]), and its size can in fact be reduced by fixing to 1 all variables u_i with $y_i^* = 1$ since $u_i = 0$ would imply $\sigma^* > \sum_{v_i \in M_k} y_i^* - 1$. Similarly, u_i can be fixed to 0 whenever $y_i^* = 0$ since in this case its weight in the KP objective function vanishes.

Notice that a violated inequality (3.3) or (3.16) identified through the above procedures is not necessarily facet-defining. This only occurs when there exists a product p_k that cannot be entirely purchased outside S. Therefore, whenever a violation of (3.3) or (3.16) occurs for a given p_k and S, a check should be made for a violation of the stronger (and facet-defining) constraint (3.15) over the same S. Finally, although the complexity of the above separation procedures may seem rather high, these can be executed quite rapidly since $\hat{G}$ is typically sparse and contains many isolated vertices. Moreover, for constraints (3.3), several maximum-flow computations can be avoided since some y_i^* values are very small.

Separation of 2-matching Inequalities

The 2-matching inequalities (3.12) can be separated in polynomial time through a simple modification of the Padberg and Rao [121] odd-cut separation scheme. However, in order to reduce the computational effort spent in the separation, we have implemented the following simple heuristic initially proposed by Fischetti, Salazar and Toth [54], also illustrated in Fig. 5.10.

In order to obtain a set of potential handles, the set $\mathcal{H} \subset \mathcal{P}(V)$ of connected components from the subgraph $G_\rho = (V, E_\rho)$ induced by $E_\rho := \{e \in E : 0 < x_e^* < \rho\}$ is computed, for every threshold ρ corresponding to an x_e^* value. Each vertex set $H \in \mathcal{H}$ becoming a clique is then considered as a potentially violated handle of a 2-matching constraint. For each one of these vertex sets tooth edges are determined by the following simple greedy procedure. Let $\delta(H) = \{e_1, \ldots, e_p\}$ with $x_{e_1}^* \geq x_{e_2}^* \geq \ldots \geq x_{e_p}^*$. The requirement that the teeth have to be pairwise disjoint is initially relaxed. Among those edge sets T satisfying $|T| \geq 3$ and odd, the best choice for T consists of edges $e_1, \ldots, e_{|T|}$. Therefore a most violated inequality corresponds to the choice of the odd integer $|T| \geq 3$ maximizing $x_{e_1}^* + (x_{e_2}^* + x_{e_3}^* - 1) + \cdots + (x_{e_{|T|-1}}^* + x_{e_{|T|}}^* - 1)$. If no violated cut could be produced in this way, then clearly no violated 2-matching constraint exists for the given handle. Otherwise we have a violated 2-matching constraint, in which two tooth edges, say e and f, may be incident to the same vertex v. In this case, we simplify the inequality by defining a new handle-teeth pair (H', T') with $T' := T \setminus \{e, f\}$, and $H' := H \setminus \{v\}$ (if $v \in H$) or $H' := H \cup \{v\}$ (if $v \notin H$). It is then easy to see that the new 2-matching inequality is stronger violated than the previous one. Indeed, replacing (H, T) with (H', T') increases the violation by at least $1 + y_i - \sum_{e \in \delta(\{v_i\})} x_e \geq 2y_i - \sum_{e \in \delta(\{v_i\})} x_e = 0$ (if $v_i \in H$), or $1 - y_i \geq 0$ (if $v_i \notin H$).

By performing this simplification step a 2-matching constraint can be detected with non-overlapping teeth. In some cases this procedure could even lead to a 2-

matching constraint with $|T| = 1$; if this occurs, we reject the inequality in favour of a constraint YSEC (3.3) or 2SEC(3.15) associated with the handle.

Fig. 5.11 illustrates a fractional point violating two 2-matching inequalities. Two handles, $H_1 := \{v_2, v_5, v_6\}$ and $H_2 := \{v_1, v_4, v_8\}$, have been detected by the previously described algorithm. And for each of them teeth set is also computed, i.e., $T_1 := \{e_{[v_1,v_7]}, e_{[v_4,v_5]}, e_{[v_8,v_3]}\}$ for handle H_1 and $T_2 := \{e_{[v_2,v_9]}, e_{[v_4,v_5]}, e_{[v_6,v_0]}\}$ for handle H_2.

Input: (G^*, x^*), $\mathcal{L}^C$, ρ
Output: $\mathcal{L}^C$
  for all $e \in E^* : x_e^* > \rho$
    $E_\rho := E_\rho \cup e$
  $V_\rho := V(E_\rho)$
  `COMPONENTS(`$G_\rho, \mathcal{H}$`)`
  for all $H \in \mathcal{H} : H$ `is a clique`
    $T := \arg\{\max_{T' \subset \delta(H)} \sum_{e \in T'} x_e - (|T'| - 1) : |T'| \geq 3$ `and odd`$\}$
    for all $e, f \in T$ `such as` $v := e \cap f \neq \varnothing$
      $T := T \setminus \{e, f\}$
      if `(` $v \in H$ `)` $H := H \setminus v$
      else $H := H \cup v$
    if `(` $(\sum_{e \in T} x_e - \sum_{e \in \delta(H) \setminus T} x_e > |T| - 1)$ ***and*** $(|T'| \geq 3$ `and odd`$))$
      $\mathcal{L}^C := \mathcal{L}^C \cup \sum_{e \in T} x_e - \sum_{e \in \delta(H) \setminus T} x_e \leq |T| - 1$

Fig. 5.10 Heuristic separation of the 2-matching inequalities.

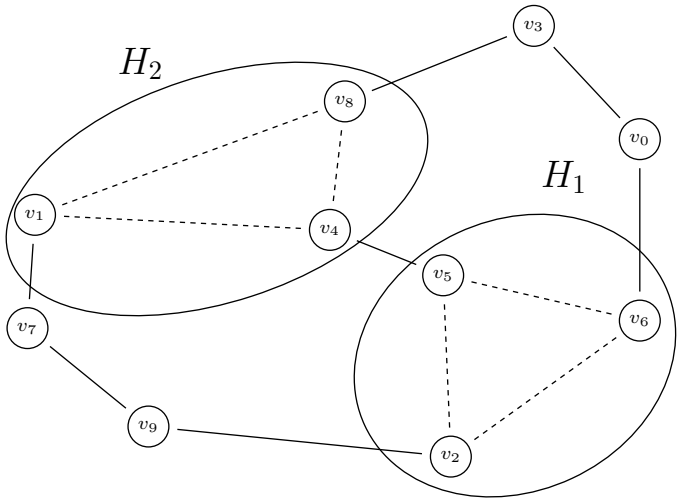


Fig. 5.11 Fractional point violating two 2-matching inequalities.

Clique Lifting and Shrinking An important application in the design of separation algorithm in that it allows one to simplify the separation problem is the shrinking procedure, proposed by Padberg and Rinaldi [117].

As a complementary concept of the shrinking *clique lifting* can be described as follows (see Balas and Fischetti [11]). Let $P(G')$ denote the $ATSP$ polytope associated with a complete digraph $G' = (V', A')$. Given a valid inequality $\beta y \leq \beta_0$ for $P(G')$, we define

$$\beta_{hh} := \max\{\beta_{ij} + \beta_{hj} - \beta_{ij} : i, j \in V' \setminus \{h\}, i \neq j\} \text{ for all } h \in V'$$

and construct an enlarge complete digraph $G = (V, A)$ obtained from G' by replacing each node $h \in V'$ by a clique S_h containing at least one node (hence, $|V| = \sum_{h \in V'} |S_h| \geq |V'|$). In other words $(S_1, \ldots, S_{|V'|})$ is a proper partition of V, in which the h-th set corresponds to the h-th node in V'.

For all $v \in V$, let $v \in S_{h(v)}$. We define a new *clique lifted* inequality for $P(G)$, say $\alpha x \leq \alpha_0$, where $\alpha_0 := \beta_0 + \sum_{h \in V'} \beta_{hh}(|S_h| - 1)$ and $\alpha_{ij} := \beta_{h(i)h(j)}$ for each $(i, j) \in A$. Balas and Fichetti [11] show that the new inequality is always valid for $P(G)$; in addition, if the starting inequality $\beta x \leq \beta_0$ defines a facet for $P(G')$, then $\alpha x \leq \alpha_0$ is guaranteed to be facet-inducing for $P(G)$.

Let $S \subset V$, $2 \leq |S| \leq n - 2$, be a vertex subset saturated by x^*, in the sense that $x^*(E(S)) = |S| - 1$, and suppose S is shrunk into a single node, say σ, and x^* is updated accordingly. Let $G' = (V', A')$ denote the shrunken digraph, where $V' := V \setminus S \cup \{\sigma\}$, and let y^* be the shrunken couterpart of x^*. Every valid inequality $\beta y \leq \beta_0$ for $P(G')$ that is violated by y^* correspond in G to a violated inequality, say $\alpha x \leq \alpha_0$, obtained through *clique lifting* by replacing back σ with the original set S. As observed Padberg and Rinaldi [118], this shrinking operation can affect the possibility of detecting violated cuts on G', as it may produce a point y^* belonging to $P(G')$ even when $x^* \notin P(G)$.

There are simple conditions on the choice of S that guarantee $y^* \notin P(G')$, provide $x^* \notin P(G)$ as in the cases of interest for separation. The simplest such condition concerns the shrinking of $1 - arcs$ (i.e., arcs (i, j) with $x^*_{ij} = 1$), and requires $S = \{i, j\}$ for a certain node pair i, j with $x^*_{ij} = 1$.

It is known that $1 - edges$ cannot be shrunk for the $STSP$, instead. In this respect $ATSP$ behaves more nicely than $STSP$, in that the information associated with the orientation of the arcs allows for more powerful shrinking.

In our Branch-and-Cut $1 - arc$ shrinking is applied iteratively, so as to replace each path of $1 - arc$ by a single node. As a result of this pre-processing on x^*, all the nonzero variables are fractional. Notice that a similar result cannot be obtained for the symmetric TSP, where each $1 - edge$ chain can be replaced by a single $1 - edge$, but not by a single node.

5.5 PRICING BY REDUCED COSTS

The size of the linear program contains too many variables to be solve explicitly. It is clear that the total number of x variables is $|V|(|V| - 1)/2$ if the graph is undirected,

or $|V|(|V|-1)$ if it is directed. On the other hand, in the worst case, that is, if all products are available in the 50% of the markets, the number of z variables is of $|M||K|/2$. That means, that if we have a problem of size $|V| = 100$ and $|K| = 100$, the total number of variables for a undirected problem is 5050.

That is why we initialize the linear program with a small subset of variables and compute the optimal solution of that linear program. Afterwards, we check if the addition of a variable, which is not in the current linear program, might improve the LP-solution. According to the linear programming theory this can be done by the computation of the reduced cost of the variables. In a linear program of the form $\min\{c^T x : x \in \mathcal{P}\}$ a variable with positive reduced cost can improve the solution.

The *reduced cost* r_j of a non basic variable j with associated column $a_j \in \mathbb{R}^m$ and objective function coefficient c_j corresponding to a basic LP-solution with dual variable $y \in \mathbb{R}^m$ is defined as $r_j = c_j - y^T a_j$.

If no variables have positive reduced cost, then the current optimal solution also solves the original problem. In this case, it is said the variables does not price out correctly. The computation of the reduced cost is called *pricing*. If a variable does not price out correctly we add it to the linear program, re-optimize and iterate.

Essential for the practically efficient solution of the TPP with more than 100 markets is the application of sparse graph techniques. It has been observed by mean computational experience that many edges of the optimal solution are contained in the 5-nearest neighbour graph and almost all edges are contained in the 10-nearest neighbour graph. It also happens with the z variables, but the size of the neighbourhood is slightly bigger. Therefore, we initialize the variable set of the root node of the branch-and-bound tree with the k-nearest neighbour graph and augment it with the the edges belonging to the feasible solution from the initial heuristic.

5.6 BRANCHING STEP

According to the algorithms described in Fig. 5.1, 5.2 and 5.3, if both new cuts, and new variables are not generated then the branching phase is active by mean the flag status. There are many different strategies to achieve a splitting of the current subproblem in two or more new subproblems. For example,

- a fractional $0/1$ variable is set to 0 and 1,
- upper and lower bounds for integer variables are changed,
- dividing the polytope by hyperplanes,
- specific strategies for the problem.

Since we choose the first method, some variable is chosen as the *branching variable* and two new BC nodes, which are the two sons of the current BC node, are created and added to the set of active BC nodes. In the first son the branching variable is set to 1, and in the second one to 0. Different strategies about how to explore the branching

tree as well as what variable should be selected for splitting the current subproblem are described in the next two sections.

Node selection

Three different strategies to select the node to be processed next are discussed in this section.

1. Best First Search

   This strategy chooses a node with the worst dual bound, i.e., a node with lowest lower bound. The goal is to improve the dual bound. However, if this fails early in the solution process, the branch-and-bound tree tends to grow considerably resulting in large memory requirements.

2. Depth First Search

   This rule chooses the node that is *deepest* in the branch-and-bound tree, i.e., whose path to the root is longest. The advantages are that the tree tends to stay small, since always one of the two sons are processed next, if the node could not be fathomed. This fact implies that the linear programs from one node to the next are very similar, usually the difference is just the change of one variable bound and thus the re-optimization goes fast. The main disadvantage is that the dual bound basically stays untouched during the solution process resulting in bad solution guarantees.

3. Breadth First Search

   This strategy, in opposite to the previous, choose the node whose the path to the root is shortest.

Some computational experience has been carried out in order to select the best strategy for this particular problem. Finally, the best performance was offered by the Best First Search strategy.

Variable selection

There are a variety of different strategies for the selection of the branching variable, and some of them are enumerated in this section. Let w^*=(x^*,y^*,z^*) the fractional solution of the last solved LP, then only variables x^* and y^* will be taken in account.

1. Select a variable with value close to 0.5 that has a big objective function coefficient.

2. Select the variable that has an LP-value closest to 0.5.

3. Select the fractional variable that has maximum objective function coefficient.

4. If there were fractional variables that are equal to 1 in the currently best known feasible solution, select the one with maximum cost of them, otherwise, apply strategy 1.

5. Select a fractional variable that is closest to one.

6. Select a set $L \subset \mathcal{L}^V$ of promising branching variables candidates. Let $\mathcal{L}^C$ be the constraints system of the last solved LP. Solve for each variable $w_i \in L$ the two linear programs

$$v_0^i = \max\{c^T w | w \in \text{conv}\{\mathcal{L}^C\}, w_i = 0\}$$

$$v_1^i = \max\{c^T w | w \in \text{conv}\{\mathcal{L}^C\}, w_i = 1\}$$

and select the branching variable w_i with

$$\max\{v_0^i, v_1^i\} = \min_{w_i \in L} \max\{v_0^i, v_1^i\}.$$

Some running time can be saved if instead of the solution of the linear programs to optimality only a restricted number of iterations of the simplex-method is performed.

The last strategy, also known as *strong branching* has been chosen according to our experimental results. The reduction on the overall running time compensates widely for the extra running time spent in solving the LP.

5.7 COMPUTATIONAL RESULTS

The algorithm described in Section 3 was coded in C++ and run on a Pentium 500 MHz computer running Linux. ABACUS 2.2 linked with CPLEX 6.0 was used as a framework (see Jünger and Thienel [89] for details on this software).

We have considered the following four classes of test instances.

Class 1 contains 33-market symmetric instances defined with the same input data as in Singh and van Oudheusden [141]. These correspond to the largest instance size solved by these authors. The routing costs are those of a 33-vertex TSP described in Karg and Thompson [90] and do not satisfy the triangle inequality. The first vertex is the depot and all markets sell all products. Product prices are generated in [1,500] according to a discrete uniform distribution. We generated five instances with $|K|$=50, 100, 150, 200 and 250.

Class 2 instances are randomly generated by using the process described in Pearn and Chien [123]. Routing costs are randomly generated in $[1, \tau]$ where τ is generated in [15,140]. Each market sells a number of products randomly generated in $[1, m]$, where $m = |K|$ is the number of products. Purchase costs are randomly generated in $[0, \omega]$ where ω is generated in [5,75] for each market. Contrary to Pearn and Chien, we have used symmetric routing cost instead of asymmetric costs. We defined instances with $|V|$=50, 100, 150, 200 and 250, and $|K|$=50, 100, 150, 200 and 250.

Class 3 instances were defined by first generating $|V|$ integer coordinate vertices in the $[0, 1000] \times [0, 1000]$ square according to a uniform distribution and defining routing costs by Euclidean distances. Each product p_k was associated with $|M_k|$ randomly selected markets, where $|M_k|$ was randomly generated in $[1, |V|-1]$. The remaining characteristics of these instances are defined as for Class 1.

Class 4 instances were generated in the same manner as Class 3 instances, with a limit on supplies. For each product p_k and each market v_i, q_{ki} was randomly generated in [1,15] and $d_k := \lceil \lambda \max_{v_i \in M_k} q_{ki} + (1-\lambda) \sum_{v_i \in M_k} q_{ki} \rceil$ for λ=0.5, 0.7, 0.9 and 0.99.

To analyze the performance of the algorithm, we have used five instances for each value of $|V|$ and $|K|$ in $\{50, 100, 150, 200\}$. Computational results are summarized in Tables 5.1 to 5.4. The column headings are defined as follows:

$|V|$**:** number of vertices (including the depot);

$|K|$**:** number of products;

Visit: number of markets visited in the optimal solution;

(3.15): number of generated constraints of type (3.15);

(3.3): number of generated constraints of type (3.3);

(3.16): number of generated constraints of type (3.16);

(3.12): number of generated constraints of type (3.12);

LB%: percentage ratio $\underline{w}$/optimum, where $\underline{w}$ is the value of the last LP solved at the root node;

UB%: percentage ratio $\overline{w}$/optimum, where $\overline{w}$ is the value of the heuristic solution value computed at the root node;

Root sec: computing time for solving the root node;

Total sec: total computing time spent by the branch-and-cut code;

Nodes: number of nodes generated (1 means that the problem required no branching).

Computational results relative with the TPP with unlimited supplies (Tables 5.1 to 5.3) indicate that the algorithm can successfully solve to optimality instances involving up to 200 vertices and 200 products within short computing time. This compares favourably with the best known exact results obtained by Singh and van Oudheusden [141] whose largest instances contained at most 20 vertices and at most 50 products (see Table 5.1). Note that constraints (3.15) are not used for these instances since each market sells all products. Results presented in Tables 5.2 and 5.3 indicate that the proportion of markets present in the optimal solution typically varies between 20% and 40%, which means that the instances are not uniquely driven by routing costs

and market selection decisions play an important role, thus increasing the difficulty of the problem. Instances from Class 2, where travel costs are uniformly distributed, are much easier to solve than instances from Class 3, which work with a Euclidean travel cost structure. This is consistent with what is observed for the TSP (Balas and Toth [15]). Problem difficulty increases with $|V|$ and $|K|$. Both the heuristic upper bound and the lower bound at the root node the search tree tend to be within a small percentage of the optimum, typically less than 5%, and their quality is more closely related to the number of markets than to the number of products. As expected, the computational time and the number of branch-and-cut nodes increase strongly with $|V|$. While performing the tests, we have observed that spending more time executing the pricing heuristic can help reducing significantly the time required by the LP solver.

Table 5.4 contains computational results for instances with limited product availability at some markets. To our knowledge, we are the first to address this more difficult variant of the classical TPP. For this class of instances, problem difficulty is clearly related to the size of λ which effectively controls the percentage of markets in the solution. When λ is small, most markets are visited and the TPP becomes very close to a TSP and is thus relatively easy to solve for the number of markets considered in our experiments. As λ grows both the upper bound and the lower bound at the root node deteriorate, which translate again into more branching and larger computation times. However, we have observed that problems generated with $\lambda > 0.95$ tend to be easier than those generated with $\lambda \leq 0.95$ since the problem is then almost identical to the unlimited supply case as λ approaches 1. Table 5.4 shows results for only four values of λ. Overall, we were able to solve instances involving up to 200 vertices and 200 products.

On average, the purchasing cost and of routing cost had similar proportions on the optimal solution values of the above described instances. To measure the difficulty of the problem comparing the routing and the purchasing costs, we have conducted some further experiments. In particular, we have generated and solved instances of Class 1 with product prices in [1,50], [1,500], [1,5000] and [1,50000], of Class 2 with $|V| = 100$ and ω in [2,30], [5,75], [20,300] and [50,750], and of Class 3 with $|V| = 100$ with product prices in [1,50], [1,500], [1,5000] and [1,50000]. Table 5.9 shows the average results of the five instances for each value of $|K|$. For each group of instances, four features are given:

Vis.: the number of visited markets in an optimal solution;

Nod.: the number of explored branching nodes;

PC: the percentage of the pricing cost over the total cost of an optimal solution;

Sec.: the total time of solving an instance.

As it is observed from the table, the difficulty of solving an instance is not strongly affected by the magnitude of the pricing costs compared to the routing costs. Solving instances from Classes 1 and 2, a problem is easier when the pricing costs are bigger

than the routing costs, even if an optimal decision involves selecting more markets. The difficulty of the problem seems to be more related with (say) $|M|$ and $|K|$.

An overview to the three unlimited-supply TPP classes shows that the proposed algorithm works better on the instances of Class 2, where the routing costs are randomly generated. This is due to better lower bounds on instances of Class 2 when compared with instances of Class 3, where more efforts (i.e., cuts) are required by the separation procedures to approximate the lower bound to 99% over the optimal solution when $|V| = 200$. The heuristic produced worse results when the routing costs are non-Euclidean, but the branch-and-cut algorithm was less sensitive to this drawback. Indeed, the algorithm executes more branching on instances of Classes 1 and 2 where the upper bound is worse, but the total computational time is smaller. Because of this observation, we did not implemented a more sophisticated heuristic.

The harder instances of our benchmarks correspond to Class 4, due to the limited supply constraints. Indeed, when $|V| = 200$, it was only possible to solve six and seven instances over the twenty trials for $|K| = 150$ and $|K| = 200$, respectively. When the execution was aborted due to the 3 hours of time limit, the average gap between the final upper and lower bounds was 2.7% and 8.8% when $|K| = 150$ and $|K| = 200$, respectively, while at the end of the root node, the average gap between lower and upper bounds was 7.5% and 21.5% when $|K| = 150$ and $|K| = 200$, respectively. We guest that these big values are mainly due to the lower bound quality and not to the primal heuristic procedure.

We have formulated, analyzed and solved the undirected TPP. Two versions were considered: the classical one where products are available in unlimited supply in the markets, and a new more difficult version where upper bounds are imposed on supplies. Facet-defining inequalities applicable to both versions were proposed, and a branch-and-cut algorithm encompassing a heuristic, a pricing mechanism as well as several separation procedures were developed. Extensive computational results on four instance classes indicate that for the classical TPP our algorithm outperforms by far all previously available methods. For both versions of the TPP, it can solve instances involving up to 200 markets and 200 products.

Table 5.1 Average results over 5 random instances of Class 1 with $|V|$=33.

$\|K\|$	*visit*	(3.15)	(3.3)	(3.16)	(3.12)	$LB\%$	$UB\%$	*Root sec*	*Total sec*	*Nodes*
50	8.4	0.0	6.2	173.2	1.4	99.468	106.312	0.4	1.6	3.0
100	10.4	0.0	43.8	1164.8	5.0	98.535	104.150	2.4	8.6	8.2
150	13.2	0.0	58.6	2245.2	6.6	98.491	104.760	4.6	17.8	10.6
200	13.8	0.0	133.2	4917.0	22.2	97.798	108.567	6.2	49.2	25.4
250	15.2	0.0	144.2	5879.6	17.2	97.541	105.478	6.4	63.0	31.0

Table 5.2 Average results over 5 random instances of Class 2.

$\|V\|$	$\|K\|$	*visit*	(3.15)	(3.3)	(3.16)	(3.12)	$LB\%$	$UB\%$	*Root sec*	*Total sec*	*Nodes*
50	50	25.8	0.6	34.0	105.4	14.0	99.791	103.545	0.8	2.6	4.6
	100	37.2	18.0	62.8	173.0	7.8	100.000	100.309	1.2	2.6	3.8
	150	43.0	1.2	24.0	76.6	0.2	100.000	100.000	0.2	1.2	1.0
	200	44.8	49.5	72.2	357.5	8.0	100.000	100.155	1.8	3.8	3.5
100	50	34.4	18.6	370.8	888.8	48.0	99.663	104.890	5.6	17.2	7.8
	100	52.0	18.0	62.2	155.0	15.4	100.000	102.059	3.0	7.0	3.4
	150	71.0	0.0	6.0	15.4	4.8	100.000	100.362	2.6	6.8	3.4
	200	76.4	44.0	178.8	468.6	12.8	100.000	100.139	5.4	12.4	5.0
150	50	38.2	23.2	87.0	385.8	33.6	99.017	123.587	8.8	33.2	10.2
	100	62.5	186.8	401.2	3001.2	100.5	99.456	108.707	17.8	147.8	32.0
	150	77.2	355.8	615.2	2454.2	111.0	99.845	101.451	12.0	163.8	37.4
	200	91.6	6.8	158.6	392.8	32.6	99.961	100.773	8.6	25.0	6.6
200	50	33.6	2.0	469.4	2804.8	114.2	98.494	153.614	10.8	222.2	31.0
	100	65.4	6.8	327.6	760.8	81.4	99.752	115.347	12.2	105.8	32.6
	150	92.6	17.2	267.4	611.0	91.0	99.857	104.444	16.6	141.6	32.2
	200	106.0	9.6	97.4	222.2	50.8	100.000	100.448	13.6	59.2	11.8

Table 5.3 Average results over 5 random instances of Class 3.

$\|V\|$	$\|K\|$	*visit*	(3.15)	(3.3)	(3.16)	(3.12)	$LB\%$	$UB\%$	*Root sec*	*Total sec*	*Nodes*
50	50	7.6	321.2	234.2	1027.4	42.6	100.000	100.000	6.2	6.4	1.0
	100	13.0	731.4	362.4	2050.4	113.6	99.959	102.549	20.2	21.6	1.4
	150	15.0	948.6	353.2	3319.4	107.8	99.938	102.286	28.4	32.2	2.2
	200	16.6	1245.2	345.8	3704.6	95.6	99.766	101.372	33.2	35.4	1.8
100	50	10.4	1052.0	1685.0	2730.0	479.6	99.988	100.000	120.0	122.4	1.4
	100	14.2	1844.0	1800.4	5929.8	537.0	99.914	100.029	293.8	309.4	2.2
	150	18.0	2895.8	1921.0	8659.4	566.8	99.494	103.706	346.4	423.0	3.0
	200	19.8	2600.0	1545.0	9117.8	528.2	99.511	105.514	280.8	344.4	6.6
150	50	10.6	1719.0	4988.0	4793.2	1454.2	99.986	100.000	953.0	957.0	1.4
	100	15.4	3340.0	4323.2	8442.4	1302.0	99.700	105.814	1795.6	1918.0	3.8
	150	20.2	4863.6	4132.0	11120.8	1244.6	99.805	111.154	1602.2	1936.6	5.0
	200	22.0	6530.0	3679.4	14500.0	1076.0	99.528	110.873	1578.0	2126.4	8.2
200	50	10.0	1238.8	4423.2	3032.0	1235.0	97.722	108.923	842.0	1125.6	2.6
	100	18.6	6780.4	9460.6	13270.8	2488.4	98.478	109.866	3401.6	4897.8	5.0
	150	21.8	12843.6	12288.8	30708.0	3558.6	95.834	109.389	4455.4	9933.4	14.6
	200	22.2	14015.8	9612.0	29578.8	2780.0	90.449	110.013	5630.0	9198.2	7.8

Table 5.4 Average results over 5 random instances of Class 4 with $|V|$=50.

$\|K\|$	λ	*visit*	(3.15)	(3.3)	(3.16)	(3.12)	$LB\%$	$UB\%$	*Root sec*	*Total sec*	*Nodes*
50	0.50	46.2	31.0	161.4	193.0	13.4	99.994	100.021	1.4	2.8	1.8
	0.70	40.6	115.4	436.4	425.4	115.6	99.649	100.324	3.8	10.6	17.4
	0.90	22.0	724.6	1281.0	2522.8	408.0	97.631	101.675	9.0	50.4	48.6
	0.99	11.6	369.6	349.8	1008.4	122.0	99.936	100.000	9.2	10.2	3.4
100	0.50	50.0	48.0	257.8	588.0	37.4	99.966	100.000	1.6	4.8	5.4
	0.70	46.0	493.4	774.8	1597.6	156.4	99.799	100.096	3.4	16.6	25.0
	0.90	30.4	739.0	864.2	2888.0	277.8	98.170	101.650	8.0	46.6	37.8
	0.99	15.0	499.4	309.6	1523.4	98.0	99.991	100.000	11.6	12.4	1.4
150	0.50	50.0	82.4	206.2	767.2	39.2	99.976	100.000	2.0	5.6	6.6
	0.70	48.4	88.2	308.0	1049.8	41.8	99.937	100.009	2.6	7.4	9.0
	0.90	35.0	844.0	949.4	3794.0	298.0	97.870	101.544	9.0	66.0	55.8
	0.99	15.2	972.0	537.8	4184.0	150.6	99.877	100.029	18.8	21.4	3.0
200	0.50	50.0	98.6	222.2	1114.8	38.0	99.982	100.000	2.8	7.2	5.4
	0.70	50.0	96.8	163.4	793.4	30.8	99.925	100.000	2.6	6.0	3.8
	0.90	38.2	972.0	952.0	4514.0	334.8	98.382	100.269	8.8	80.4	53.8
	0.99	16.6	608.4	418.6	4290.6	126.0	99.728	100.244	17.8	20.8	3.4

Table 5.5 Average results over 5 random instances of Class 4 with $|V|$=100.

$\|K\|$	λ	*visit*	(3.15)	(3.3)	(3.16)	(3.12)	$LB\%$	$UB\%$	*Root sec*	*Total sec*	*Nodes*
50	0.50	100.0	37.0	593.8	316.2	89.8	99.976	100.000	10.8	16.2	3.4
	0.70	87.8	89.8	834.0	394.4	139.2	99.939	100.069	17.4	24.0	5.0
	0.90	40.4	3719.2	8760.6	11818.8	2411.4	98.402	100.729	57.0	409.8	94.2
	0.99	12.6	686.2	1460.0	2245.2	445.2	99.042	100.027	68.8	98.0	7.0
100	0.50	100.0	57.2	536.2	525.2	79.6	99.988	100.001	11.2	17.4	4.2
	0.70	91.0	221.0	1414.4	1471.6	200.6	99.979	100.004	25.4	32.8	3.8
	0.90	57.8	6860.8	20450.2	32340.0	5621.8	98.233	102.915	49.8	906.4	166.6
	0.99	16.8	1431.4	1742.4	5872.0	515.4	99.367	100.054	164.2	225.6	9.4
150	0.50	97.8	187.8	812.2	1264.2	111.4	99.994	100.004	19.0	24.8	3.4
	0.70	99.8	136.8	720.0	981.2	163.2	99.969	100.001	16.8	32.2	10.6
	0.90	64.2	16607.4	33279.2	80920.8	7810.8	98.719	101.303	65.2	1651.8	196.2
	0.99	19.2	3551.6	2978.2	16949.0	817.2	98.273	100.675	219.8	688.2	27.4
200	0.50	100.0	131.0	596.8	1391.2	90.4	99.994	100.000	16.0	22.8	3.4
	0.70	100.0	134.0	523.8	1146.4	81.2	99.984	100.000	16.6	24.0	3.8
	0.90	72.4	7147.8	20553.4	39334.6	5170.6	98.882	101.453	64.6	1315.2	234.6
	0.99	24.6	23831.4	9501.8	80025.0	2892.2	98.292	100.862	312.6	1900.2	57.0

Table 5.6 Average results over 5 random instances of Class 4 with $|V|$=150.

$\|K\|$	λ	*visit*	(3.15)	(3.3)	(3.16)	(3.12)	$LB\%$	$UB\%$	*Root sec*	*Total sec*	*Nodes*
50	0.50	148.6	59.0	5332.4	1459.6	1955.4	99.924	100.027	20.4	204.2	91.4
	0.70	132.8	414.8	9517.0	2465.6	1812.4	99.911	100.162	62.2	275.6	81.4
	0.90	62.8	18174.0	69428.4	62122.6	22382.8	98.909	102.216	216.8	4721.8	185.4
	0.99	14.6	1473.0	4727.2	4266.0	1071.8	99.084	101.151	433.0	610.8	13.0
100	0.50	149.8	582.8	8600.8	4465.4	2040.2	99.958	100.008	26.6	288.2	86.6
	0.70	148.0	341.4	10684.0	5054.6	3245.8	99.893	100.037	36.8	401.2	127.0
	0.90	88.6	43316.2	167347.4	177371.0	46220.6	99.109	101.829	194.0	8334.6	418.2
	0.99	22.0	10440.6	12072.8	49225.2	3078.4	97.745	100.534	921.4	4712.6	41.0
150	0.50	149.0	796.4	19342.6	12959.2	5218.2	99.974	100.015	32.8	914.8	208.6
	0.70	148.6	360.4	6565.2	5096.4	2229.4	99.925	100.005	36.4	370.6	109.8
	0.90	101.8	27404.6	93793.0	104034.4	26977.8	99.290	100.990	179.6	4877.4	238.2
	0.99	26.2	26033.2	23285.2	109225.4	6444.8	96.918	102.375	1171.2	8564.0	84.2
200	0.50	150.0	436.4	5409.4	6151.0	1575.4	99.977	100.001	36.8	345.8	90.6
	0.70	150.0	635.8	7702.2	7570.8	2450.6	99.936	100.017	40.0	421.4	96.6
	0.90	117.0	29086.0	92983.4	120420.8	21252.0	99.502	100.872	158.0	5907.2	329.4
	0.99	30.6	28492.8	28961.8	143598.8	8131.8	96.444	101.582	876.0	8125.4	63.8

Table 5.7 Average results over 5 random instances of Class 4 with $|V|$=200.

$\|K\|$	λ	*visit*	(3.15)	(3.3)	(3.16)	(3.12)	$LB\%$	$UB\%$	*Root sec*	*Total sec*	*Nodes*
50	0.50	197.4	2667.4	139468.6	26936.2	27509.8	99.931	100.040	84.0	4804.2	608.6
	0.70	187.8	3640.4	187580.2	39422.2	28992.0	99.650	100.196	72.8	5898.6	697.0
	0.90	87.8	4224.0	34169.0	17022.4	9356.4	99.452	100.505	377.4	2164.4	99.4
	0.99	20.8	7200.8	22881.4	22076.0	4199.2	98.627	101.280	2363.0	6253.0	51.8
100	0.50	200.0	272.2	16759.2	5036.0	4208.5	99.968	100.002	53.5	745.8	112.0
	0.70	197.0	774.0	37428.3	11764.7	9154.3	99.894	100.036	70.0	1578.7	249.7
	0.90	121.8	33020.4	174537.8	140296.6	44897.6	99.095	102.024	447.6	11925.6	297.4
	0.99	28.2	18219.4	24826.8	70204.8	6359.2	93.539	107.540	1435.2	11872.4	67.0

Table 5.8 Statistics using different pricing and routing costs

Class 1: $|V| = 33$

	50				500				5000				50000			
$\|K\|$	*Vis.*	*Nod.*	*PC*	*Sec.*	*Vis.*	*Nod.*	*PC*	*Sec.*	*Vis.*	*Nod.*	*PC*	*Sec.*	*Vis.*	*Nod.*	*PC*	*Sec.*
50	3.6	1.0	33.1	0.0	8.4	3.0	48.8	1.6	16.8	4.2	60.2	1.8	25.2	2.2	84.8	0.6
100	4.4	1.4	41.1	4.8	10.4	8.2	52.2	8.6	22.0	2.6	67.8	1.2	29.2	1.0	89.6	0.0
150	5.8	2.6	40.7	11.4	13.2	10.6	49.8	17.8	24.4	1.0	72.4	0.0	32.4	1.0	91.3	0.0
200	6.4	1.8	43.3	17.4	13.8	25.4	54.5	49.2	26.6	1.0	74.2	0.0	32.2	1.8	93.5	0.0
250	7.0	2.6	44.8	27.0	15.2	31.0	55.4	63.0	27.2	1.0	77.2	0.0	33.0	1.0	94.6	0.0

Class 2: $|V| = 100$

	$[2, 30]$				$[5, 75]$				$[20, 300]$				$[50, 750]$			
$\|K\|$	*Vis.*	*Nod.*	*PC*	*Sec.*	*Vis.*	*Nod.*	*PC*	*Sec.*	*Vis.*	*Nod.*	*PC*	*Sec.*	*Vis.*	*Nod.*	*PC*	*Sec.*
50	13.6	12.6	18.5	56.4	34.4	7.8	44.4	17.2	45.6	11.4	60.8	3.6	49.0	1.8	71.8	0.0
100	24.2	43.8	21.1	168.0	52.0	3.4	59.2	7.0	66.8	5.4	76.2	1.8	66.2	6.6	86.7	0.6
150	35.6	59.0	29.5	196.2	71.0	3.4	71.5	6.8	79.2	6.2	85.4	3.6	77.2	4.2	92.5	1.2
200	42.4	42.6	32.0	141.0	76.4	5.0	80.0	12.4	83.6	12.6	90.5	9.0	81.8	2.6	97.4	1.2

Table 5.9 Statistics using different pricing and routing costs (cont.)

Class 3: $|V| = 100$

	50				500				5000				50000			
$\|K\|$	*Vis*	*Nod*	*PC*	*Sec*	*Vis*	*Nod*	*PC*	*Sec*	*Vis*	*Nod*	*PC*	*Sec*	*Vis*	*Nod*	*PC*	*Sec*
50	9.2	19.8	9.4	104.4	10.4	1.4	43.1	122.4	35.0	8.6	74.5	81.0	41.8	21.0	94.8	66.6
100	13.0	14.2	10.7	179.8	14.2	2.2	50.0	309.4	48.8	3.0	81.5	34.8	61.2	140.2	97.1	115.2
150	17.8	193.0	12.6	1552.8	18.0	3.0	55.3	423.0	60.6	11.8	85.4	69.0	63.4	185.0	98.0	638.4
200	19.0	11.8	15.0	256.4	19.8	6.6	56.9	344.4	77.2	42.2	88.2	154.2	77.0	481.4	98.3	2557.8

6

A Branch-and-Cut Algorithm for the ATPP

A Branch-and-Cut approach for the Asymmetric Traveling Purchaser Problem (ATPP) is performed in this chapter. The general scheme described in the previous chapter is also followed, but taking into account the model for the asymmetric case. It also proposes a transformation of the ATPP into its symmetric version, so a second exact method is also presented. An extensive computational analysis on several classes of instances from literature evaluates the proposed approaches. A previous work by Singh and Oudheusden published in 1999 solves instances with up to 25 markets and 100 products, while the here-presented approaches prove optimality on instances with up to 200 markets and 200 products.

Since the Branch-and-Cut skeleton is quite similar to the one described in Chapter 5, only those specific issues of this problem are included in this chapter. The first section is devoted to a specific heuristics involved in this approach. Section 6.2 sketches the above mentioned transformation. Section 6.3 describes the separation algorithms, and finally, our computational experience is shown in Section 6.4. This experience compares the classical implementation of the Branch-and-Cut, the improved implementation by the branching heuristic and transformation approach.

6.1 B&C BASED HEURISTIC

The hardness of solving ATPP instances motivates the develop of a heuristic approach based on the branch-and-cut algorithm. According to the procedure introduced by Fischetti, Lodi and Toth [52], a reasonable branching scheme (called *local branching*) considers the incumbent feasible ATPP circuit $A^* \subseteq A$ at the end of each node of the

decision tree, and proceed by creating two new nodes with the disjunction:

$$\sum_{a \in A^*} (1 - x_a) \leq k \qquad \text{or} \qquad \sum_{a \in A^*} (1 - x_a) \geq k + 1, \tag{6.1}$$

where k is an small integer number. Notice that $\sum_{a \in A^*}(1 - x_a)$ is the number of arcs in the solution x and not in the heuristic circuit A^*. Clearly, solving the left-hand side node is similar to explore a k-neighbourhood of the ATPP solution represented by A^*. This exploration could only be done by full enumeration when $k \leq 3$, while the branch-and-cut code could succeed for some larger values of the parameter k, even if the ideal situation of solving the node when $k = n$ (i.e., solving the original ATPP instance to optimality) is unlikely. The right-hand side node is more difficult to be solved, and indeed it is as difficult as the original problem when k is an small value.

Based on the above consideration we have modified the branch-and-cut code to produce a good solutions for large ATPP instances. The mechanism consists in solving the root node by using the algorithm described in Chapter 5. Let $A^* \subseteq A$ the arcs representing the best feasible ATPP solution provided by the heuristic routines in the code (i.e., the initial and primal heuristics, both ensuring a 3-optimality on the circuit). To look for a better ATPP solution, we add the constraint

$$k_1 \leq \sum_{a \in A^*} (1 - x_a) \leq k_2,$$

where $k_1 := 3$ and $k_2 := 6$ where choosing after some computational experiments. This constraint addresses the branch-and-cut algorithm to explore outside a 3-neighbourhood (which has been explored by the heuristic) and inside a 6-neighbourhood (which is probably a limited region that can be explored by the branch-and-cut).

Based on our computational experiments, solving the new problem requires many computational effort, hence we also decided to fix some variables. In particular, when a new node contains only the variables with a fractional value on the current fractional solution, plus all the variables that have been useful in a previous ATPP solution.

The heuristic approach stops when a better ATPP solution is not found when solving the current node, or after a given time limit.

6.2 TRANSFORMATION OF THE ATPP INTO THE STPP

We now introduce an alternative proposal to find an optimal of an ATPP instance when an exact algorithm for solving symmetric instances is available. The aim is to transform an asymmetric instance into a symmetric one in the spirit of similar works done for similar routing problems. Inspired by the *3-node transformation* of Karp [91] and the *2-node transformation* of Jonker and Volgenant [86], both for the ATSP, we next propose a transformation for the ATPP into the STPP.

We propose a simple *2-node* transformation for the TPP (see Fig. 6.1 for details). A complete undirected graph $G'(V', E)$ with $2|V|$ vertices is built from the original directed one $G(V, A)$ as follows. For each vertex $v_i \in V$ a new vertex, v_{n+i} is added

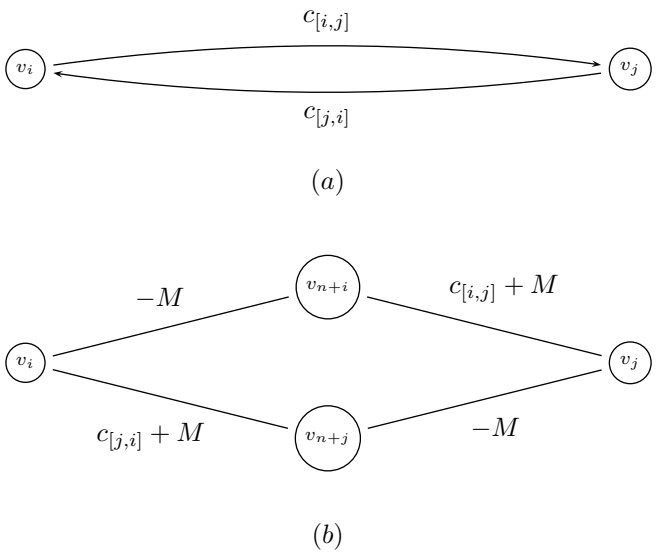


Fig. 6.1 Transformation of the ATPP into the STPP.

to V', that is $V' = V \cup \bigcup_{v_i \in V} \{v_{i+|V|}\}$. In addition, for each pair of vertices v_i and v_j , the cost of both edges $[v_i, v_{n+i}]$ and $[v_j, v_{n+j}]$ are set to $-M$, and the cost of the edges $[v_{n+i}, v_j]$ and $[v_{n+j}, v_i]$ are set to $c_{ji} + M$ and $c_{ij} + M$ respectively. It is easy show that an optimal solution of the ATPP in G induces an optimal solution for the STPP in G' and *vice versa*. Then, a minimum-cost cycle in G' solving the STPP corresponds to a minimum-cost circuit in G solving the ATPP, and vice versa. Indeed, an optimal cycle cannot use two consecutive edges with negative cost, so it will alternate positive and negative cost, so the value of M will not affect the total cost.

There are in literature transformations for the ATSP into its symmetric version. Nevertheless, these transformations are not useful when the TPP is approached. Two of the most important transformations are described below.

The *3-node* Transformation for the TSP

This transformation, called the *3-node* transformation was proposed by Karp [91]. A complete undirected graph with $3n$ vertices is obtained from the original complete directed one by adding two copies, v_{n+i} and v_{2n+i}, of each vertex $v_i \in V$, and by (i) setting to 0 the cost of the edges $e_{[i,n+i]}$ and $e_{[n+i,2n+i]}$ for each $v_i \in V$, (ii) setting to c_{ij} the cost of edge $e_{[2n+i,j]}$ for all $v_i, v_j \in V$, and (iii) setting to $+\infty$ the cost of all remaining edges.

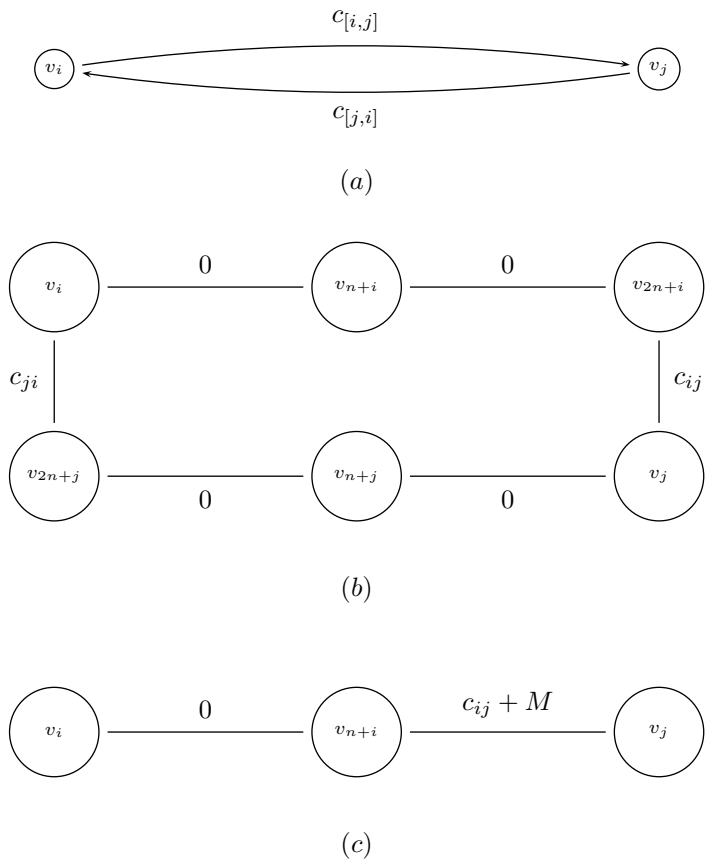


Fig. 6.2 Transformation of the ATSP into the STSP.

The *2-node* Transformation for the TSP

The *2-node* transformation was proposed by Jonker and Volgenant [86] (see also Jünger, Reinelt and Rinaldi [87]). A complete undirected graph with $2n$ vertices is obtained from the original complete directed one by adding a copy, v_{n+i}, of each vertex $v_i \in V$, and by (i) setting to 0 the cost of the edge $e_{[v_i,v_{n+i}]}$ for each vertex $v_i \in V$, (ii) setting to $c_{ij} + M$ the cost of the edge $e_{[n+i,j]}$ for all $v_i, v_j \in V$, where M is a sufficiently large positive value, and (iii) setting to $+\infty$ the cost of all the remaining edges. The transformation value nM has to be subtracted from the STSP optimal cost.

Figure 2.3 illustrates these two last transformations.

6.3 SEPARATION PROCEDURES

Separations algorithms for our implementation of the branch-and-cut algorithm for the ATPP are described in this section.

Separation of YSEC$^+$ Inequalities

Proposition 6.1. *Given a market v_i with $y_i^* > 0$, a most violated YSEC$^+$ constraint (3.20) corresponds to a minimum-capacity directed cut (min-cut) $(S, V^* \setminus S)$ with $v_i \in S$ and $v_0 \notin S$ in the support graph G^* by imposing a capacity x_a^* on each arc $a \in A^*$.*

This separation algorithm is quite similar to the algorithm described for the symmetric case. Instead, a YSEC$^+$ is violated if the value of the directed cut is less than 1 (See algorithm in Fig. 6.3 for details). Notice that similarly to the symmetric case, constraints YSEC$^+$ are introduced in the same short form, that is (5.1). However it derives from the following equation obtained adding (3.19) for each vertex v_i of a violated set S.

$$\sum_{a \in A(S)} x_a + \sum_{a \in \delta^+(S)} x_a = \sum_{v_i \in S} y_i. \tag{6.2}$$

Replacing (6.2) in (3.20) we obtain (5.1).

Figure 6.4 shows a fractional point violating the following (3.20).

$$\sum_{a \in \delta^+(S)} x_a \geq y_9$$

With $S := \{v_4, v_9\}$, since $\sum_{a \in \delta^+(S)} x_a = 0.5$ and $y_9 = 1$, and then $0.5 \ngeq y_9$.

Input: (G^*, x^*, y^*), $\mathcal{L}^C$
Output: $\mathcal{L}^C$
COMPONENTS($G^*, \mathcal{C}$)
for all $S \in \mathcal{C}$
  if ($v_0 \notin S$)
    for all $v_i \in S$
      $\mathcal{L}^C := \mathcal{L}^C \cup \sum_{a \in \delta^+(S)} x_a \geq y_i$
  else
    for all $v_i \in S \setminus \{v_0\}$
      cut-value := DIRECTED MINCUT($G^*, v_0, v_i,$ S');
      if (cut-value $< y_i$)
        $\mathcal{L}^C := \mathcal{L}^C \cup \sum_{a \in A(S)} x_a - \sum_{v_j \in S \setminus \{v_i\}} y_j \leq 0$

Fig. 6.3 Separation of the YSEC$^+$ inequalities.

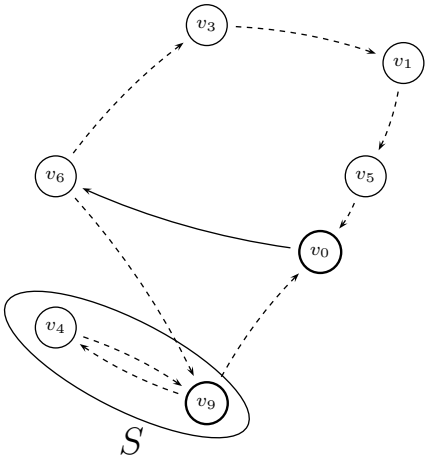


Fig. 6.4 Fractional point violating a YSEC$^+$ inequality.

Separation of ZSEC$^+$ Inequalities

Proposition 6.2. *Given a product p_k, determining a most violated ZSEC$^+$ constraint (3.36) is equivalent to finding a subset S with minimum value of*

$$\sum_{a\in\delta^+(S)} x_a^* + \sum_{v_i\in M_k\setminus S} \frac{z_{ki}^*}{d_k}.$$

Figure 6.6 shows a fractional solution violating (3.36). In this particular case $0.5/d_4$ and $0.5/d_4$ units of the product p_4 are offered at markets v_2 and v_4 respectively. The subset S' consist of the vertices $S' := \{\hat{v}, v_2, v_4\}$ and according to the Figure 6.6 $\sum_{a\in\delta(S'\setminus\hat{v})} x_a^* = 0.5$ and $\sum_{v_i\in M_4} z_{i4}/d_4 = 1$. Therefore inequality

$$\sum_{a\in\delta^+(S)} x_a \geq \frac{1}{d_k} \sum_{v_i\in S\cap M_4} z_{4i}, \qquad\qquad S := \{v_2, v_4\}$$

is violated.

Separation of the D_l^+ and D_l^- Inequalities

The separation problem for the class of D_l^+ inequalities calls for a vertex sequence $(i_1,\ldots,i_l)$, $1 \leq l \leq n-1$, for which the degree of violation

$$\begin{aligned}\phi(i_1,\ldots,i_l) := x_{i_1 i_l}^* + \textstyle\sum_{h=2}^{l} x_{i_h,i_{h-1}}^* + 2\sum_{h=2}^{l-1} x_{i_1 i_h}^* + \\ \textstyle\sum_{h=3}^{l-1} x^*(\{i_2,\ldots,i_{h-1}\},i_h) - \sum_{h=1}^{l} y_{i_h}^* + y_{i_l}^*\end{aligned} \tag{6.3}$$

Input: (G^*, x^*, y^*, z^*), $\mathcal{L}^C$ ***Output:*** $\mathcal{L}^C$

for all $p_k \in K$
  $\hat{V} := V^* \cup \hat{v}$
  for all $v_i \in M_k$
    $\hat{A} := A^* \cup a = [\hat{v}, v_i]$
    $x^*_a = 2\frac{z^*_{ik}}{d_k}$
  $\hat{G} = (\hat{V}, \hat{A})$
  `cut-value := MINCUT(`$\hat{G}$`,`v_0`,`$\hat{v}$`, `S' `);`
  if `( cut-value` < 1`)`
    $S := S' \setminus \{\hat{v}\}$
    $\mathcal{L}^C := \mathcal{L}^C \cup \sum_{a \in \delta(S)} x^*_a \geq 2 \sum_{v_i \in M_k \cap S} \frac{z_{ki}}{d_k}$

Fig. 6.5 Separation of the ZSEC$^+$ inequalities.

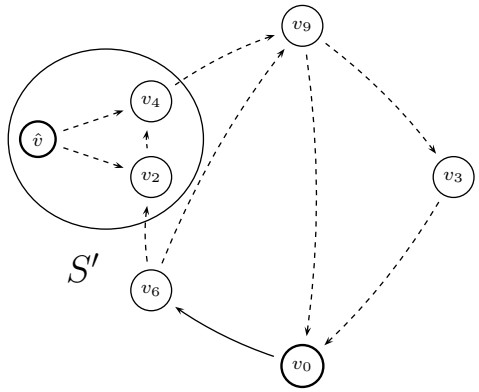


Fig. 6.6 Fractional point violating a ZSEC$^+$ inequality.

is as large as possible. This is itself a combinatorial optimization problem and can be solved by the following implicit enumeration scheme. The scheme starts with an empty node sequence. Then, iteratively, we extend the current sequence in any possible way and evaluate the degree of violation of the corresponding D_l^+ inequality. The process can be seen by mean of a branch-decision tree. The root node of the tree represent the empty sequence. Each node at level l $(1 \leq l \leq n-1)$ correspond to a sequence of the type $(i_1, \ldots, i_l)$; when $l \leq n-1$, each such node generates $n - l$ descending nodes, one for each possible extended sequence $(i_1, \ldots, i_l, i_{l+1})$. Exhaustive enumeration of all nodes of the tree is clearly impractical, even for small values of n. However, a very large number of nodes can be fathomed by mean the following simple upper bound computation. Let $(i_1, \ldots, i_l)$ be the sequence associated with the current branching node, say ν, and let ϕ_{max} denote the maximum degree if violation so far found during the enumeration. Consider any potential descendent node of ν, associated with a sequence of the type $(i_1, \ldots, i_l, i_{l+1}, \ldots, i_m)$. Then, directly from the definition (6.3) one has

$$\phi(i_1,\ldots,i_l,i_{l+1},\ldots,i_m) := x^*_{i_1 i_m} + \sum_{h=2}^{m} x^*_{i_h,i_{h-1}} + \sum_{h=2}^{m-1} x^*_{i_1 i_h} +$$

$$\sum_{h=2}^{m-1} x^*(\{i_1,\ldots,i_{h-1}\},i_h) - \sum_{h=1}^{m} y^*_{i_h} + y^*_{i_m} =$$

$$x^*_{i_1 i_m} + \sum_{h=2}^{l} x^*_{i_h,i_{h-1}} + \sum_{h=l+1}^{m} x^*_{i_h,i_{h-1}} +$$

$$\sum_{h=2}^{l} x^*(\{i_1,\ldots,i_{h-1}\},i_h) + \sum_{h=l+1}^{m-1} x^*(\{i_1,\ldots,i_{h-1}\},i_h) +$$

$$\sum_{h=2}^{m-1} x^*_{i_1 i_h} - \sum_{h=1}^{m-1} y^*_{i_h}$$

we establish the following bound

$$\phi(i_1,\ldots,i_l,i_{l+1},\ldots,i_m) \le \pi(i_1,\ldots,i_l) + x^*_{i_{l+1} i_l} +$$

$$\left[x^*(\delta^+(i_1)) - y^*_{i_1}\right] + \sum_{h=l+1}^{m-1} \left[x^*(\delta^-(i_h)) - y^*_{i_h}\right]$$

and then from (3.18)–(3.19) we have

$$\pi(i_1,\ldots,i_l) + x^*_{i_{l+1} i_l} \tag{6.4}$$

where has been defined

$$\pi(i_1,\ldots,i_l) := \sum_{h=1}^{l} x_{i_h i_{h-1}} + \sum_{h=2}^{l} x^*(\{i_1,\ldots,i_{h-1}\},i_h) - \sum_{h=2}^{l} y^*_{i_h}.$$

Notice that $\pi(i_1,\ldots,i_l)$ cannot exceed the degree of violation of the YSEC associated with $S := \{i_1,\ldots,i_l\}$; hence one as $\pi(i_1,\ldots,i_l) \le 0$ whenever all YSEC are satisfied by x^*.

According to (6.4), the only descendent nodes of ν that need to be generated are those associated with a sequence $(i_1,\ldots,i_l,i_{l+1})$ such that

$$x^*_{i_l i_{l+1}} > \phi_{max} - \pi(i_1,\ldots,i_l). \tag{6.5}$$

Notice that both quantities $\phi(i_1,\ldots,i_l)$ and $\pi(i_1,\ldots,i_l)$ can be computed along the branching tree as

$$\phi(i_1,\ldots,i_l) = \phi(i_1,\ldots,i_{l-1}) + x^*_{i_1 i_l} + x^*_{i_l i_{l-1}} + x^*(\{i_1,\ldots,i_{l-2}\},i_{l-1}) - y_{i_l}$$

and

$$\pi(i_1,\ldots,i_l) = \pi(i_1,\ldots,i_{l-1}) + x^*_{i_l i_{l-1}} + x^*(\{i_1,\ldots,i_{l-1}\},i_l) - y_{i_l},$$

where $\phi(i_1) := \pi(i_1) := 0$ for a singleton sequence (i_1).

Restriction (6.5) is very effective in practice, and reduce dramatically the number of nodes typically generated in the enumeration. The described separation procedure proved to be rather successful in that it requires a small fraction of the overall computational time, but unfortunately does not produce a big number of violated cuts.

Symmetric Inequalities

According to Fischetti and Toth [58] and Fischetti, Lodi and Toth [52], an ATPP inequality $\alpha x+\beta y+\gamma z \leq \alpha_0$ is called symmetric when $\alpha_{ij} = \alpha_{ji}$ for all $(v_i, v_j) \in A$. Indeed, symmetric inequalities can be thought of as derived from valid inequalities for the STPP. In a similar way than we have defined the variables x, let us define the variables $\overline{x}$ as follows. Let $\overline{x}_{ij} = 1$ if edge $[v_i, v_j] \in E$ belongs to the optimal STPP solution; $\overline{x}_{ij} = 0$ otherwise. Every inequality $\sum_{[v_i,v_j]\in E} \alpha_{ij}\overline{x}_{ij} + \beta y + \gamma z \leq \alpha_0$ for STPP can be transformed into a valid ATPP inequality by simply replacing $\overline{x}_{ij}$ by $x_{ij} + x_{ji}$ for all edges $[v_i, v_j] \in E$. This produces the symmetric inequality $\alpha x + \beta y + \gamma z \leq \alpha_0$, where $\alpha_{ij} = \alpha_{ji} = \alpha$ for all $v_i, v_j \in V, \quad i \neq j$. Conversely, every symmetric ATPP inequality $\alpha x + \beta y + \gamma z \leq \alpha_0$ corresponds to the valid STPP inequality $\sum_{[v_i,v_j]\in E} \alpha_{ij}\overline{x}_{ij} + \beta y + \gamma z \leq \alpha_0$.

The above correspondence implies that every separation algorithm for inequalities from the STPP can be used, as a "black box", for the ATPP as well. To this end, given the ATPP fractional point (x^*, y^*, z^*) one first defines the undirected counterpart $(\overline{x}^*, y^*, z^*)$ of (x^*, y^*, z^*) by means of the transformation

$$\overline{x}^*_{ij} := x^*_{ij} + x^*_{ji} \qquad \text{for all } [v_i, v_j] \in E,$$

and then applies the STPP separation algorithm to $(\overline{x}^*, y^*, z^*)$. On return, the detected most violated STPP inequality is transformed into ATPP counterpart, both inequalities having the same degree of violation.

Thus, we have that the separation algorithms for the inequalities described in Laporte, Riera and Salazar [102] and in the previous chapter for the symmetric TPP are also valid for the ATPP.

6.4 COMPUTATIONAL RESULTS

The here-proposed approaches have been implemented in C++ on a PC AMD 1333 MHz. ABACUS 2.2 linked with CPLEX 6.0 has been used as a framework (see Jünger and Thienel [89] for details on this software). A time limit of two hours has been established for the running time of our algorithms.

To test the performances of our code, we have considered ATPP instances obtained by using the random generator described in Singh and van Oudheusden [141], since this is the only today's article in which algorithms for the ATPP are tested. It is a generator of unrestricted ATPP instances in which the routing costs c_a are randomly generated in $[1, \tau]$, where τ is generated in [15,140]. Each market sells a number of products randomly generated in $[1, m]$, where $m = |K|$ is the number of products. Purchasing costs are randomly generated in $[0, \omega]$ where ω is generated in [5,75] for each market. We have defined instances with $|V| \in \{50, 100, 150, 200\}$ and $|K| \in \{50, 100, 150, 200\}$. For each type we have generated five instances by considering different seeds, thus our benchmark library contains 80 unrestricted ATPP instances.

In order to consider ATPP instances with restricted offers, the generator of Singh and van Oudheusden has been extended in the following way. For each prod-

uct p_k and each market v_i, q_{ki} has been randomly generated in [1,15] and $d_k := \lceil \lambda \max_{v_i \in M_k} q_{ki} + (1-\lambda) \sum_{v_i \in M_k} q_{ki} \rceil$ for $\lambda \in \{0.5, 0.8, 0.9, 0.95, 0.99\}$. Observe that parameter λ controls the demand of each product, and therefore it affects the number of visited markets in an optimal ATPP circuit: the smaller λ is, the bigger is the number of visited markets.

Tables 6.1–6.3 show statistical results from our computational experiments testing the branch-and-cut algorithm for the ATPP. The heading columns have the following meaning:

$|V|$**:** number of vertices (i.e., $n+1$);

$|K|$**:** number of products (i.e., m);

λ**:** value of the parameter λ in the generation (only for restricted ATPP instances);

solved: number of instances solved before the time limit (over 5 trials);

$\#$**:** average number of vertices in the optimal solutions;

2sec: average number of constraints (3.27) separated;

ysec: average number of constraints (3.20) separated;

zsec: average number of constraints (3.36) separated;

2mat: average number of constraints (3.12) separated;

D_l^+**:** average number of constraints (3.32)–(3.33) separated;

Nodes: average number of nodes explored during the branch-and-cut execution;

%UB: average gap between the heuristic and the optimal solutions at the end of the root node, over the optimal solution value;

%LB: average gap between the fractional and the optimal solutions at the end of the root node, over the optimal solution value;

Root-t: average computational time at the end of the root node;

Total-t: average computational time for the whole branch-and-cut execution.

According to Table 6.1, the branch-and-cut algorithm described above was able to solve most of the 80 unrestricted ATPP instances. Only 15 instances has not been solved before the time limit. Only those instances stopping before the time limit have been taken into account in the average. The difficulty of the problem grows clearly as the number of available markets and required products do. The number of visited markets in an optimal solutions tends to remain small, even when $|K| = 200$. This is mainly due to the ratio between the routing cost and the pricing cost in an optimal solution. That is, the higher the pricing costs are, the bigger the number of markets in the optimal solution is.

All the separations procedures described in Section 6.3 succeeded in finding some violated constraints. Computational experience shows that constraints (3.27), (3.20) and (3.36) become quite relevant, since the average computational time grows when the their separation procedures are unavailable. This behaviour cannot be extended to Constraints (3.12), (3.32) and (3.33). Computational time shows a no important increase when those constraints are avoided. By considering all the separated constraints, the lower bound at the end of root node has never been bigger than 1% with respect to the cost of an optimal solution. A similar result applies also to the upper bound obtained by applying the initial and primal heuristics. Because of the small gap between the lower and the upper bounds at the end of the root node, the exact algorithm requires a branching scheme, but despite of this, our approach have concluded, in most of instances, with the optimality proof before the time limit.

Tables 6.2 and 6.3 show the statistical results when the branch-and-cut code is used to solve the restricted ATTP instances. Columns $\#$ shows how important is the choice of the values of λ if different size of the optimal solutions must be considered.

The hardness of solving the restricted instances with the branch-and-cut code is observed in tables 6.2 and 6.3. The total computational time grows with the parameter λ (which is also related to the length of an optimal circuit). The computational time attains its maximum at $\lambda = 0.95$, and immediately the hardness begins to decrease. As it has also been observed in the unrestricted ATPP instances, the most relevant constraints are (3.27), (3.20) and (3.36), and the upper and lower bounds at the end of the root node are very close to the optimal solution value.

All instances of Table 6.2 (100 ATPP instances with $|V| = 50$) have been solved up to optimality before the time limit. However, 25 over 100 instances with $|V| = 100$ remain unsolved with our time limit, as observed in Table 6.3. Notice that, only one over 20 instances with $\lambda = 0.95$ has been solved before the time limit in this table.

The transformation from the ATPP into the STPP presented in Section 2.4 has been also computationally tested on the two previous families of instances. Notice that a transformation of an ATPP instance with $|V|$ vertices produces a new STPP instance with $2|V|$ vertices and with a minor increment of the number of edges. Therefore, the size of the STPP instance is still reasonable for available exact algorithms.

Tables 6.4, 6.5 and 6.6 show a comparative study between the specific branch-and-cut algorithm for the ATPP and the above mentioned transformation. As in the previous tables, the three first columns show the cardinality of the instance and the number of markets involved in the optimal circuit generated. The next four columns, both for the specific branch-and-cut and for the transformation respectively, show the number of instances solved before the time limit (*Solved*), the percentage of the gap between the upper and lower bound over the upper bound at the root node (*%gap*), the computational time consumed at the root node (*Root-t*), and the total computational time taken by the optimal algorithm (*Total-t*).

Table 6.4 compares the original branch-and-cut algorithm against the transformation approach in the unrestricted ATPP instances. The specific branch-and-cut seems to be more efficient not only with respect to the running time but also with respect to the gap between the upper and lower bound. However, as long as the number of markets is increased, the different between these two approaches is reduced.

Tables 6.5 and 6.6 are related to restricted TPP instances. Also on these harder instances, the branch-and-cut code shows better performance than the transformation approach, even if there are several exceptions when $|V| = 100$. The smaller gap of the direct approach is due to the new *ad hoc* inequalities and the heuristic approaches.

Table 6.1 Solving unrestricted ATPP instances with the branch-and-cut algorithm.

$\|V\|$	$\|K\|$	#	*Solved*	*2sec*	*ysec*	*zsec*	*2mat*	D_l^+	*Nodes*	*%UB*	*%LB*	*Root-t*	*Total-t*
50	50	9.2	5	506.8	60.6	848.4	8.0	8.6	13.8	0.47	0.33	1.0	5.6
	100	12.4	5	586.8	110.0	1357.2	6.4	2.8	17.0	0.58	0.20	1.6	8.8
	150	14.0	5	827.8	152.8	2345.2	5.4	4.0	19.4	0.38	0.12	1.8	15.8
	200	16.0	5	1329.4	220.4	3925.0	6.6	4.4	25.8	0.33	0.15	1.8	25.6
100	50	6.0	5	223.8	44.0	1062.0	5.2	2.8	7.8	0.12	0.17	17.0	36.8
	100	12.0	5	9378.2	1124.2	12619.4	32.8	18.6	76.2	0.79	0.47	21.2	411.0
	150	14.8	5	31966.6	4024.8	55081.8	124.2	43.2	265.0	0.76	0.48	25.4	1646.4
	200	17.2	5	44964.0	6431.6	97084.0	246.0	62.4	329.4	0.50	0.38	24.2	2237.6
150	50	7.6	5	5343.2	598.6	6544.2	20.8	28.2	49.4	0.90	0.66	87.2	812.2
	100	10.6	5	17261.4	1958.4	26797.4	34.4	17.8	93.4	0.77	0.57	99.0	2302.8
	150	14.4	5	13148.4	2301.2	31251.8	77.2	22.0	147.4	0.74	0.44	69.0	2247.4
	200	15.8	1	32748.6	4261.0	78140.4	132.0	31.4	300.2	0.68	0.68	100.4	1428.0
200	50	7.8	5	4186.6	326.8	6694.2	19.0	16.8	32.2	0.57	0.59	277.8	1484.8
	100	10.2	3	15039.2	2374.4	30172.0	64.2	25.8	117.4	0.51	0.61	172.6	3037.7
	150	13.2	1	25074.2	2315.8	41014.2	86.8	20.8	126.6	0.64	0.65	241.0	1605.0
	200	16.8	0	32170.0	2351.6	45997.8	48.4	7.6	119.8	0.65	0.78	307.8	-

Table 6.2 Solving ATPP instances with $|V| = 50$ with the branch-and-cut algorithm.

$\|K\|$	λ	*solved*	#	*2sec*	*ysec*	*zsec*	*2mat*	D_l^+	*Nodes*	*%UB*	*%LB*	*Root-t*	*Total-t*
	0.50	5	50.0	636.2	1.8	35.0	90.2	133.4	2.2	0.01	0.00	0.0	0.6
	0.80	5	40.2	294.4	3.2	32.4	132.4	175.6	14.0	0.04	0.01	0.0	1.6
50	0.90	5	27.4	390.8	22.4	179.0	712.8	857.6	41.2	0.07	0.08	0.0	15.2
	0.95	5	18.0	664.8	62.4	718.8	1017.4	2723.0	72.6	0.24	0.19	0.0	31.4
	0.99	5	10.0	484.0	20.0	370.2	115.2	1070.2	5.0	0.62	0.30	0.6	6.2
	0.50	5	50.0	1325.2	2.4	84.2	166.6	525.2	7.4	0.00	0.00	0.2	3.0
	0.80	5	50.0	594.4	2.8	64.0	88.2	253.2	40.2	0.01	0.00	0.0	3.0
100	0.90	5	42.8	668.8	19.4	257.0	739.8	1032.4	45.6	0.09	0.04	0.0	27.8
	0.95	5	28.6	12748.2	635.4	10177.4	15147.2	37812.2	543.8	0.21	0.18	0.2	800.8
	0.99	5	15.8	1436.8	40.0	1840.6	590.4	5039.8	17.4	0.34	0.24	1.6	31.6
	0.50	5	50.0	1979.6	2.4	101.0	134.2	678.6	8.6	0.00	0.00	0.0	2.8
	0.80	5	50.0	909.8	3.4	133.6	176.2	650.0	8.0	0.01	0.00	0.8	3.2
150	0.90	5	46.2	617.0	4.6	148.8	229.4	764.0	22.8	0.02	0.01	0.0	8.0
	0.95	5	34.2	5457.2	209.4	3176.8	5070.2	15189.8	212.8	0.12	0.10	0.2	424.4
	0.99	5	18.4	2669.4	70.8	2464.2	626.0	6527.4	28.8	0.42	0.18	2.4	71.4
	0.50	5	50.0	2704.2	2.0	94.0	61.4	542.8	12.8	0.00	0.00	0.0	1.8
	0.80	5	48.8	1254.8	2.2	154.6	184.2	917.6	6.4	0.00	0.00	0.8	5.2
200	0.90	5	48.8	779.6	3.8	154.2	145.8	692.2	8.0	0.02	0.01	0.8	6.4
	0.95	5	40.0	3652.8	124.4	2639.2	3538.4	11679.8	102.6	0.08	0.07	1.0	344.8
	0.99	5	21.4	7344.0	180.0	5862.6	1434.6	16103.6	67.0	0.29	0.20	2.6	221.6

Table 6.3 Solving ATPP instances with $|V| = 100$ with the branch-and-cut algorithm.

$\|K\|$	λ	*solved*	#	*2sec*	*ysec*	*zsec*	*2mat*	D_l^+	*Nodes*	*%UB*	*%LB*	*Root-t*	*Total-t*
	0.50	5	100.0	1546.8	122.2	136.2	1095.2	502.2	576.4	0.01	0.00	2.4	265.0
	0.80	5	91.2	714.2	28.8	78.2	752.8	292.0	269.4	0.02	0.00	2.2	83.0
50	0.90	5	60.8	4998.4	272.0	2834.0	16540.6	8709.6	393.6	0.06	0.04	1.8	1311.6
	0.95	1	35.0	19694.2	1189.8	13954.4	65805.4	57756.2	1440.8	0.26	0.16	1.8	6767.8
	0.99	5	13.6	6407.6	262.4	8251.0	3857.8	17495.6	100.0	0.13	0.56	5.6	2029.6
	0.50	5	100.0	2856.0	13.0	101.4	433.6	534.4	270.0	0.00	0.00	4.4	114.2
	0.80	5	90.4	1325.0	6.0	81.0	476.4	460.6	87.8	0.01	0.00	3.2	33.0
100	0.90	5	71.6	1271.6	25.6	343.2	1320.4	1390.2	76.2	0.03	0.01	3.0	190.2
	0.95	0	-	-	-	-	-	-	-	0.14	0.13	4.0	-
	0.99	3	20.0	111815.4	3087.8	89527.4	31175.0	189650.4	988.4	0.74	0.33	10.8	5854.4
	0.50	5	100.0	4542.6	23.4	108.4	388.2	670.6	347.8	0.00	0.00	2.6	185.4
	0.80	5	100.0	22563.2	1763.4	46.8	901.2	585.6	5.4	0.00	0.00	16.2	16.2
150	0.90	5	90.4	3122.6	1120.8	152.8	2632.0	1672.0	96.2	0.02	0.01	8.8	8.8
	0.95	0	-	-	-	-	-	-	-	0.07	0.10	5.0	-
	0.99	5	18.4	108864.2	1837.0	25643.2	20542.6	174276.0	694.2	0.30	0.26	12.2	5623.6
	0.50	5	100.0	5866.4	15.0	215.8	625.8	1181.0	203.4	0.00	0.00	5.0	97.6
	0.80	5	100.0	3338.4	77.4	347.0	676.6	1089.4	495.0	0.00	0.00	46.4	294.6
200	0.90	5	95.0	2355.8	119.6	656.0	999.8	1580.8	617.8	0.01	0.01	9.4	7.2
	0.95	0	-	-	-	-	-	-	-	0.06	0.09	8.6	-
	0.99	2	22.0	103687.6	1666.8	46234.8	13034.6	146714.6	438.0	0.31	0.23	17.6	6675.6

Table 6.4 Branch-and-cut vs Transformation for unrestricted ATPP instances.

			Branch-and-cut				*Transformation*			
$\|V\|$	$\|K\|$	#	*Solved*	*%Gap*	*Root-t*	*Total-t*	*Solved*	*%Gap*	*Root-t*	*Total-t*
50	50	9.2	5	0.80	1.0	5.6	5	1.42	3.0	13.2
	100	12.4	5	0.78	1.6	8.8	5	1.01	2.8	20.0
	150	14.0	5	0.51	1.8	15.8	5	0.50	3.6	40.8
	200	16.0	5	0.49	1.8	25.6	5	0.45	3.6	61.0
100	50	6.0	5	0.29	17.0	36.8	5	0.55	30.2	57.6
	100	12.0	5	1.26	21.2	411.0	5	1.37	44.4	956.0
	150	14.8	5	1.23	25.4	1646.4	4	1.72	38.0	2137.0
	200	17.2	5	0.88	24.2	2237.6	4	1.05	28.2	1157.0
150	50	7.6	5	1.56	87.2	812.2	5	2.31	148.6	1371.4
	100	10.6	5	1.34	99.0	2302.8	5	1.69	204.4	3256.0
	150	14.4	5	1.19	69.0	2247.4	5	1.50	111.4	3880.0
	200	15.8	1	1.35	100.4	1428.0	1	1.58	131.0	2609.0
200	50	7.8	5	1.17	277.8	1484.8	5	1.23	290.6	1663.2
	100	10.2	3	1.13	172.6	3037.7	3	1.55	189.6	3511.7
	150	13.2	1	1.29	241.0	1605.0	1	1.47	279.2	2162.0
	200	16.8	0	1.43	307.8	-	0	1.74	418.0	-

Table 6.5 Branch-and-cut vs Transformation for ATPP instances with $|V| = 50$.

			Branch-and-cut				*Transformation*			
$\|K\|$	λ	#	*Solved*	*%Gap*	*Root-t*	*Total-t*	*Solved*	*%Gap*	*Root-t*	*Total-t*
	0.50	49.0	5	0.01	0.0	0.6	5	0.03	0.2	6.0
	0.80	40.6	5	0.05	0.0	1.6	5	0.05	0.2	5.4
50	0.90	27.4	5	0.15	0.0	15.2	5	0.20	0.0	37.6
	0.95	18.0	5	0.43	0.0	31.4	5	0.41	0.0	62.0
	0.99	10.0	5	0.93	0.6	6.2	5	1.35	1.4	18.6
	0.50	50.0	5	0.00	0.2	3.0	5	0.01	1.0	2.8
	0.80	50.0	5	0.01	0.0	3.0	5	0.01	0.8	2.0
100	0.90	42.8	5	0.14	0.0	27.8	5	0.12	0.4	74.0
	0.95	27.3	5	0.38	0.2	800.8	5	0.32	0.3	921.3
	0.99	15.8	5	0.58	1.6	31.6	5	0.79	2.0	82.0
	0.50	50.0	5	0.00	0.0	2.8	5	0.01	1.2	5.4
	0.80	50.0	5	0.01	0.8	3.2	5	0.00	0.8	6.6
150	0.90	46.4	5	0.04	0.0	8.0	5	0.03	0.8	17.4
	0.95	34.0	5	0.22	0.2	424.4	5	0.18	1.0	443.3
	0.99	18.2	5	0.61	2.4	71.4	5	0.52	3.0	222.8
	0.50	49.8	5	0.00	0.0	1.8	5	0.00	2.2	6.8
	0.80	48.8	5	0.01	0.8	5.2	5	0.01	1.6	3.2
200	0.90	49.0	5	0.03	0.8	6.4	5	0.03	1.2	9.2
	0.95	40.0	5	0.15	1.0	344.8	5	0.14	1.0	561.8
	0.99	21.4	5	0.48	2.6	221.6	5	0.50	2.0	520.2

Table 6.6 Branch-and-cut vs Transformation for ATPP instances with $|V| = 100$.

			Branch-and-cut				*Transformation*			
$\|K\|$	λ	#	*Solved*	*%Gap*	*Root-t*	*Total-t*	*Solved*	*%Gap*	*Root-t*	*Total-t*
	0.50	99.0	5	0.01	2.4	265.0	5	0.02	6.6	35.4
	0.80	90.8	5	0.03	2.2	83.0	5	0.03	7.0	92.8
50	0.90	64.0	5	0.10	1.8	1311.6	1	0.03	2.0	129.0
	0.95	35.0	1	0.42	1.8	6767.8	0	-	-	-
	0.99	13.5	5	0.69	5.6	2029.6	4	0.70	9.8	1947.0
	0.50	97.2	5	0.00	4.4	114.2	5	0.01	8.6	101.6
	0.80	90.2	5	0.01	3.2	33.0	5	0.01	9.6	34.8
100	0.90	69.0	5	0.05	3.0	190.2	4	0.05	3.3	535.8
	0.95	-	0	-	-	-	0	-	-	-
	0.99	19.0	3	1.07	10.8	5854.4	2	1.06	12.0	5736.5
				0.00						
	0.50	100.0	5	0.00	2.6	185.4	5	0.00	21.4	21.4
	0.80	99.8	5	0.00	16.2	16.2	5	0.01	9.0	9.0
150	0.90	90.2	5	0.03	8.8	8.8	5	0.03	7.8	7.8
	0.95	-	0	-	-	-	0	-	-	-
	0.99	19.0	5	0.56	12.2	5623.6	1	0.88	11.0	4729.0
				0.00						
	0.50	100.0	5	0.00	5.0	97.6	5	0.00	30.4	122.2
	0.80	100.0	5	0.00	46.4	294.6	5	0.00	10.8	46.8
200	0.90	95.4	5	0.02	9.4	7.2	5	0.02	6.0	7.2
	0.95	-	0	-	-	-	0	-	-	-
	0.99	22.0	2	0.54	17.6	6675.6	2	0.56	17.5	5584.0

7

The Biobjective Symmetric Traveling Purchaser Problem

The purpose of this chapter is to present a new approach to solve the *Biobjective Traveling Purchaser Problem* (2TPP), defined as the biobjective version of the Traveling Purchaser Problem (TPP), referred as 1TPP to emphatize the unicriterion function. A new computational technique to improve the efficiency of our approach is also introduced in this Chapter. This technique consists in making use of previously computed cuts, in order to enlarge the initial cut pool during the computation of further non-dominated points.

A simple cycle in G passing through the depot v_0 and a subset of markets is called *feasible solution* if for each product p_k the cycle visits enough markets in M_k to allow buying the required d_k units. Given a feasible solution $\sigma = (V(\sigma), E(\sigma))$ visiting nodes $V(\sigma) \subseteq V$ and routing edges $E(\sigma) \subseteq E$, there is an associated pricing cost defined by

$$\texttt{price}(\sigma) := \sum_{p_k \in K} \min \left\{ \begin{array}{r} \sum_{v_i \in M_k \cap V(\sigma)} b_{ki} z_{ki} : \sum_{v_i \in M_k \cap V(\sigma)} z_{ki} = d_k \\ \text{and } z_{ki} \leq q_{ki} \quad \text{for all } v_i \in M_k \cap V(\sigma) \end{array} \right\} \tag{7.1}$$

where z_{ik} is a unknown value representing the amount of product p_k to be purchased at v_i, and an associated Traveling cost defined as

$$\texttt{travel}(\sigma) := \sum_{e \in E(\sigma)} c_e.$$

Then the 2TPP looks for determining a feasible solution σ minimizing both $\texttt{price}(\sigma)$ and $\texttt{travel}(\sigma)$.

To our knowledge all previous studies of the 2TPP are restricted to the case where the two objective functions are replaced by a single composite objective function obtained by simply adding the Traveling and the pricing costs. This single-objective problem will be denoted by 1TPP. Nevertheless, in real applications both objectives are not comparable, and a more sophisticated procedure is needed to address the biobjective structure of the 2TPP.

Our aim in this chapter is to address the 2TPP, providing a new approach to generate non-dominated and extreme efficient points with respect to both criteria. This approach takes the advantage of solving 1TPPs by a branch-and-cut algorithm making use of a heuristic method based on a common cut-pool structure that saves the previously generated cuts. Since the model of this problem has been already described, we refer to the reader to Chapter 3 for details. The general algorithm of this approach is described in Section 7.1. In this section we also describe the common cut-pool structure and give an illustrative example. Finally computational results are shown in Section 7.4, showing the good performance of our approach on instances with $|V| \leq 100$ and $|K| \leq 200$.

7.1 THE OVERALL ALGORITHM

The aim of this section is to describe the general procedure we have developed to generate a set of solutions for the 2TPP. Fundamental concepts on Multicriteria Optimization are presented bellow. However, we have to remark that these definitions are not unique in literature, and we are following the notation described in Ehrgott [49].

Definition 7.1. Let f_1 and f_2 be two performance criteria. Then the *criterion space* is

$$\mathcal{Z} = \{z \in \mathbb{R}^2 | z = (f_1(\sigma), f_2(\sigma)), \sigma \in \mathcal{P}\},$$

where $\mathcal{P}$ is the decision space defined by the convex hull of all feasible solutions of the 2TPP.

Definition 7.2. A solution $\sigma^* \in \mathcal{P}$ is called *Pareto optimal* if there is not $\sigma \in \mathcal{P}$ such that $f_1(\sigma) \leq f_1(\sigma^*)$ and $f_2(\sigma) \leq f_2(\sigma^*)$, where at least one of the inequalities is strict. If σ^* is Pareto optimal then $(f_1(\sigma^*), f_2(\sigma^*))$ is called *efficient* point. The set of all Pareto optimal solutions is called the *Pareto set*. The set of all efficient points is called the *efficient set*.

Definition 7.3. The set of the *supported efficient solutions* is the set of Pareto optimal solutions which are optimal for any weighted sum of the objectives. The remaining Pareto optimal solutions are the *non-supported efficient solutions*.

The aim of the proposed algorithm is to compute the efficient set, including both supported and non-supported points in the criterion space. Our proposal is based on a hybrid method that combines the *weighting method* with additional constraints (see Ehrgott and Gandibleux [50] for details on similar algorithms for bicriterion combinatorial optimization problems). It is also related to the *parametric approach*

introduced by Solan [142] and used to solve other problems, e.g. T'kindt, Billaut and Proust [146] and Visée, Teghem, Pirlot and Ulungu [150]. This scheme combines linearly both criteria and introduce a weighting factor for each of them. If these weighting coefficients, denoted here by ω_1 and ω_2, are interpreted as parameters then we obtain a linear weighting method which can be used for the generation of the supported set. Without loss of generality the normalization $\omega_1 + \omega_2 = 1$ may be applied, so we will only relate to ω. This weighting procedure is embedded in a binary search algorithm which explores different regions of the decision space by making use of additional constraints restricting the criterion space. This mechanism allows us to find the set of non-supported efficient points as well.

The basic steps of the general method are showed in Figure 1.4. The initial step computes the two starting points ($f_1^{(1)}$,$f_2^{(1)}$) and ($f_1^{(2)}$,$f_2^{(2)}$) by optimizing hierarchically both criteria f_1 and f_2. How to obtain these two initial subproblems is described in Section 7.1. These two initial points define the two first efficient points and therefore initialize the efficient set SE. Moreover, they also define the first interval in $\mathbb{R}^2$ (denoted by ([($f_1^{(1)}$,$f_2^{(1)}$)...($f_1^{(2)}$,$f_2^{(2)}$)]) to be explored, which initializes the list of pending intervals $\mathcal{L}^I$. This method get iteratively an interval from $\mathcal{L}^I$, solves a single objective problem (if it is feasible) providing an efficient point to be included in SE and two new intervals to be explored, which are stored in $\mathcal{L}^I$. This loop is repeated while $\mathcal{L}^I$ is not empty. Figure 1.6 illustrates this step. Notice that this step performs a binary search of efficient points in the criterion space.

Each single-objective problem, named W1TPP, is defined by a given interval [($f_1^{(1)}$,$f_2^{(1)}$)...($f_1^{(2)}$,$f_2^{(2)}$)] as follows

$$\min \omega f_1(\sigma) + (1-\omega) f_2(\sigma) \tag{7.2}$$

subject to

$$\sigma \in \mathcal{P} \tag{7.3}$$

$$f_1(\sigma) < f_1^{(1)} \tag{7.4}$$

$$f_2(\sigma) < f_2^{(2)}, \tag{7.5}$$

where $\mathcal{P}$ can be replaced by constraints (3.2)–(3.9), and $\omega := \frac{\alpha}{\alpha - 1}$, where $\alpha := \frac{f_2^{(2)} - f_2^{(1)}}{f_1^{(1)} - f_1^{(2)}}$. If only the supported efficient points were required, constraints (7.4) and (7.5) should be extended with the following constraint

$$\omega_1 f_1(\sigma) + (1-\omega_1) f_2(\sigma) < \omega_1 f_1^{(1)} + (1-\omega_1) f_2^{(2)}. \tag{7.6}$$

Since we assume that the input data are integer numbers, the constraints (7.3)–(7.6) can be replaced by

$$\sigma \in \mathcal{P}$$

$$f_1(\sigma) \leq f_1^{(1)} - 1$$

$$f_2(\sigma) \leq f_2^{(2)} - 1$$

$$\omega f_1(\sigma) + (1-\omega) f_2(\sigma) \leq \lfloor \omega_1 f_1^{(1)} + (1-\omega_1) f_2^{(2)} - \epsilon \rfloor,$$

respectively, for $\epsilon > 0$ an small parameter. Figure 7.2 shows how this approach is able to generate the non-supported efficient points as well if there were not supported efficient points in the explored interval.

$f_1^{(1)} := \min_{\sigma \in \mathcal{P}} f_1(\sigma)$
$f_2^{(1)} := \min_{\sigma \in \mathcal{P}} (f_2(\sigma) | f_1(\sigma) \leq f_1^{(1)})$
$f_2^{(2)} := \min_{\sigma \in \mathcal{P}} f_2(\sigma)$
$f_1^{(2)} := \min_{\sigma \in \mathcal{P}} (f_1(\sigma) | f_2(\sigma) \leq f_2^{(2)})$
$\mathcal{L}^I :=$ `{[(`$f_1^{(1)}$`,`$f_2^{(1)}$`)...(`$f_1^{(2)}$`,`$f_2^{(2)}$`)]}`
$ND := \{(f_1^{(1)}, f_2^{(1)}), (f_1^{(2)}, f_2^{(2)})\}$
while $\mathcal{L}^I \neq \varnothing$
  `Select from` $\mathcal{L}^I$ `an interval` $[(f_1^{(1)}, f_2^{(1)}) \dots (f_1^{(2)}, f_2^{(2)})]$
  $\mathcal{L}^I := \mathcal{L}^I \setminus \{[(f_1^{(1)}, f_2^{(1)}) \dots (f_1^{(2)}, f_2^{(2)})]\}$
  $\alpha := \frac{f_2^{(2)} - f_2^{(1)}}{f_1^{(1)} - f_1^{(2)}}$
  $\omega := \frac{\alpha}{\alpha - 1}$
  $\sigma^* :=$ arg `W1TPP(`ω`,`$f_1^{(1)}$`,`$f_2^{(2)}$`)`
  if $\sigma^* \neq \varnothing$
    $ND := ND \cup (f_1(\sigma^*), f_2(\sigma^*))$
    $\mathcal{L}^I := \mathcal{L}^I \cup \{[(f_1^{(1)}, f_2^{(1)}) \dots (f_1(\sigma^*), f_2(\sigma^*))], [(f_1(\sigma^*), f_2(\sigma^*)) \dots (f_1^{(2)}, f_2^{(2)})]$

Fig. 7.1 Pseudocode of the hybrid algorithm.

Since we assume that the input data are integer numbers, the constraints (7.3)–(7.6) can be replaced by

$$\sigma \in \mathcal{P}$$

$$f_1(\sigma) \leq f_1^{(1)} - 1$$

$$f_2(\sigma) \leq f_2^{(2)} - 1$$

$$\omega f_1(\sigma) + (1-\omega) f_2(\sigma) \leq \lfloor \omega f_1^{(1)} + (1-\omega) f_2^{(2)} - \epsilon \rfloor,$$

respectively, for $\epsilon > 0$ an small parameter.

In order to solve each W1TPP we make use of a branch-and-cut algorithm similar to the algorithm proposed by Laporte, Riera and Salazar [102] for the 1TPP, but adapted for W1TPP to manage (7.4)–(7.6). The following two sections describe both the branch-and-cut approach and the specific changes performed for solving 2TPP.

Initial Efficient Points

Two specific implementations of the W1TPP have been performed for the computation of both initial points ($f_1^{(1)}$,$f_2^{(1)}$) and ($f_1^{(2)}$,$f_2^{(2)}$). These implementations perform a two phases procedure. The first phase optimizes f_1, that is $f_1^{(1)} := \min_{\sigma \in \mathcal{P}} f_1(\sigma)$, and f_2, that is $f_2^{(2)} := \min_{\sigma \in \mathcal{P}} f_2(\sigma)$. The second phase optimizes f_2 and f_1 subject to the

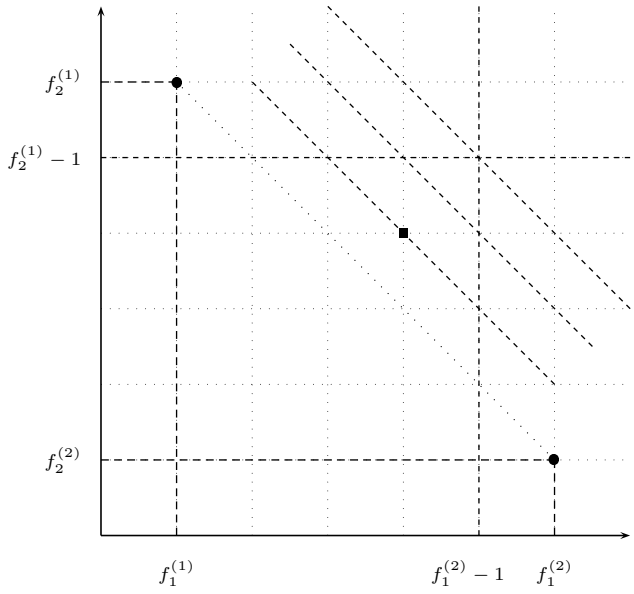


Fig. 7.2 Optimization step

optimal values obtained in the previous phase, that is $f_2^{(1)} := \min_{\sigma\in\mathcal{P}}\{f_2(\sigma)|f_1(\sigma) \leq f_1^{(1)}\}$ and $f_1^{(2)} := \min_{\sigma\in\mathcal{P}}\{f_1(\sigma)|f_2(\sigma) \leq f_2^{(2)}\}$. Notice that, ${f_1}^{(1)}$ is obtained solving the polynomial problem described by (7.1). A simple *ad hoc* algorithm with time complexity $O(|V|\log|V||K|)$ has been developed for this specific subproblem. Additionally, $f_2^{(1)}$ is the minimum cost feasible cycle. In this particular case the branch-and-cut approach has been specifically adapted by making use of an specific heuristic to solve the *Cycle Problem*. Once both $f_1^{(1)}$ and $f_2^{(1)}$ have been obtained, the problems $\min_{\sigma\in\mathcal{P}}\{f_2(\sigma)|f_1(\sigma) \leq f_1^{(1)}\}$ and $\min_{\sigma\in\mathcal{P}}\{f_1(\sigma)|f_2(\sigma) \leq f_2^{(1)}\}$ are solved by the branch-and-cut for the W1TPP with additional constraints and choosing the weighting values properly.

An important observation is that this constrained 1TPP subproblem could stops with no feasible tour. This justifies the impossibility of finding feasible solutions in Step 1 and 4 on some instances.

7.2 COMMON CUT-POOL HEURISTIC

As already mentioned, the algorithm to solve solve the W1TPP subproblems has be embedded into an iterative approach that generates supported and non-supported efficient points. In order to speed up the performance of the overall procedure we have made use of a common cut-pool structure that saves those valid inequalities

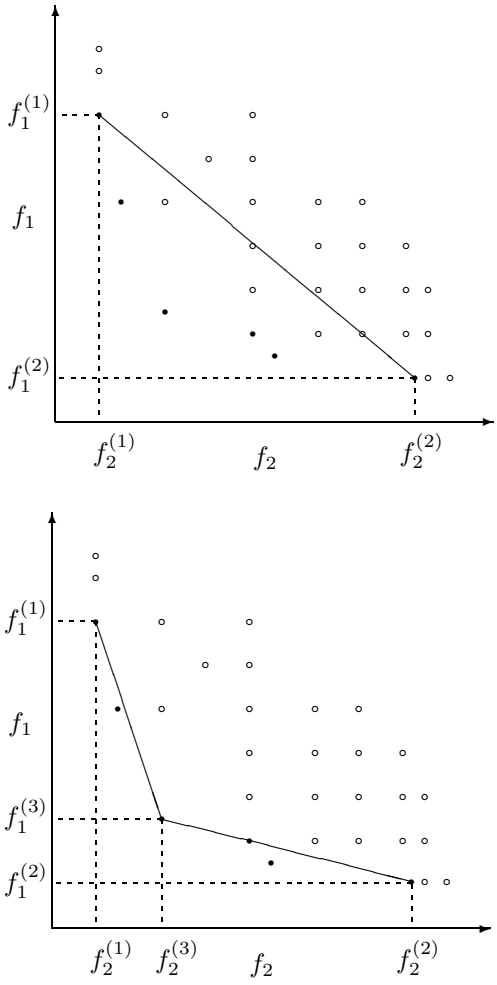


Fig. 7.3 Two new intervals obtained from the first optimization step.

separated during the resolution of subproblems, for being used latter on in forthcoming subproblems.

The motivation of saving useful cuts for a former subproblem in a cut pool to be available when solving latter subproblems is based on the connectivity of the Pareto optimal solutions (see Steuer [143]). Therefore, if the latter subproblems have in advance a pool structure containing cuts that have been useful in former subproblems, then they might be solved with less computational effort. This can be done since the cuts generated by our branch-and-cut during each subproblem are valid

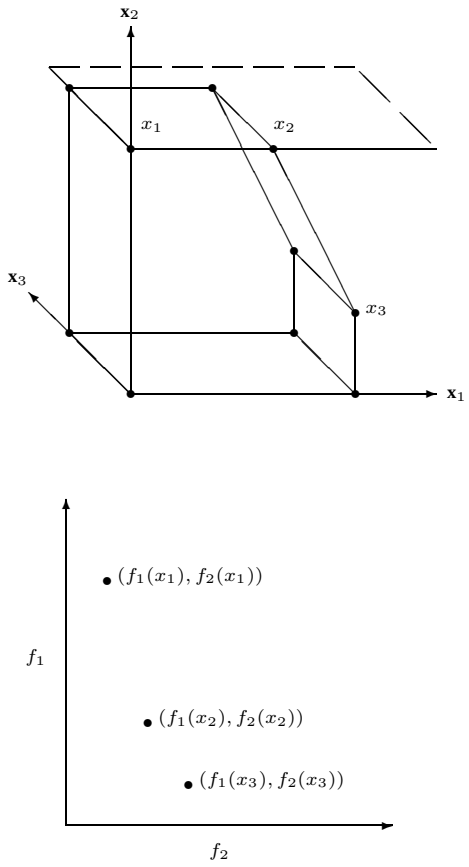


Fig. 7.4 Common Pool.

inequalities for all feasible tours, since the W1TPP's polyhedron is included in the 1TPP's polyhedron $\mathcal{P}$. This is the aim of the proposed heuristic approach.

More precisely, since the full description of the problem's polyhedron is unknown, a dynamic constraint generation has to be performed for each single criterion subproblem. Each single criterion problem produces a set of cuts that, as Fig. 7.4 illustrates, might be useful for solving forthcoming subproblems, (i.e. for computing other efficient points). In order to exploit those cuts, the data structure $\mathcal{L}^C$ is shared and updated by all subproblems. Since this structure has a limited size we do not allow the introduction of all violated cuts generated by all the separation procedures. On the contrary, we rank the violated inequalities and select the most violated ones (no more than 50). The selected inequalities are saved in the cut-pool and used to strengthen the current LP. To keep small the size of each LP, some unnecessary constraints are removed from the LP every five iterations. In this way, the final cut-pool list $\mathcal{L}^C$is the initial list of candidate inequalities to strengthen the next subproblem.

A similar consideration does not arise with the variables. In fact, a variable-pool is useful when each single W1TPP is solved due to the large number of z_{ik} variables, but according to our experiences, it was useless to save the variable-pool structure from one W1TPP to another. Thus a variable pool is initialized when approaching new subproblems. Computational behavior of this improvement is showed in Section 7.4, which compares some instance resolutions performed with and without the common-cut-pool structure.

7.3 ILLUSTRATIVE EXAMPLE

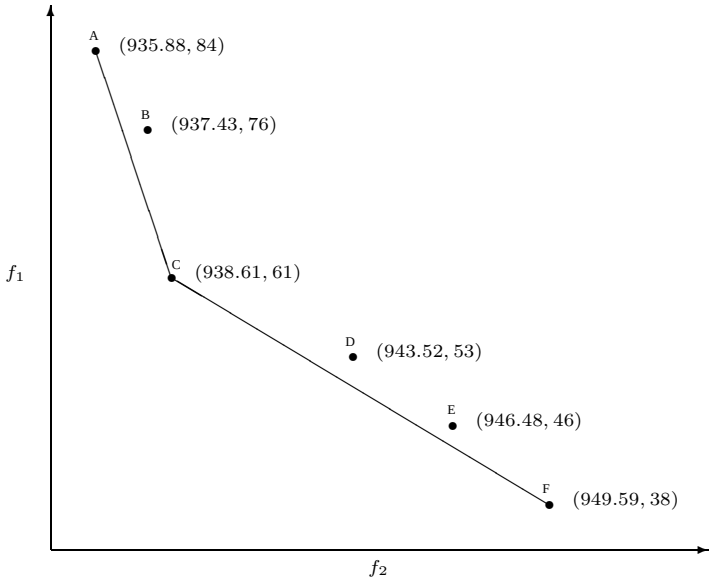


Fig. 7.5 The six efficient points from the example in Burstall [25].

In this section we introduce an example to illustrate how this approach can provide insight about the choice of a set of states as well as the sequence in which they have to be processed.

We consider an example proposed in the first article on the Traveling Purchaser Problem by Burstall [25] (named Batch#12) extracted from a real world applications in the industrial context. This example arose from the difficulty encountered by a firm manufacturing steel tubes, a member of the Tube Investments Group. A set of steel tubes has to be manufactured. According to the required final product the tubes are grouped into 8 batches. Those batches have to be processed by a multi-purpose

Table 7.1 The six efficient points from the example in Burstall [25].

	A	*B*	*C*	*D*	*E*	*F*
States	$s_{11} \rightarrow s_3 \rightarrow s_{17}$ $\rightarrow s_1 \rightarrow s_{13}$	$s_7 \rightarrow s_{11} \rightarrow s_{14}$ $\rightarrow s_{13} \rightarrow s_3$	$s_{11} \rightarrow s_1$ $\rightarrow s_{13} \rightarrow s_{17}$	$s_7 \rightarrow s_4$ $\rightarrow s_3 \rightarrow s_6$	$s_{11} \rightarrow s_1$ $\rightarrow s_{14}$	$s_4 \rightarrow s_6$ $\rightarrow s_7$
j_1	s_3	s_3	s_1	s_3	s_1	s_4
j_2	s_3	s_3	s_1	s_3	s_1	s_4
j_3	s_1	s_7	s_1	s_7	s_1	s_7
j_4	s_{11}	s_{11}	s_{11}	s_7	s_{11}	s_7
j_5	s_1	s_7	s_1	s_7	s_1	s_7
j_6	s_1	s_7	s_1	s_7	s_1	s_7
j_7	s_{13}	s_{13}	s_{13}	s_4	s_{14}	s_4
j_8	s_{17}	s_{14}	s_{17}	s_6	s_{14}	s_6
Change-over	84	76	61	53	46	38
Processing	935.88	937.43	938.61	943.52	946.48	949.59

machine, which is able to change its state depending on the task to be carried out. The machine has 19 potential states. Changing from a state to another takes a change-over time. On the other hand, the processing time of a batch depends on the state of the machine it is processed. Thus, a batch might be processed in several states of the machine, but spending different processing time.

We have solved the problem representing a trade-off between total change-over time and total processing time. We have obtained the six non-dominated points (A, B,..., F) (see Fig. 7.5) as well as the three extreme points (A, C, F). For each non-dominated point one solution is also described in Table 7.1, which contains not only information about the sequence of the states but also the assignment of jobs to states. The two first points and the related interval have been obtained by mean the procedure described in section 7.1. In the first step of optimization the supported point C has been obtained. This produces two news interval, [A, C] and [C, F], producing on its turn two new points (B and E) and four new intervals. The next interval to be examined is [A,B] (notice that in our particular implementation the list $\mathcal{L}^I$ is organized as a *queue*), but it produces a non-feasible subproblem, and therefore neither point nor intervals are obtained. The process continues iterating while the list of intervals remains no-empty.

7.4 COMPUTATIONAL RESULTS

To evaluate the performance of our proposal on 2TPP instances from literature the procedures were implemented in C++ and run on a Pentium 500 MHz computer running Linux. ABACUS 2.2 linked with CPLEX 6.0 was used as a framework (see Jünger and Thienel [89] for details on this software).

We have considered the following three classes of 2TPP test instances:

Class 1 contains 33-market symmetric instances defined with the same input data as in Singh and van Oudheusden [141]. These correspond to the largest instance size solved by these authors. The routing costs are those of a 33-vertex TSP described in Karg and Thompson [90] and do not satisfy the triangle inequality. The first vertex is the depot and all markets sell all products. Product prices are generated in [1,500] according to a discrete uniform distribution. We generated five instances with $|K|$=50, 100, 150, 200 and 250.

Class 2 instances are randomly generated by using the procedure described in Pearn and Chien [123]. Routing costs are randomly generated in $[1, \tau]$ where τ is generated in [15,140]. All products are available at all markets. Purchase costs are randomly generated in $[0, \lambda]$ where λ is generated in [5,75]. Contrary to Pearn and Chien, we have used symmetric routing cost instead of asymmetric costs. We defined instances with $|V|$=50 and 100, and $|K|$=50, 100 and 150.

Class 3 instances were defined by first generating $|V|$ integer coordinate vertices in the $[0, 1000] \times [0, 1000]$ square according to a uniform distribution and defining routing costs by Euclidean distances. Each product p_k was associated with $|M_k|$

randomly selected markets, where $|M_k|$ was randomly generated in $[1, |V|-1]$. The remaining characteristics of these instances are defined as for Class 1.

Tables (1)–(3) show details on our experiment. Each line in a table reports the average result on 5 instances. The columns have the following meaning:

$|V|$**:** number of markets plus one (the depot).

$|K|$**:** number of products.

#sub: average number of W1TPP instances solved for each 2TPP.

#P: average number of feasible W1TPP instances among #sub (i.e. number of points in the criterion space).

no-pool-t: average CPU time to solve each 2TPP without the common cut-pool data structure.

pool-t: average CPU time to solve each 2TPP using the common cut-pool structure.

The last four columns appear twice. The first set of columns refers to the computation of all efficient points, while the second set refers to the computation of the supported efficient points. Notice that the proposed mechanism produces one Pareto optimal solution in the decision space for each efficient point in the criterion space.

According to Tables 1 to 3, it is clear the benefit of using the dynamic pool structure described in Section 3.2. More precisely, on instance for Class 2, there is a minor penalty on the computational effort when computing the efficient solutions on small instances (for $|V| = 50$, the penalty is close to 4%), but in all the other situations there is a clear saving. Indeed, the common cut-pool approach provides a 19%, 14% and 30% of time saving when the set of efficient points is computed in instances of Classes 1, 2 and 3 respectively; and a 6%, 10% and 40% time saving computing when the set of supported efficient points is computed. The analysis is more evident from the summarizes in Tables 4-6, which show the percentage of improving time using the pool structure.

The CPU time taken to solve each LP was in all cases quite small if it is compared to the overall algorithm. Indeed, an estimation on the average CPU time for solving each W1TPP instance is obtained by dividing the CPU time showed in the tables by its correspondent value in column *#sub*. The time consumed is close to one minute in the worst case, but it includes the time taken to compute the heuristic, to call the separation procedure, to perform the LP solver, to maintenance the pool structure, and to perform other components of the branch-and-cut-algorithm.

Finally, our experiments have also proved that the overall approach is able to fully manage problem resolutions with up to 100 markets and 200 products.

Table 7.2 Results solving instances from Class 1

$\|V\|$	$\|K\|$	Efficient				Supported			
		#sub	#P	no-pool-t	pool-t	#sub	#P	no-pool-t	pool-t
33	50	215.8	109.2	1095.6	1196.2	49.0	25.0	43.2	42.8
	100	297.6	149.6	3606.2	4462.2	55.8	28.4	129.0	136.6
	150	342.6	172.0	11251.6	14726.2	58.6	29.8	361.4	405.4
	200	331.0	166.0	18960.0	25296.0	58.0	29.5	683.0	755.0

Table 7.3 Results solving instances from Class 2

$\|V\|$	$\|K\|$	Efficient				Supported			
		#sub	#P	no-pool-t	pool-t	#sub	#P	no-pool-t	pool-t
50	50	64.4	36.0	247.4	234.4	28.2	15.4	32.8	34.2
	100	82.4	45.0	478.0	443.2	32.0	16.6	65.8	60.6
	150	102.6	56.2	822.0	776.6	40.0	21.0	104.0	105.0
	200	100.6	56.0	1555.6	1447.0	36.6	19.0	235.0	227.2
100	50	56.0	32.8	2422.6	2314.4	27.6	15.0	590.4	581.2
	100	91.0	53.4	9466.4	7345.6	41.8	22.4	3745.4	3175.4
	150	126.6	75.8	22853.8	15979.2	54.0	28.6	7715.8	6020.4
	200	131.4	81.2	61313.0	41054.4	49.8	26.8	9467.4	5944.6

Table 7.4 Results solving instances from Class3

$\|V\|$	$\|K\|$	Efficient				Supported			
		#sub	#P	no-pool-t	pool-t	#sub	#P	no-pool-t	pool-t
50	50	69.6	44.0	692.0	447.8	22.6	12.4	75.2	41.0
	100	92.0	55.2	1303.2	979.0	33.0	18.0	219.6	139.6
	150	112.2	67.0	2064.4	1455.2	38.4	20.2	340.6	201.4
	200	115.4	68.6	3275.4	2215.2	37.4	20.0	407.0	255.2
100	50	58.8	36.4	5646.4	3413.8	28.6	16.0	1459.2	877.8
	100	82.6	49.8	15636.8	13604.6	33.2	17.6	4092.0	2247.4
	150	119.0	74.0	47885.4	29884.8	44.6	23.6	10406.6	6734.0
	200	128.8	80.4	66728.2	44162.4	45.8	24.8	10448.4	6113.6

Table 7.5 Summarize of results in Table 7.2

$\|V\|$	$\|K\|$	%-Eficient	%-Supported
33	50	8.41	-0.93
	100	19.18	5.56
	150	23.59	10.85
	200	25.05	9.54
		19.06	6.25

Table 7.6 Summarize of results in Table 7.3

$\|V\|$	$\|K\|$	%-Eficient	%-Supported
50	50	5.25	-4.27
	100	7.28	7.90
	150	5.52	-0.96
	200	6.98	3.32
100	50	4.47	1.56
	100	22.40	15.22
	150	30.08	21.97
	200	33.04	37.21
		14.38	10.24

Table 7.7 Summarize of results in Table 7.4

$\|V\|$	$\|K\|$	%-Eficient	%-Supported
50	50	29.47	41.76
	100	26.42	35.92
	50	30.50	43.77
	200	30.44	37.00
100	50	38.32	37.57
	100	22.28	43.46
	150	37.21	36.99
	200	35.45	43.01
		30.75	40.22

8

A Heuristic Approach for the STPP

As already mentioned, no efficient algorithm to solve TPP up to optimality can be found, unless $\mathcal{P} = \mathcal{NP}$. That is why the literature on TPP is mostly directed towards the development of heuristic or near optimal methods. Section 2.1 has given a review on, among others, heuristic algorithms for obtaining upper bounds of this problem. This chapter is devoted to the development of a heuristic algorithm for our problem. Moreover, this method is not only a specific technique for this particular problem but also a general approach, which could be extended to other similar problems as those described in Section 2.3.

A *feasible solution* σ of the TPP consists in a cycle in G defined by an edge subset $E^\sigma \subset E$ and a vertex subset $V^\sigma \subseteq V$ such that:

i) the depot is visited, i.e., $v_0 \in V^\sigma$,

ii) for each vertex $v \in V^\sigma$ the degree of v is exactly 2,

iii) it is possible to purchase the required demand, i.e.

$$\sum_{v_i \in V^\sigma \cap M_k} q_{ki} \geq d_k \qquad \text{for all } p_k \in K.$$

The set of all feasible solutions will be denoted by Ω. In addition, let us define the *routing cost* of σ as

$$\texttt{travel}(\sigma) := \sum_{e \in E^\sigma} c_e;$$

the *purchasing cost* of product p_k in σ as

$$\texttt{price}(\sigma,k) := \min\left\{ \sum_{v_i \in V^\sigma \cap M_k} z_{ki} b_{ki} : \begin{array}{l} \sum_{v_i \in V^\sigma \cap M_k} z_{ki} = d_k \\ z_{ki} \leq q_{ki} \quad \text{for all } v_i \in V^\sigma \cap M_k \end{array} \right\};$$

and, the *total purchasing cost* of σ as

$$\texttt{price}(\sigma) := \sum_{p_k \in K} \texttt{price}(\sigma,k).$$

The value $f(\sigma) := \texttt{travel}(\sigma) + \texttt{price}(\sigma)$ is called the *total cost* of the feasible solution σ represented by the cycle (V^σ, E^σ) in G. The TPP searches for a feasible solution with minimum total cost, i.e.,

$$\min\{f(\sigma) : \sigma \in \Omega\}.$$

Very special instances of the TPP arise when $d_k = 1$ and $q_{ki} = 1$ for each $p_k \in K$ and $v_i \in M_k$, leading to the previously introduced unrestricted TPP. As mentioned in the introduction, most of the articles in literature are concerned with this unrestricted version. Clearly, a feasible solution σ of the unrestricted TPP is a simple cycle (V^σ, E^σ) in G such that:

i) $v_0 \in V^\sigma$,

ii) $V^\sigma \cap M_k \neq \emptyset$ for all products $p_k \in K$,

and the purchasing cost of a product p_k in σ is simply stated as

$$\texttt{price}(\sigma,k) := \min_{v_i \in V^\sigma \cap M_k} b_{ki}.$$

As mentioned in the introductory chapter, the TPP is $\mathcal{NP}$-hard in the strong sense since it reduces to the TSP when $m = n$ and $|M_k| = 1$ for all p_k. The TPP also reduces to the UFLP when $M_k = M$ for all p_k, $q_{ki} = d_k$ for all $v_i \in M$ and all $p_k \in K$, and $c_e = (f_i + f_j)/2$ for all $e = [i,j] \in E$, with f_i the cost of opening facility v_i ($f_0 := 0$) and b_{ki} the cost of serving customer p_k from facility v_i.

Next sections establishes the main idea of our local-search proposal, which is based on two families of neighbourhoods. A specific procedure to achieve a local minimum is developed for each of them. The first procedure performs an iterative scheme exchanging l consecutive vertices in a given feasible cycle with a set of vertices not belonging to that cycle. The value l is reduced as soon as a local optimum is achieved. The above mentioned procedure is called $l-$`ConsecutiveExchange`. The second procedure inserts as many vertices as possible, whenever each insertion implies a reduction in the objective value. This procedure is called `Insertion`. We next describe more details on each one, starting with a data structure to speed up the evaluation of an insertion/deletion of a market in a partial solution.

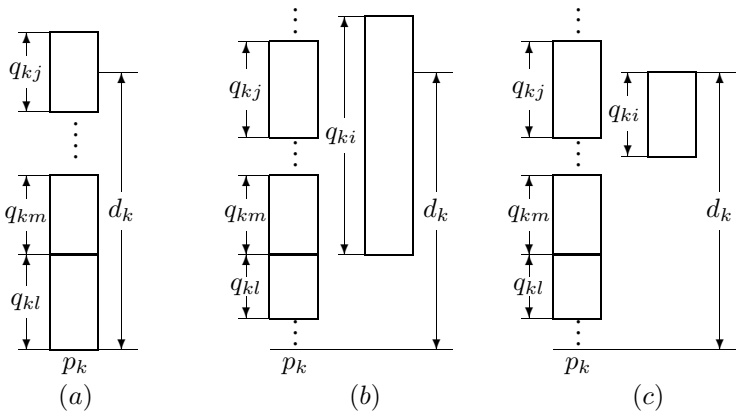


Fig. 8.1 (a): List for product p_k previously sorted such that $b_{kl} \leq b_{km} \leq \ldots \leq b_{kj}$. (b) and (c): Evaluation of a potential insertion of product p_k available at market v_i, where $b_{kl} \leq b_{ki} < b_{km}$.

8.1 DATA STRUCTURE

Whenever a solution is evaluated and an insertion/deletion occurs, the evaluation of a modified solution can be efficiently recomputed using an *ad hoc* data structure. A partial solution σ has an internal representation consisting of a sequence of markets in V^σ and a dynamic array for each product $p_k \in K$. Each component of these arrays corresponds to a market $v_i \in V^\sigma \cap M_k$ and contains the offer q_{ki} as well as the unit price b_{ki}. These items are pre-sorted according to the purchasing cost b_{ki}. Figure 8.1.a illustrates the array for a product p_k, and Figures 8.1.b and 8.1.c help with the idea of inserting a new market v_i selling p_k. According to this data structure, the insertion of a new market $v_i \in M \setminus V^\sigma$ in a partial solution σ would take a time complexity of $O(\log |V^\sigma \cap M_k|)$ for each product $p_k \in K$. Hence, the evaluation of the total purchasing cost would take a time complexity of $O(|V^\sigma||K|)$. In the particular case of the unrestricted TPP, this complexity would be reduced to $O(1)$ for each product $p_k \in K$.

8.2 L-CONSECUTIVE EXCHANGE

Making use of the above data structure, the algorithm l-`ConsecutiveExchange` proceeds by exchanging a set of l consecutive vertices belonging to a feasible cycle σ, with other vertices outside the cycle, in a two stage-procedure. The first stage (called l-`ConsecutiveDrop`) tries to reduce the length of the cycle by removing

l consecutive vertices. The second stage (called `RestoreFeasibility`) tries to restore the feasibility if it is lost in the previous stage.

This idea generalizes the procedure *IMP1* described in Voß [151], in which exactly one single vertex is removed from a feasible cycle, and a number of consecutive insertions is performed as long as an improvement in the objective function is achieved. A similar idea has also been proposed by Keller [92] for the *Orienteering Problem*, where two consecutive vertices are replaced by others if it leads to a better feasible route.

This set of moves defines a very large neighbourhood. Because of this, in order to select a good neighbour, the classical complete enumeration of the neighbourhood is avoided, and the following heuristic procedure is performed. Given an initial solution σ, an starting value l is chosen according to a self-tuning procedure. An iterative mechanism to remove each sequence of l consecutive markets is performed by the procedure l-`ConsecutiveDrop`. Whenever a modified cycle turns out to be infeasible, the procedure `RestoreFeasibility` is called upon. The solution σ is updated if it improves the previous one. However, if no improvement is achieved or restoring feasibility fails, value l is decreased by one unit. This procedure continues iteratively, stopping when $l = 0$. See Figure 8.2 for a pseudocode of the described procedure.

```
Input:  a feasible cycle σ and 1 ≤ l < |V^σ|
Output:  a feasible cycle σ
while l ≥ 1      σ' := l-ConsecutiveDrop(σ,l)
  if σ' is not feasible
    σ' := RestoreFeasibility(σ')
  if f(σ') ≥ f(σ)  or σ' is not feasible
    l := l − 1
  else
    σ := σ'
return σ
```

Fig. 8.2 Procedure l-ConsecutiveExchange.

l-Consecutive Drop

This routine selects l consecutive vertices according to an estimation of the objective function reduction (i.e., the reduction in travel cost after removal and the increase in purchasing cost). For each path $P \subset E^\sigma$ consisting of $l+1$ consecutive edges $\{[s,u_1],[u_1,u_2],\cdots,[u_{l-1},u_l],[u_l,t]\}$ belonging to a feasible cycle σ, let $V(P) := \{u_1,\cdots,u_l\}$ be internal vertices of P. The potential reduction in the travel cost after removing the vertices in $V(P)$ is computed as follows

$$\texttt{TravelReduction}(P) := \sum_{e\in P} c_e - c_{[s,t]}.$$

In addition, the potential increase in the purchasing cost (and referred as Price Increase(P)), as well as the set of non-satisfied products after the removal of ver-

tices in $V(P)$, are also computed. More precisely, those units of product that were purchased in markets in $V(P)$ have to be acquired at markets of $V^\sigma \setminus V(P)$, adding the extra cost to `PriceIncrease`(P); if those units of product cannot be purchased in $V^\sigma \setminus V(P)$ then we do not penalize `PriceIncrease`(P) so as not to discourage selecting a path leading to an infeasible solution. These evaluations are performed on each possible path P with $l+1$ edges, and all of them are ranked according to `TravelReduction`(P)$-$`PriceIncrease`(P). The path with the biggest rank is selected to be removed.

After removing those selected l consecutive vertices, an improvement procedure is applied to reduce the routing cost of the new (and possibly non-feasible) cycle σ. In our implementation this improvement is a specific version of the Lin and Kernighan [106] algorithm, available in Applegate, Bixby, Cook and Chvátal [4]. The Lin-Kernighan algorithm performs a sequence 3-opt edge interchanges, each one followed by a sequence of 2-opt edge interchanges.

Restoring Feasibility

This procedure tries to extend an infeasible cycle σ so as to restore the feasibility. To this end, new markets must be inserted. Denoting by V^* the set of vertices in the previous feasible cycle, the new markets are allowed to be selected from $M \setminus V^*$ to guarantee the generation of different cycles. In our experiments this decision proved to give better results than using $M \setminus V^\sigma$ as candidate markets.

The method proceeds by computing the non-satisfied amount $\bar{d}_k := \max\{0, d_k - \sum_{v_j \in V^\sigma \cap M_k} q_{kj}\}$ for each product $p_k \in K$, and selecting a subset $T \subseteq M_k \setminus V^*$ of markets selling the required amount for each product.

A basic greedy approach finds an initial subset T such as $\sum_{v_i \in T} q_{ki} \geq \bar{d}_k$, for all $p_k \in K$, by choosing the cheapest markets provider in $M_k \setminus V^*$ for each product p_k. However, this procedure admits the following improvement.

Given a non-feasible solution σ, for each vertex $v_i \in M \setminus V^*$ not belonging to σ, two weights are computed. These weights are the routing increase, denoted by $\rho(v_i, \sigma)$, and the purchasing cost reduction, denoted by $\mu(v_i, \sigma)$. After computing these estimations, a subset $T \subseteq M \setminus V^*$ of markets is selected conveniently by solving the following problem:

$$\min_{T \subseteq M \setminus V^*} \left\{ \sum_{v_i \in T} \rho(v_i, \sigma) - \mu(v_i, \sigma) : \sum_{v_i \in T} q_{ki} \geq \bar{d}_k, \text{ for all } p_k \in K \right\}.$$

This combinatorial optimization problem is a generalization of the *set covering* problem, which is known to be $\mathcal{NP}$-hard (see Karp [91]). Despite this, in our computational experience this combinatorial problem turns out to be easy to solve since it is concerned with small-size instances.

The two above mentioned weights try to establish a discernment for the selection of the set T according to the two criteria involved in the objective function.

The routing increase $\rho(v, \sigma)$ of a market v in the current solution σ, which is related to the routing cost, describes how much the routing cost could increase after the insertion of v in σ. In order to calculate ρ, the classical *saving criterion* (see Clarke and Wright [35]) is used.

It should be notice that, in contrast to the case in which the saving is related to a single vertex, there are some cases in which the sum of the individual "saving" costs is not an upper bound of the increase in the travel cost. More precisely, let us denote the sum of the individual routing cost of a vertex set $T \subseteq M \setminus V^*$ by

$$\rho(T, \sigma) := \sum_{v \in T} \rho(v, \sigma),$$

then condition

$$\rho(T, \sigma) \geq \texttt{travel}(\sigma') - \texttt{travel}(\sigma) \tag{8.1}$$

where σ' is the feasible cycle obtained after inserting the vertex set T in σ, does not hold in some cases. Indeed, let us focus on a subset T to be inserted between two vertices s and t belonging to the cycle σ, and let P a path through the vertices of T with extreme vertices s and t. According to the classical saving criterion, each single insertion of a vertex $v \in M \setminus V^*$ is

$$\rho(v, \sigma) := c_{[s,v]} + c_{[v,t]} - c_{[s,t]}.$$

Hence, we have that

$$\texttt{travel}(\sigma') - \texttt{travel}(\sigma) = \sum_{e \in P} c_e - c_{[s,t]} = c_{[s,u_1]} + c_{[u_l,t]} - c_{[s,t]} + \sum_{i=1}^{l-1} c_{[u_i,u_{i+1}]}. \tag{8.2}$$

In addition,

$$\rho(T, \sigma) = c_{[s,u_1]} + c_{[u_l,t]} - c_{[s,t]} + \sum_{l=1}^{l-1} (c_{[s,u_{i+1}]} + c_{[u_i,t]} - c_{[s,t]}). \tag{8.3}$$

Therefore, from (8.2) and (8.3) it follows that condition (8.1) holds when

$$\sum_{i=1}^{l-1} (c_{[s,u_{i+1}]} + c_{[u_i,t]}) \geq \sum_{i=1}^{l-1} (c_{[u_i,u_{i+1}]} + c_{[s,t]}). \tag{8.4}$$

Figure 8.3.a illustrates a simple example evaluating the travel cost for the insertion of a path between s and t. In this particular case the condition (8.4) does not hold, since $c_{[s,u_{i+1}]} + c_{[u_i,t]} < c_{[u_i,u_{i+1}]} + c_{[s,t]}$ for $i = 1, 2, 3$ (see Figure 8.3.b). In spite of this drawback, the criterion of estimating $\texttt{TravelReduction}(P)$ by adding the single savings provided a good behaviour in our experiments, as it is shown in the Section 8.5.

The purchasing cost reduction $\mu(v, \sigma)$ describes the reduction in the purchasing cost after the insertion of a single vertex v in the current solution σ, since a cheaper product may be provided by this new market.

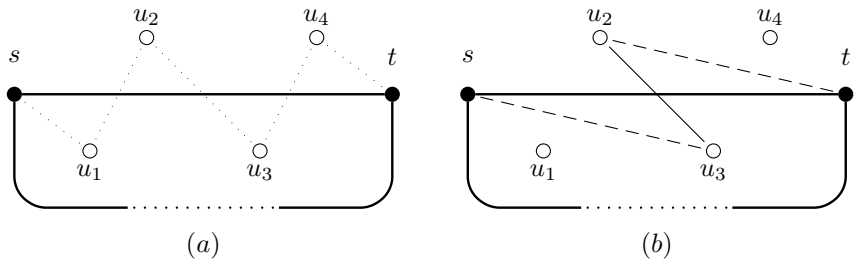


Fig. 8.3 Evaluation of the increase in the travel cost of a set of vertices to be inserted between the vertices s and t.

For each product p_k and each vertex $v_i \in M_k \setminus V^*$, the savings in the purchasing cost `price`(σ') of the new cycle on $V^{\sigma'} := V^\sigma \cup \{v_i\}$ can be efficiently computed by inserting v_i in the dynamic array described in Section 8.1. More precisely, σ' inherits from σ the amount of p_k purchased in markets $v_j \in V^\sigma$ when $b_{kj} \leq b_{ki}$, while the other amount must change taking into account the new market v_i. In particular, it is convenient to purchase

$$r_{ki}(V^\sigma) := \min \left\{ q_{ki}, \max \left\{ 0, d_k - \sum_{v_j \in V^\sigma \cap M_k, b_{kj} \leq b_{ki}} q_{kj} \right\} \right\}$$

units of p_k in the new market v_i. The same amount of units of p_k must be not purchased in the markets of $V^\sigma \cap M_k$ with $b_{kj} > b_{ki}$, and the computation of this adjustment is immediate using the data structure pointed in Section 8.1. The total decrease $\sum_{p_k \in K} [\texttt{price}(\sigma', k) - \texttt{price}(\sigma, k)]$ is considered as the insertion price of v_i in σ, and denoted by $\rho(v_i, \sigma)$.

By using the stated dynamic data structure the theoretical complexity is equivalent to the direct evaluation of the pricing cost, but in practice we notice that the above described procedure reduces the computational effort.

Finally, the procedure inserts the vertices in the selected T, one after the other, by considering the maximum saving criterion. The obtained cycle is re-optimized by using the Lin-Kernighan refinement, already mentioned at the end of the l-Consecutive Drop section.

8.3 INSERTION

The procedure `Insertion` adds a new vertex to the current feasible cycle σ if such insertion implies a reduction in the total cost of σ. In order to perform this procedure both the $\rho(v, \sigma)$ and $\mu(v, \sigma)$ are computed for each vertex $v \in M \setminus V^\sigma$. The vertex maximizing $\rho(v, \sigma) - \mu(v, \sigma)$ such that $\rho(v, \sigma) - \mu(v, \sigma) > 0$ is inserted in σ if such vertex exists.

8.4 THE OVERALL ALGORITHM

An initial solution containing all vertices is built by the well-known nearest-neighbour TSP heuristic, and it is then improved by the previously mentioned Lin-Kernighan procedure. Afterwards, the following iterative scheme is performed. An inner loop computes the value l, as mentioned in Section 8.2, and obtains iteratively better solutions by applying l-`ConsecutiveExchange` and `Insertion`, until no further improvement is achieved. Once the inner loop concludes, a perturbation procedure is carried out in order to augment the current cycle with new vertices.

A *perturbation scheme* (called `Shaking`) controls both the number of added vertices and the number of iterations of the outer loop. In particular, for a given solution σ, each vertex not in V^σ enlarges σ if the travel cost of the augmented solution does not increase by more than φ percent. The percentage φ is iteratively reduced and the procedure stops when no vertex is inserted. The choice of the values for φ was taken based on our computational experiences: initially $\varphi := 35$ and iteratively it is reduced in one unit. See Figure 8.4 for a pseudo-code illustrating the general procedure. The aim of this perturbation scheme is to provide different initial solutions to the above procedure so as to escape from local minimum solutions.

```
Input:  a TPP instance
Output:  a feasible cycle σ
σ := NearestneighborTSP(V)
σ' := σ      repeat
   repeat
      σ' := l-ConsecutiveExchange(σ')
      σ' := Insertion(σ')
   until not improvement
   if f(σ') < f(σ)
   σ := σ'
until Shaking(σ')
return σ
```

Fig. 8.4 Overall algorithm.

8.5 COMPUTATIONAL RESULTS

Our proposal has been tested on the series of randomly generated problems described in Boctor, Laporte and Renaud [22], containing instances for the restricted and unrestricted TPP versions. More precisely, $m + 1$ randomly generated points have been located in the square $[0, 1000] \times [0, 1000]$ according to a uniform distribution and defining routing costs by Euclidean distances. The first location corresponds to the domicile. Each product p_k has been associated with $|M_k|$ randomly selected markets, where $|M_k|$ has been randomly generated in interval $[1, m]$. Product prices b_{ki} are generated in the interval $[1, 500]$ according to a discrete uniform distribution. For the restricted case limit on supplies and demands have also been generated in the follow-

ing way. For each product p_k and each market v_i, q_{ki} has been randomly generated in [1,15] and $d_k := \lceil \lambda \max_{v_i \in M_k} q_{ki} + (1-\lambda) \sum_{v_i \in M_k} q_{ki} \rceil$ for λ=0.1, 0.3, 0.5, 0.7, 0.8, 0.9, 0.95 and 0.99. Notice that, the bigger the λ value is, the shorter the length of its optimal cycle is. For instance, with $\lambda = 0$ becomes a TSP, while $\lambda = 1$ becomes the unrestricted TPP. Five instances were generated for each value of n, m and λ. Therefore, the first family contains 140 cases and the second family contains 960 cases.

Tables 8.1 and 8.2 compare our results with [22] on the unrestricted and restricted TPP instances, respectively. Columns *CAH1*, *CAH2*, *UPH1*, *UPH2* and *CPH* correspond to the different approaches proposed by Boctor, Laporte and Renaud [22]; and columns *LS* correspond to the local search algorithm described in this article. Each column shows the quality of the heuristic solution over the optimum solution obtained by using the exact method described in Laporte, Riera and Salazar [102] (column *%gap*), and the CPU seconds consumed by the heuristic approach on a PC Celeron 500 MHz (column *Sec.*,). Each row contains the average results over the subset of instances solved to optimality by the exact method and grouped according to the value m, n and λ. The column denoted by $\#$ gives the number of instances involved in each row (i.e., the number of instances with a known optimal solution from the exact method described in [102] using a time limit of 2 hours of the Celeron 500 MHz). The column *%Visited* shows the average number of markets involved in an optimal solution computed by the exact algorithm described in [102].

Tables 8.1 and 8.2 clearly show that our approach provides solutions very close to the optimal ones. On the restricted TPP instances (harder than the associated unrestricted ones), the average computational time was close to one minute of the PC Celeron 500 MHz. Solving the set covering subproblems (i.e., calling `Restore Feasibility`) took about 7% of the total computational time. Even if the set covering problem is a hard problem, the low consumed time in our experiments is explained by the small size of the instances of the subproblem we solved. Both quality and consumed time on these small/medium instances were not so dependent on λ and n as on m. The quality is slightly better when λ approximates to 1, which is explained because $\lambda = 0$ is the TSP, while $\lambda = 0.9$ produces instances involving both the optimal routing and selection of markets. This conclusion coincides with similar studies on other routing-location problems (see, e.g., Keller [92]).

Regarding both the quality of the solutions and the computational effort, we observe that our local search proposal improves on the approaches proposed in Boctor, Laporte and Renaud [22].

The quality of our heuristic approximation for selecting the vertices to be added in the procedure `Restore Feasibility` is measured in Table 8.3. It shows the maximum and average percentage of the difference between this heuristic and the exact choice. More precisely, procedure `Restore Feasibility`, which selects a set of vertices based on estimations of the reduction in the objective function, has also been solved optimally using the branch-and-cut code described in [102]. It is observed from this table that `Restore Feasibility` selects and inserts new markets with a gap of 2.5% of error. In spite of this gap the procedure proved to be effective in our experiments.

In order choose the initial value of parameter l, as well as to prove that the l-`Consecutive Exchange` has better performance than the classical approach in which exactly one drop and several adding moves are performed, the following experiment have been carried out. For each benchmark restricted TPP instance with $\lambda \in \{0.8, 0.9, 0.95, 0.99\}$, the procedure l-`Consecutive Exchange` has been executed with different initial values of l. Notice that this procedure starts from a TSP solution, iteratively removes sequences of consecutive vertices and inserts others to restore feasibility, as described in Section 8.2. The objective value of the generated TPP solutions and the computational time have been normalized in the rank $[0, 1]$ with respect to the minimum and maximum values obtained varying the parameter l. The average (normalized) values are computed for each l and Figure 8.5 plots them, those points associated to the objective function with boxes and those points associated to the consumed time with circles. For very small values of l, the computational time is close to be proportional to the number of iterations thus, e.g., it is bigger for $l = 1$ than for $l = 3$. Moreover, big l values imply strong modifications of the cycle, hence the gap and the computational effort increase with l. The best results are obtained for l between 2 and 25, thus inspiring our proposal for managing l in the procedure.

We have also experimented with the described approach on (restricted and unrestricted) TPP instances involving up to $m = 350$ and $n = 200$, obtaining similar performances when comparing quality of the heuristic solution with the LP-relaxation of the model in [102]. The average gap was close to 0.5% while the computing time was never more than one minute.

Table 8.1 Average computational results, unrestricted (optimal) instances.

				Boctor, Laporte & Renaud [22]									
				CAH1		*CAH2*		*UPH1*		*UPH2*		*LS*	
		%Visit.	*#*	*%gap*	*Sec.*	*%gap*	*Sec.*	*%gap*	*Sec.*	*%gap*	*Sec.*	*%gap*	*Sec.*
m	50	26	20	3.05	1	0.53	12	0.33	12	0.26	12	0.07	3
	100	16	20	1.89	9	0.64	104	0.28	60	0.30	50	0.14	10
	150	11	20	2.12	20	0.97	263	0.59	213	0.71	184	0.03	14
	200	9	18	2.59	34	0.95	513	0.65	261	0.53	324	0.32	19
	250	7	11	2.66	38	1.04	649	0.47	233	0.63	253	0.06	25
n	50	8	25	0.78	6	0.43	111	0.34	17	0.35	16	0.07	5
	100	13	22	2.03	17	0.73	216	0.49	74	0.59	71	0.24	13
	150	16	22	3.14	23	1.05	371	0.48	191	0.45	225	0.10	20
	200	17	20	4.20	31	1.09	414	0.56	336	0.50	331	0.08	21

Table 8.2 Average computational results, restricted (optimal) instances.

				Boctor, Laporte & Renaud [22]							
				CAH1		*CAH2*		*CPH*		*LS*	
		%Visit.	*#*	*%gap*	*Sec.*	*%gap*	*Sec.*	*%gap*	*Sec.*	*%gap*	*Sec.*
m	50	73	140	1.42	5	0.49	12	0.41	10	0.15	5
	100	71	135	2.00	33	1.06	86	0.97	75	0.50	26
	150	73	119	2.55	113	1.81	361	1.39	225	0.43	52
	200	69	46	4.61	156	2.81	1465	1.32	510	1.15	100
n	50	63	131	3.32	34	1.72	461	1.11	162	0.72	43
	100	71	117	1.83	65	1.27	281	0.92	154	0.56	38
	150	77	94	1.51	58	0.87	141	0.78	113	0.42	32
	200	80	98	1.96	83	1.02	161	0.89	118	0.29	32
λ	0.1	100	70	0.16	81	0.10	190	0.15	26	0.00	8
	0.5	99	68	0.50	96	0.32	393	0.48	48	0.20	25
	0.7	88	66	1.65	71	0.77	387	0.76	129	0.10	29
	0.8	85	68	3.02	72	1.57	526	0.99	290	0.57	62
	0.9	59	64	5.24	49	3.62	262	1.88	329	1.43	86
	0.95	39	41	3.09	11	1.95	54	1.13	85	0.68	26
	0.99	19	63	2.51	5	0.88	27	1.36	51	0.31	19

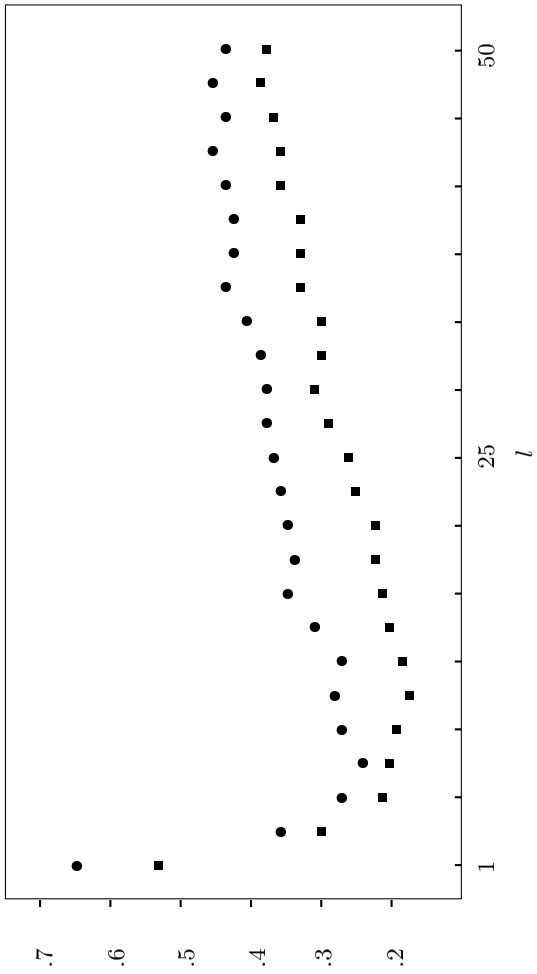


Fig. 8.5 Deviation of the objective function and CPU time from their minimum values for different values of l.

Table 8.3 Average and maximum value of the percentage of the difference between the optimal and heuristic estimation in procedure `RestoreFeasibility`

	n				m		
	50	100	150	200	50	100	150
Avg.	2.68	2.23	2.88	2.69	2.37	2.87	2.77
Max.	13.20	11.21	17.33	11.88	13.20	17.33	14.87

Conclusions

In this dissertation we have studied a problem arising from a job scheduling real world application: the Traveling Purchaser Problem (TPP), which is a generalization of the well known Traveling Salesman Problem. No more than eigth articles have addressed the TPP so far, and this work is the first serious attempt to solve the problem up to optimality.

After making a deep study of the *state of the art* we have realize that the attempts of developing exact algorithms had achieved poor results. Therefore, the purpose of our first approach was to perform two exact algorithms for both cases the symmetric and asymmetric TPP. To this end, new linear integer models were described in Chapter 3, as well as new valid inequalities to strength the LP-relaxation.

In addition, in order to avoid those redundant inequalities and to perform a more efficient branch-and-cut algorithm, a polyhedral study on the TPP has been carried out in Chapter 4, proving that some of the valid inequalities described in Chapter 3 defined facet for the polytope associated to the TPP.

Specific features of the both branch-and-cut algorithms for the symmetric and asymmetric case are described in Chapters 5 and 6 respectively. New separation algorithms for facets and valid inequalities described in previous chapters are proposed in these two chapters, whose efficiency has been tested by mean a computational experience. The efficiency of these algorithms as a heuristic approach has been also tested. This computational experience has been performed on all instances proposed in the previous works on the TPP and on our own random instances as well. Chapter 6 also includes a computational experience on a transformation of the asymmetric version into the symmetric one.

Since no previous work had not approached the bicriterion TPP, the next chapter start on this topic. Chapter 7 describes an algorithm which computes the set of non-dominated points and the set of extreme points of the efficient frontier. This algorithm is the first based on branch-and-cut for this purposed. The procedure combines the classical methods in bicriterion programming with a branch-and-cut algorithm to solve single-objective subproblems. The key point is to use a common cut-pool structure to save cuts separated during the solution of a subproblem that can help in the solution of other forthcoming subproblems, and thus reducing the computational effort. The algorithm has been implemented and tested on three families of test-bed instances from literature proving the good performance of the proposal. A similar idea could be extended to other bicriterion problems where there is a branch-and-cut algorithm available for the single-objective problem.

Most of the research about TPP has been directed towards the development of heuristic approaches. However, a new competitive heuristic approach has been develop, and its efficiency is measured in Chapter 8. This algorithm, which is based on local search, is not only valid for the classical TPP but also for the restricted TPP. The basic idea is the definition of an exponential neighbourhood explored by a heuristic procedure. Given a solution, each neighbour is obtained by removing a path of consecutive vertices and by inserting a new one so as to restore feasibility. The performance is favourable compared with other tabu search approaches recently proposed on Euclidean travel cost instances in literature. These new neighbourhoods adapt easily to similar problems in which a subcycle has to be obtained minimizing the sum of two objectives, and therefore, it could be an interesting contribution to other local search approaches.

References

1. A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computers Algorithms*. Addison Wesley, Reading, Massachusett, 1974.

2. S. Anily and R. Hassin. The swapping problem. *Networks*, 22:419–433, 1992.

3. S. Anily and G. Mosheiov. The traveling salesman problem with delivery and backhauls. *Operations Research Letters*, 16:11–18, 1994.

4. D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Concorde: a code for solving traveling salesman problem. http://www.kerk.caam.rice.edu/concorde.html, 1997.

5. Esther M. Arkin, Yi-Jen Chiang, Joseph S. B. Mitchell, Steven S. Skiena, and Tae-Cheon Yang. On the maximum scatter tsp. *SIAM Journal on Computing*, 29:515–544, 1999.

6. Norbert Ascheuer, Matteo Fischetti, and Martin Götschel. Solving the asymmetric salesman problem with tiem windows by branch-and-cut. *Mathematical Programming*, 90:475–506, 2001.

7. A. Bachem and M. Grötschel. New aspects of polyhedral theory. In Korte [93], pages 51–106.

8. E. Balas. The prize collecting traveling salesman problem. *Networks*, 19:621–636, 1989.

9. E. Balas. The prize collecting traveling salesman problem: II. polyhedral resutls. *Networks*, 17:1001–1018, 1995.

10. E. Balas. The prize collecting traveling salesman problem and its applications. In Guting and Punnen [83], chapter 14, pages 663–696.

11. E. Balas and M. Fischetti. A lifting procedure for the asymmetric traveling salesman polytope and a large new class of facets. *Mathematical Programming*, 58:325–352, 1993.

12. E. Balas and S. M. Ng. On the set covering polytope: I. All the facets with coefficients in $\{0, 1, 2\}$. *Mathematical Programming*, 43:57–69, 1989.

13. E. Balas and M. Oosten. On the cycle polytope of a directed graph. *Networks*, 36:34–46, 2000.

14. E. Balas and N. Simonetti. Linear time dynamic programming algorithms for some classes of restricted tsp'. Technical Report MSRR-617, Carnegie Mellon University, Pittsburgh, USA, 1996.

15. E. Balas and P. Toth. Branch and bound methods. In Lawler et al. [103], chapter 10.

16. M. O. Ball, T. L. Magnanti, C. L. Monma, and G.L. Nemhauser, editors. *Handbooks in Operations Research and Management Science: Network Models*. North–Holland, Amsterdam, 1995.

17. J.M. Basart and L. Huguet. An approximate algorithm for the tsp. *Information Processing Letters*, 31:77–81, 1989.

18. P. Bauer. The circuit polytope: Facets. *Mathematics of Operations Research*, 22:110–145, 1997.

19. M. Bellmore and G. L. Nemhauser. The traveling salesman problem: a survey. *Operations Research*, 16:538–558, 1968.

20. C. Berge. *Graphs and hypergraphs*. North-Holland, 1973.

21. L. Bianco, A. Mingozzi, and S. Ricciardelli. Dynamic programming strategies and reduction techniques for the traveling salesman problem with time windows and precendece contraints. *Operations Research*, 45:365–377, 1997.

22. F. F. Boctor, G. Laporte, and J. Renaud. Heuristics for the traveling purchaser problem. Internal report, po2001-10-x crt-2001-10, University of Montreal, 2001.

23. J. A. Bondy and U. S. R. Murty. *Graph theory with applications*. University Press, Belfast, 1976.

24. M. Bourgeois, G. Laporte, and F. Semet. Heuristics for the black and white traveling salesman problem. Technical Report CRT-99-33, Centre for Research on Transportation, Montreal, 1999.

25. R. M. Burstall. A heuristic method for a job sequencing problem. *Operational Research Quarterly*, 17:291–304, 1966.

26. J. A. Buzacott and S. K. Dutta. Sequencing many jobs on a multipurpose facility. *Naval Research Logistics Quarterly*, 18:75–82, 1971.

27. A. Caprara and M. Fischetti. Branch and cuts algorithms. In Dell'Amico et al. [44], pages 45–64.

28. G. Carpaneto, S. Martello, and P. Toth. An algorithm for the bottleneck traveling salesman problem. *Operations Research*, 27:380–389, 1984.

29. I. M. Chao, B. L. Golden, and E. A. Wasil. A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88:475–489, 1996.

30. N. Chistofides. *Graph Theory. An algorithmic approach*. Academic Press, New York, 1975.

31. N. Chistofides. Vehicle routing. In Lawler et al. [103], pages 431–448.

32. N. Chistofides, A. Mingozzi, and P. Toth. The vehicle routing problem. In Chistofides et al. [33], pages 315–338.

33. N. Chistofides, A. Mingozzi, P. Toth, and C. Sandi, editors. *Combinatorial Optimization*. Wiley, Chichester, 1979.

34. V. Chvátal. Edmonds polytope and weakly Hammiltonian graphs. *Mathematical Programming*, 5:29–40, 1973.

35. G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a volume of delivery points. *Operations Research*, 12:568–581, 1964.

36. W. Cook, L. Lovász, and P. Seymour, editors. *Combinatorial Optimization*. DIMACS Series in Discrete Mathematical and Theoretical Computer Science American Mathematical Society, 1995.

37. C. Coullard and W. R. Pulleyblank. On clyce cones and polyhedra. *Linear Algebra Applied*, 114/115:613–640, 1989.

38. H. Crowder, E. L. Johnson, and M. W. Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31:803–834, 1983.

39. J. R. Current. *Multiobjective Desing of Transportation Networks*. PhD thesis, Deparment of Geography and Enviromental Engineering, The Johns Hopkins University, 1981.

40. J. R. Current, C. Revelle, and J. Cohon. The shortest covering path problem: an application of locational constraints to network design. *Journal of Regional Science*, 24:161–183, 1984.

41. J. R. Current and D. A. Schilling. The covering salesman problem. *Transportation Science*, 23:208–213, 1989.

42. G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson. Solution of large-scale traveling salesman problem. *Operations Research*, 2:393–410, 1954.

43. G. B. Dantzig and R. H. Ramser. The truck dispatching problem. *Management Science*, 6:80–91, 1959.

44. M. Dell'Amico, F. Maffioli, and S. Martello, editors. *Annotated bibliographies in combinatorial optimization*. Wiley, 1997.

45. M. Desrochers, J. K. Lenstra, M. W. P. Savelsbergh, and F. Soumis. Vehicle routing with time windows: Optimization and approximation. In Golden and Assad [72], pages 64–84.

46. J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis. Time constrained routing and scheduling. In Ball et al. [16], chapter 2, pages 35–139.

47. Y. Dumas, J. Desrosiers, E. Gelinas, and M. M. Solomon. An optimal algorithm for the traveling salesman problem with time windows. *Operations Research*, 42:367–371, 1995.

48. J. Edmonds. Maximum matching and a polyhedron with 0,1 vertices. *J. Res. Nat. Bur. Standards*, 69B, 65.

49. M. Ehrgott. *Multicriteria Optimization*. Lecture Notes in Economics and Mathematical Systems. Springer,Verlag, Berlin, 2000.

50. M. Ehrgott and X. Gandibleux. An Annotated Bibliography of Multi-objective Combinatorial Optimization. Technical Report 62/2000, Fachbereich Mathematik, Universitat Kaiserslutern, 2000.

51. M. Fischetti. Facets of the asymmetric traveling salesman polytope. *Mathematics of Operations Research*, 16:42–56, 1991.

52. M. Fischetti, A. Lodi, and P. Toth. Exact methods for the asymmetric traveling salesman problem. In Guting and Punnen [83], chapter 4, pages 169–206.

53. M. Fischetti, J. J. Salazar, and P. Toth. The symmetric generalized travelling salesman polytope. *Networks*, 26:113–123, 1995.

54. M. Fischetti, J. J. Salazar, and P. Toth. A branch-and-cut algorithms for the symmetric generalized traveling salesman problem. *Operations Research*, 45:378–394, 1997.

55. M. Fischetti, J. J. Salazar, and P. Toth. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10:133–148, 1998.

56. M. Fischetti, J. J. Salazar, and P. Toth. The generalized traveling salesman problem and orieteering problems. In Guting and Punnen [83], chapter 13, pages 609–662.

57. M. Fischetti and P. Toth. An additive approach for the optimal solution of hte prize collecting traveling salesman problem. In Golden and Assad [72], pages 319–343.

58. M. Fischetti and P. Toth. A polyhedral approach to the asymmetric traveling salesman problem. *Management Science*, 43:1520–1536, 1997.

59. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

60. R. S. Garfinkel and K. C. Gilbert. The bottleneck traveling salesman problem: Algorithms and probabilistic analysis. *Journal of the ACM*, 25:435–448, 1978.

61. T. J. Gaskell. Bases for vehicle fleet scheduling. *Operations Research Quarterly*, 18:281–294, 1967.

62. M. Gendreau, A. Hertz, and G. Laporte. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 1992.

63. M. Gendreau, M. Labbé, and G. Laporte. Efficient heuristics for the design of ring networks. *Telecommunication systems*, 4:177–188, 1995.

64. M. Gendreau, G. Laporte, and F. Semet. A branch-and-cut algorithm for the undirected selective traveling salesman problem. Working paper, Centre de recherche sur les transports, Montréal, 1995.

65. M. Gendreau, G. Laporte, and F. Semet. The covering tour problem. *Operations Research*, 45:568–576, 1997.

66. M. Gendreau, G. Laporte, and D. Vigo. Heuristics for the traveling salesman problem with pickup and delivery. *Computers & Operations Research*, 26:699–714, 1999.

67. G. Ghiani, G. Laporte, and F. Semet. The black and white traveling salesman problem. Technical Report CRT-99-47, Centre for Research on Transportation, Montreal, 1999.

68. P. C. Gilmore and R. E. Gomory. Sequencing a one state-variable machine. a solvable case of the traveling salesman problem. *Operations Research*, 12:655–679, 1964.

69. A. Goicoechea, L. Duckstein, and M. M. Fogel. Multiobjective programming in watershed management: A case study of the charleston watershed. *Water Resources Research*, 12:1085–1092, 1976.

70. A. Goicoechea, D. R. Hansen, and L. Duckstein. *Multiobjective Decision Analysis with Engineering and Business Applications*. John Wiley & Sons, New York, 1982.

71. A. Goldberg and R. Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM*, pages 921–940, 1988.

72. B. L. Golden and A. A. Assad, editors. *Vehicle Routing: Methods and Studies*. North–Holland, Amsterdam, 1988.

73. B. L. Golden, L. Levy, and R. Dahl. Two generalizations of the traveling salesman problem. *Omega*, 9:439–445, 1981.

74. B. L. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistic*, 34:359–366, 1987.

75. B. L. Golden, Q. Wang, and L. Liu. Multifaceted heuristic for the orienteering problem. *Naval Research Logistic*, 35:359–366, 1988.

76. L. Gouvieia and J.M. Pires. Models for a steiner ring network design problem with revenues. Technical report, Facultade de Ciencias da Universidade de Lisboa, 1998.

77. M. Grötschel. Polyedrische charackterisierungen kombinatorischer optimierungsprobleme. Hain, Meisenheim an Glan, 1977.

78. M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer–Verlag, Berlin, 1988.

79. M. Grötschel and M. W. Padberg. On the symmetric travelling salesman problem I: inequalities. *Mathematical Programming*, 16:265–280, 1979.

80. M. Grötschel and M. W. Padberg. On the symmetric travelling salesman problem II: lifting theorems and facets. *Mathematical Programming*, 16:281–302, 1979.

81. M. Grötschel and M. W. Padberg. Polyhedral theory. In Lawler et al. [103], chapter 8, pages 251–305.

82. B. Grünbaum. *Convex Polytopes*. Wiley, London, 1967.

83. G. Guting and A. Punnen, editors. *The Traveling Salesman Problem and its variations*. Kluwer Academic Publishers, Dordrecht, 2002.

84. H. H. Hahn, editor. *Tourenplanung bei der Abfallbeseitigung*. Schmidt, Bielefeld, 1977.

85. H. Hernández-Pérez and J. J. Salazar-González. A branch-and-cut algorithm for the pickup-and-delivery travelling salesman problem. Working paper, DEIOC, Universidad de La Laguna, 2000.

86. R. Jonker and T. Volgenant. Transforming asymmetric into symmetric traveling salesman problems. *Operations Research Letters*, 2:161–163, 1983.

87. M. Jünger, G. Reinelt, and G. Rinaldi. The traveling salesman problem. In Ball et al. [16], chapter 4.

88. M. Jünger, G. Reinelt, and S. Thienel. Practical problem solving with cutting plane algorithms in combinatorial optimization. In Cook et al. [36], pages 111–152.

89. M. Jünger and S. Thienel. Introduction to ABACUS —a branch-and-cut system. *Operations Research Letters*, 22:83–95, 1998.

90. R. L. Karg and G. L. Thompson. A heuristic approach to solving traveling salesman problem. *Management Science*, 10:225–248, 1964.

91. R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

92. C. P. Keller. Algorithms to solve the orienteering problem: A comparison. *European Journal of Operational Research*, 41:224–231, 1989.

93. B. Korte, editor. *Modern Applied Mathematics, Optimization and Operations Research*. North-Holland, Amsterdam, 1982.

94. M. Labbé, G. Laporte, I. Rodríguez, and J. J. Salazar. The median cycle problem. Working paper, DEIOC, Universidad de La Laguna, 2001.

95. M. Labbé, G. Laporte, I. Rodríguez, and J. J. Salazar. The ring star problem: Polyhedral analysis and exact algorithm. Working paper, DEIOC, Universidad de La Laguna, 2001.

96. G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59:345–358, 1992.

97. G. Laporte and S. Martello. The selective traveling salesman problem. *Discrete Applied Mathetics*, 26:193–207, 1990.

98. G. Laporte, H. Mercure, and Y. Norbert. Generalize travelling salesman through n sets of nodes: The asymmetrical case. *Discrete Applied Mathematic*, 18:185–197, 1987.

99. G. Laporte and Y. Norbert. Finding the shortest cycle through k specified nodes. *Congressus Numerantium*, 48:155–167, 1983.

100. G. Laporte and Y. Norbert. Generalize travelling salesman through n sets of nodes: An integer programming approach. *INFOR*, 21:61–75, 1983.

101. G. Laporte and Y. Norbert. Exact algorithms for the vehicle routing problem. In Martello et al. [108], pages 147–184.

102. G. Laporte, J. Riera-Ledesma, and J. J. Salazar-González. A branch-and-cut algorithm for the undirected traveling purchaser problem. Working paper, DEIOC, Universidad de La Laguna, 2001.

103. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors. *The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization*. Wyley, Chichester, 1985.

104. Y. Lee and J. Sanchez S. Y. Chiu. A branch and cut algorithm for the steiner ring star problem. *IJMS*, 4:21–34, 1999.

105. A. C. Leifer and M. B. Rosenwein. Strong linear programming relaxations for the orienteering problem. *European Journal of Operational Research*, 73:517–523, 1994.

106. S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21:498–516, 1973.

107. Z. A. Lomnicki. Job scheduling. *Operational Research Quarterly*, 17:314–316, 1966.

108. S. Martello, G. Laporte, M. Minoux, and C. Ribeiro, editors. *Surveys in Combinatorial Optimization*. North–Holland, Amsterdam, 1987.

109. S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, Chichester, 1990.

110. K. Mehlhorn. *Data Structures and Algorithms*, volume 1. Springer Publishing Company, 1984.

111. G. Mosheiov. The traveling salesman problem with pickup and delivery. *European Journal of Operational Research*, 79:299–310, 1994.

112. D. Naddef. Polyhedral theory and branch-and-cut algorithms for the symmetric tsp. In Guting and Punnen [83], chapter 2, pages 29–116.

113. G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, editors. *Handbooks in Operations Research and Management Science*, volume 1. North–Holland, 1989.

114. G. L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1999.

115. C. E. Noon and J. C. Bean. A lagrangian based approach for asymmetric generalized traveling salesman problem. *Operation Research*, 39:623–632, 1991.

116. H. L. Ong. Approximate algorithms for the traveling purchaser problem. *Operations Research Letters*, 1:201–205, 1982.

117. M. Padberg and G. Rinaldi. An efficient algorithm for the minimum capacity cut problem. *Mathematical Programming*, 47:19–36, 1990.

118. M. Padberg and G. Rinaldi. Facet identification for the symmetric traveling salesman polytope. *Mathematical Programming*, 47:219–257, 1990.

119. M. Padberg and G. Rinaldi. A branch and cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33:60–100, 1991.

120. M. W. Padberg. A note on zero-one programming. *Operations Research*, 23:833–837, 1975.

121. M. W. Padberg and M. R. Rao. Odd minimum cut-sets and b-matchings. *Mathematics of Operations Research*, 1982.

122. H. Paessens and H. K. Weuthen. Tourenplanung in städtischen straßennetzen mit einem heuristischen verfahren. In Hahn [84].

123. W. L. Pearn and R. C. Chien. Improved solutions for the traveling purchaser problem. *Computers & Operations Research*, 25:879–885, 1998.

124. K. Penavic. Optimal firing sequences for cat scans. Technical report, Department of Applied Mathematics, SUNY Stony Brook, NY, 1994.

125. W. R. Pulleyblank. Polyhedral combinatorics, optimization. In Nemhauser et al. [113], pages 371–446.

126. T. Ramesh. Traveling purchaser problem. *Opsearch*, 18:78–91, 1981.

127. T. Ramesh, Y. S. Yoon, and M. H. Karwan. An optimal algorithm for the orienteering tour problem. *ORSA Journal on Computing*, 4:155–165, 1992.

128. G. Reinelt. *The Traveling Salesman Problem: Computational Solutions for TSP*. Springer-Verlag, Berlin, 1994.

129. J. Riera-Ledesma and J. J. Salazar-González. A heuristic approach for the traveling purchaser problem. Working paper, DEIOC, Universidad de La Laguna, 2001.

130. J. Riera-Ledesma and J. J. Salazar-González. The biobjective travelling purchaser problem. Working paper, DEIOC, Universidad de La Laguna, 2002.

131. J. Riera-Ledesma and J. J. Salazar-González. Solving the directed traveling purchaser problem. Working paper, DEIOC, Universidad de La Laguna, 2002.

132. R. T. Rockafellar. *Convex Analysis*. Princeton University Press, New Jersey, 1970.

133. I. Rodríguez. *Cycle Location Problems*. PhD thesis, DEIOC, Universidad de La Laguna, 2000.

134. D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis. Approximate algorithms for the travelling salesperson problem. *SIAM Journal on Computing*, 6(3), 1977.

135. J. J. Salazar. *Algoritmi per il problema del Commesso Viaggiatore Generalizzato*. PhD thesis, Royal Spanish College in Bologna, 1992.

136. J. J. Salazar-González. On the cycle polytope of an undirected graph. Working paper, DEIOC, Universidad de La Laguna, 1994.

137. J. E. Savage. *The Complexity of Computing*. John Wiley, New York, 1976.

138. A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1986.

139. M. M. Sepehri. *The symmetric generalized travelling salesman problem*. PhD thesis, University of Tennessee, Knoxville, 1991.

140. S. Sergeev and A. B. Chernyshenko. Algorithms for the minmax problem of the traveling salesman ii: Dual approach. *Automation and Remote Control*, 56:1155–1168, 1995.

141. K. N. Singh and D. L. van Oudheusden. A branch and bound algorithm for the traveling purchaser problem. *European Journal of Operational Research*, 97:571–579, 1997.

142. R. M. Soland. Multicriteria optimization: A general characterization of efficient solutions. *Decision Sciences*, 10:27–38, 1979.

143. R. E. Steuer. *Multiple criteria optimization: theory, computation, and application*. Wiley, 1986.

144. J. Stoer and C. Witzgall. *Convexity and Optimization in Finite Dimensions I*. Springer, Berlin, 1970.

145. S. Thienel. *ABACUS: A Branch and Cut System*. PhD thesis, Universität zu Köln, 1995.

146. V. T'kindt, J-C. Billaut, and C. Proust. An interactive algorithm to solve bicriteria scheduling problems on unrelated parallel machines. *European Journal of Operational Research*, 135:42–49, 2001.

147. T. Tsiligirides. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, 35:797–809, 1984.

148. J. Tsitsiklis. Special cases of traveling salesman and repairman problems with time windows. *Networks*, 22:263–282, 1992.

149. A. M. Turing. On computable volume, with an application to the entscheidungproblem. In *Proceedings of the London Mathematical Society*, volume 42, pages 230–265, 1936. [correction: 43 (1937) 544–546].

150. M. Visée, J. Teghem, M. Pirlot, and E. Ulungu. Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. *Journal of Global Optimization*, 12:139–155, 1998.

151. S. Voß. Dynamic tabu search strategies for the traveling purchaser problem. *Annals of Operations Research*, 63:253–275, 1996.

152. L. A. Wolsey. *Integer Programming*. John Wiley & Sons, Inc., 1998.

153. J. Xu, S. Y. Chiu, and F. Glover. Optimizing a ring-based private line telecommunication network using tabu search. *Management Science*, 45:330–345, 1999.