The Service Allocation Problem at the Gioia Tauro Maritime Terminal

Jean-François Cordeau[*], Manlio Gaudioso[†]
Gilbert Laporte[*] and Luigi Moccia[*†]

May 19, 2005

Abstract

The *Service Allocation Problem* (SAP) is a tactical problem arising in the yard management of a container transshipment terminal. The objective is the minimization of the container rehandling operations inside the yard. This study of the SAP was undertaken for the Gioia Tauro port which is located in Italy and is the main hub terminal for container traffic in the Mediterranean Sea. The SAP can be formulated as a *Generalized Quadratic Assignment Problem* (GQAP) with side constraints. Two mixed integer linear programming formulations are presented. The first one exploits characteristics of the yard layout at Gioia Tauro where the berth and the corresponding yard positions extend along a line. The second formulation is an adaptation of a linearization for the GQAP. In both cases only small instances can be solved optimally. An evolutionary heuristic was therefore developed. For small size instances the heuristic always yields optimal solutions. For larger sizes it is always better than a truncated branch-and-bound algorithm applied to the exact formulations.

Keywords: *yard management; maritime container terminal; memetic heuristic; genetic algorithm; tabu search.*

[*]Canada Research Chair in Distribution Management, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7
[†]Dipartimento di Elettronica, Informatica e Sistemistica, Università della Calabria, 87036 Rende (CS) - Italy.

1 Introduction

In maritime transportation the use of containers for general cargo has increased steadily in the last 20 years. Containers are large metal boxes made in standard dimensions and measured in multiples of twenty feet called "twenty foot equivalent units" (TEUs). In 2003 the production of containers reached two million TEUs, with China being responsible for more than 90 percent of this output. The world container port throughput for 2002 reached 266.3 million TEUs, an increase of 22.5 million TEUs from the level of 243.8 million TEUs reached in 2001. For statistics on maritime transport we refer the reader to UNCTAD (2004). In this industry the hub and spoke arrangement is widely adopted. Deep sea vessels, also called mother vessels, operate between a limited number of transshipment terminals (hubs). Smaller vessels (feeders) link the hubs with the other ports (spokes). This network topology results in the consolidation of capacity along the routes linking the hubs and in their growth. In recent years, mother vessels have strongly increased in size attaining up to 8000 TEUs and larger sizes are planned. Transshipment ports are large intermodal platforms, and a limited number of them handle an important share of the world traffic. Thus, in 2002 the first 20 container ports handled 48% of the total traffic. Ultra-large container vessels cut down transport cost. However, hub ports are forced to invest heavily to accommodate these ships by deepening and widening channels and constructing new berthing services of sufficient depth and length. These trends require a continuous improvement in managerial practices at transshipment terminals which can be viewed as large material handling systems. Advanced communication and information technologies already exist and the next step will be the introduction of customized optimization techniques. Customization is a key factor since maritime terminals differ from each other in their layouts and material handling equipments. The need for an optimal management of logistics activities at modern container terminals is well recognized (Daganzo, 1990; De Castilho and Daganzo, 1993; Taleb-Ibrahimi et al., 1993). For a recent overview and classification of the various equipments and decision problems in such systems, see Vis and Koster (2003) and Steenken et al. (2004). A review of operational research issues in maritime logistics, focused on ship routing and scheduling, is presented in Christiansen et al. (2004).

In a modern container terminal the most important problem is the coordination between the storage of the containers into the yard, the loading and unloading of the vessels, and the transfer from and to the gates for land transportation (e.g., by rail and truck). Most operations have their origin or destination in the yard. Yard management is a complex process involving several

decisional areas like berth allocation for the vessels, operator roster scheduling, and container transfer. In the latter case the problem is to manage an heterogeneous fleet of vehicles to transfer containers between the quay, the gates and the yard, and to perform relocations inside the yard. The yard layout and technology are strategic decisions that affect the nature of the management problems. Container terminals in the Asia-Pacific region rely on the "Indirect Transfer System" (ITS) in which the containers are stacked in compact sections and a dedicated gantry crane, called transtainer, moves them from and to the vehicles which link the yard with the quay and the gates. The ITS minimizes yard surface requirements. European and North-American container terminals are based upon the "Direct Transfer System" (DTS) in which the vehicles that transfer the containers between the yard, the quay and the gates are also able to move the containers from and to the slots assigned into the yard. The DTS requires a larger surface because dedicated lanes in the yard are necessary for the vehicles in order to have access to the slot positions. In our study we consider the DTS in use at the port of Gioia Tauro, located in southern Italy.

1.1 The Gioia Tauro Maritime Terminal

This study was initiated at the request of the Gioia Tauro Maritime Terminal, a port mainly devoted to transshipment activities involving mother vessels and feeders operating in a hub and spoke manner. Nearly fifty spoke ports are linked to Gioia Tauro. In just a few years, Gioia Tauro has become the largest transshipment port on the Mediterranean Sea. This achievement is impressive since the port only opened in 1995, and no infrastructures existed in 1993. The traffic handled in 2004 consists of 3.26 million TEUs. The harbor entrance is 250 meters wide and the water depth is 18 meters. The quay length is 3100 meters. The channel along the quay presents a multi-water depth configuration, ranging from 13.5 to 15.5 meters. There are 23 available quay cranes, 18 of which are mounted on rail, while the others roll on tires. A fleet of 75 vehicles, called straddle carriers, transfer the containers between the quay cranes and the yard. These vehicles are capable of transporting up to two containers at a time and can insert containers directly into the right yard slot. Straddle carriers are usually assigned to the transport of full containers over relatively short distances (less than 500 meters), while different vehicles are used for longer transfers.

The yard surface occupies 1.1 million square meters and can store nearly 59,000 TEUs (1,100 of which can be refrigerated). The yard contains four main areas. Three of these, called A, B and C, are parallel to the quay, with area A being the closest to the sea and area C being the most distant.

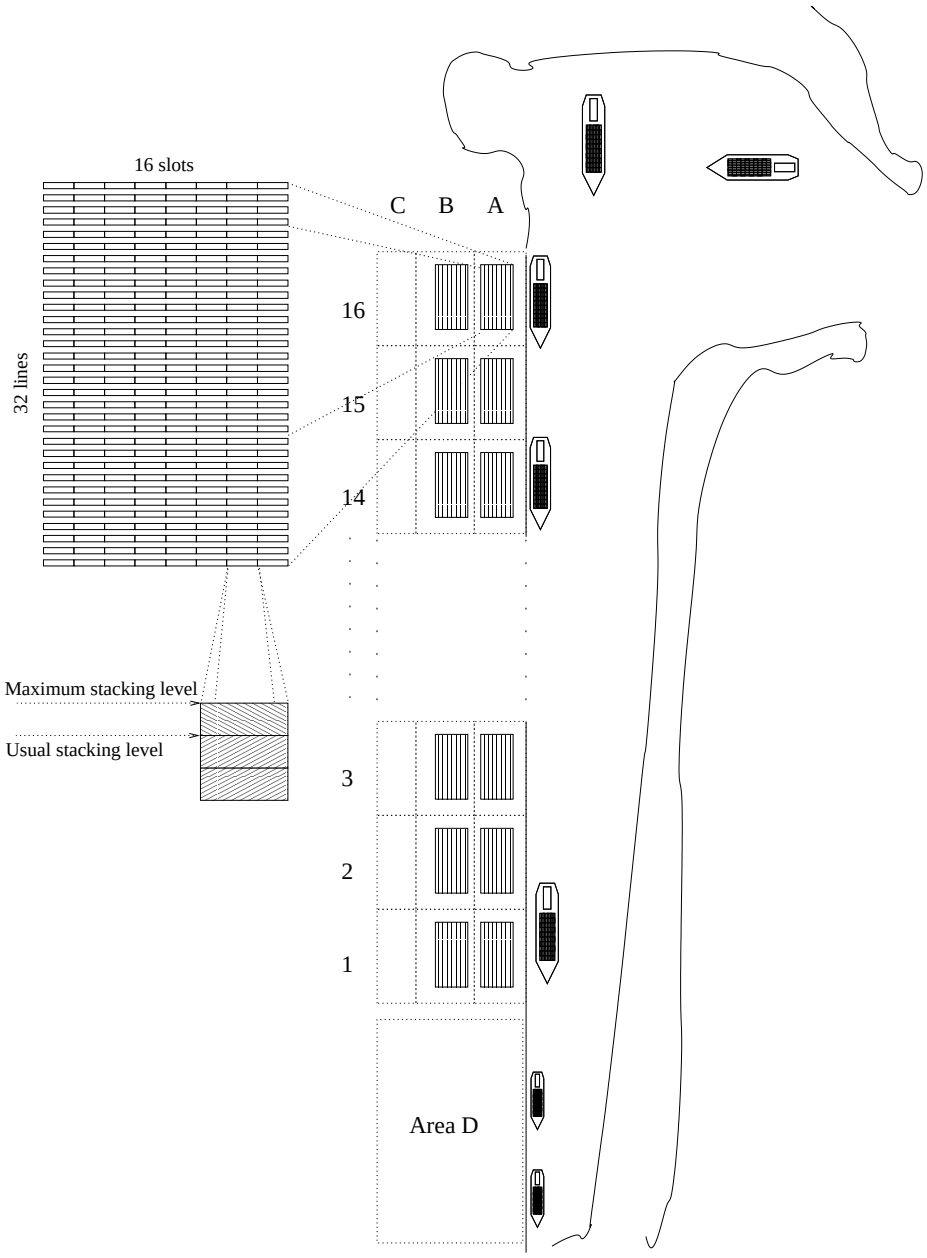

Figure 1: Terminal layout at Gioia Tauro

Areas A and B are divided into 16 bays each. Thus a bay is identified by a level index from one to 16 and by an area code. The distance between adjacent bay centers is 150 meters. The layout of the Gioia Tauro yard is represented in Figure 1 which illustrates only a subset of the bays to keep the figure dimensions reasonable, and also because the central part of the yard presents additional details which will be explained later. An average bay consists of 32 lines, each containing 16 slots. A slot can host up to three one-TEU containers stacked one on top of the other, but it is preferable

to stack only two containers. Accordingly, the yard capacity is computed assuming two TEUs per slot, and thus the average bay has a capacity of 1024 TEUs. There is a significant variation in the capacity of the bays of area B, which range from 749 to 890 TEUs, and in the bays of area A, which range from 1002 to 1240 TEUs. When we consider the total capacity of the bays of areas A and B having the same level index, this quantity ranges from 1882 to 2106 TEUs. Area C is dedicated to the empty containers which have an eight to ten day average dwell time, much more than for a full container. This is the reason why these are located in the most remote positions of the yard. Empty containers can be stacked up to five high. Area D is a dynamic area used for empty containers or full containers, depending on the situation.

1.2 Overview of the service allocation problem

The objective of this study is to solve a tactical yard management problem called the *Service Allocation Problem* (SAP). A *service,* also called *port route*, is the sequence of ports visited by a vessel. Shipping companies plan their port routes in order to match the demand for freight transportation. Scheduling and assigning vessels to port routes are decisional problems solved by shipping companies. A shipping company will usually ask the terminal management to dedicate specific areas of the yard and the quay to their services. This problem, solved by the yard management, is known as the SAP. The allocated area of a service is called the *favorite* area of that service. A service is defined as incoming or outgoing depending on the container flow: when a container arrives in the terminal with a given service, this service is incoming; when a container leaves the terminal, the corresponding service is outgoing. Therefore, the same service can be incoming, outgoing, or both. In a hub terminal the number of outgoing services is usually larger than the number of incoming services (1.5 times as many in our case study). The service allocation has an impact on the number of handling operations inside the yard. In fact, a container arriving with a service is stored in the favorite area of this service, and is transferred to the favorite area of the outgoing service one day before departure in order to speed up loading operations. Terminal managers want to minimize handling operations resulting from yard to yard container transfer, also called *housekeeping*, occurring when the area of the incoming and outgoing services differ. It is important to observe that the SAP is a tactical problem because service allocation decisions are made over a three-month horizon. Decisions made at the tactical level may later be modified when ships actually arrive in the port. An operational problem must then be solved: in order to

avoid traffic congestion and reduce service time, a ship may be assigned to a quay area different from its favorite area. This operational problem is known as the *Berth Allocation Problem* (Cordeau et al., 2005b). The same principle applies to the allocation of storage areas to services in the yard. However, the SAP solution serves as a reference for the operational space allocation problem.

Only areas A and B are relevant to the SAP at Gioia Tauro. The combined capacity of these two areas is 30,000 TEUs. When deciding where a service should be allocated in the yard, only the position of the bay along the quay (indicated in Figure 1 by the numbers 1 to 16) is relevant. The allocation of a container to either area A or area B is determined at the operational level according to the container dwelling time (the dwelling time is the time spent in the yard by a container). Therefore two bays with the same position with respect to the quay are considered for the SAP as a single one. In the following we consider a bay as the union of a bay of area A and the corresponding bay of area B.

Some services require that two adjacent bays be assigned to them. These services correspond to vessels with a length exceeding 250 meters. In fact, since the distance between adjacent bays is 150 meters, the berthing of one of these vessels prevents the use of at least two bays by other vessels. Furthermore, these long vessels generate substantial loading and unloading operations and require an average of three quay cranes. Using two bays facilitates the vehicle flow from the quay cranes to the container slots in the yard. A service with this double requirement is treated by the SAP as a pair of split services.

The Gioia Tauro quay can be approximated with a line showing a discontinuity in the middle, between bays 8 and 9 (Figure 2), which creates an additional constraint for the SAP relative to split services: a pair of split services requiring two bays cannot be allocated to bays 8 and 9.

Assigning a set of services to a bay generates loading and unloading operations (called moves) for the quay cranes operating in front of this bay. The yard space requirement and the quay crane moves associated with a given service are not correlated because of the differences in the container dwelling times between services. For example, assume two services i and j have the same amount of container traffic expressed in number of moves, and service i presents a lower dwelling time than service j. Thus service i will require a lower space requirement than service j in the yard, while the requirements of these two services in terms of moves are equal. It is therefore necessary to impose capacity constraints both on the number of crane moves and on space requirements. The desired operational crane load is set to less than the largest possible

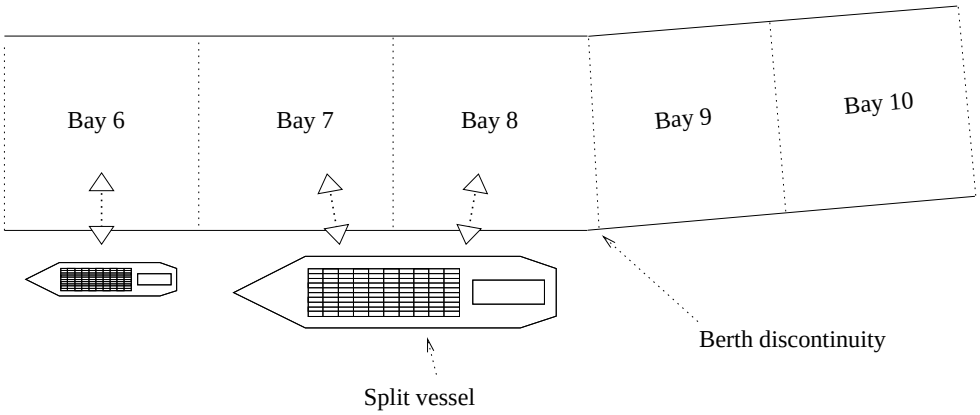

Figure 2: Central part of the Gioia Tauro layout illustrating the berthing constraint for split services at bays 8 and 9

feasible value in the hope of uniformly spreading the expected crane load operations along the quay, thus increasing berth availability and therefore reducing expected vessel waiting time. In fact, reducing housekeeping and vessel waiting times are two conflicting objectives. Reducing housekeeping time usually translates in a higher concentration of vessels in some parts of the quay, which leads to queueing and to longer average service times. Conversely, reducing waiting time often leads to assigning vessels to the first available area along the quay without consideration of handling operations in the yard.

Another SAP constraint derives from the multi-draft channel configuration, i.e. the available draft (water depth) varies along the quay. Before entering the port, each vessel declares the draft it requires. This value depends on the characteristics of the vessel and on its load. Vessels assigned to the same service have similar characteristics and average load, and therefore their draft requests have a small variance. Thus, the draft value for a service can be estimated accurately for the time horizon of the problem. Exceptions are handled at the operational level.

An assignment of services to bays generates housekeeping traffic that can be estimated by multiplying the traffic intensity between services with the distance between the corresponding bays. Given the linear layout and the grid structure of the road network, these distances are computed with the L_1 norm between the centers of the bays. Using the L_1 norm, the distance between two points $a = (a_1, ..., a_n)$ and $b = (b_1, ..., b_n)$ is computed as $\sum_{i=1}^{n} |a_i - b_i|$.

The number of services operating at Gioia Tauro is 62. Currently the SAP is solved by experienced planners without optimization tools. The 62 services are aggregated into 18 families of services and one of the 16 positions is assigned to each family, after which the assignment is

reevaluated to consider splits.

The SAP can be regarded as a constrained version of the *Generalized Quadratic Assignment Problem* (GQAP). In the GQAP we are given n weighted facilities, m capacitated sites, a traffic intensity matrix between facilities, a distance matrix between sites, unit traffic costs, and assignment costs of facilities to sites. The aim is to determine an assignment of facilities to sites in such a way that the sum of assignment and traffic costs is minimized and the total weight of all facilities assigned to the same site does not exceed the capacity of the site. The GQAP was introduced by Lee and Ma (2003). It generalizes the *Quadratic Assignment Problem* (QAP) in which $n = m$ and exactly one facility must be assigned to each site. It is therefore NP-hard (Garey and Johnson, 1979) and the same holds for the SAP.

1.3 Outline of the remainder of this article

In the following we first model the SAP with a quadratic programming formulation and we then present two linearization approaches (Section 2). Since only small size instances can be solved optimally within a reasonable time, an evolutionary heuristic is introduced in Section 3, followed by computational results in Section 4, and by the conclusions in Section 5.

2 Mathematical model

We present in Section 2.1 an integer quadratic programming formulation for the SAP and we then introduce in Section 2.2 a linearization that exploits the linear layout of the Gioia Tauro yard. A more general case with a two-dimensional grid yard is considered in Section 2.3. For comparison purposes a second formulation for our problem is introduced in Section 2.4. It is an adaptation of the linearization for the GQAP introduced by Lee and Ma (2003) and is based on the *Quadratic Assignment Problem* (QAP) linearizations proposed by Frieze and Yadegar (1983) and by Padberg and Rijal (1996).

2.1 Quadratic SAP formulation

To model the SAP we denote by $\tilde{N} = \{1, ..., \tilde{n}\}$ the set of services, and by $D = \{1, ..., n_d\} \subset \tilde{N}$ the subset of services requiring a double assignment. A set $D^{'} = \{\tilde{n} + 1, ..., \tilde{n} + n_d\}$ is introduced to represent the split services associated with D, where $\tilde{n} + i$ is the split service associated with

$i \in D$. The SAP is therefore defined on a set $N = \tilde{N} \cup D^{'}$, where $|N| = n = \tilde{n} + n_d$. For notational convenience define $N^{+}(i) = \{j \in N : j > i\}$. The set of bays is defined by M, with $|M| = m$. Binary decision variables x_{ik} are equal to 1 if and only if service $i \in N$ is assigned to bay $k \in M$. The other input data for the SAP are:

- $t_{ij},$ the traffic intensity between services $i \in N$ and $j \in N^{+}(i)$, expressed in average number of containers per day; the value of t_{ij} is in fact the sum of the outgoing containers from i to j and from j to i;
- $q_i,$ the space requirement of service $i \in N$, expressed in average number of TEUs per day;
- $Q_k,$ the space available at bay $k \in M$, expressed in average number of TEUs per day;
- $c_i,$ the average number of crane moves required for service $i \in N$ per day;
- $C_k,$ the average number of crane moves allowed at bay $k \in M$ per day;
- $M(i)$, a subset of M representing the feasible bay assignments for service i resulting from the draft constraint.

Service allocation decisions are made over a three-month horizon. The input data are thus the expected values over this period. In practice, the average daily values (i.e., the estimate divided by the number of days in the period) are preferred since they relate to common performance measures (vessels per day, moves per day, etc.).

Given the linear layout of the Gioia Tauro yard, each bay can be identified by one position coordinate and the distances between bays is calculated as the difference between these coordinates. Thus define:

- $a_k,$ the position coordinate of bay $k \in M$,
- $d_{hk} = |a_h - a_k|\,,$ the distance between bays $h \in M$ and $k \in M$.

A straightforward integer quadratic programming formulation of the SAP is given by:

$$\text{minimize} \sum_{i \in N \setminus \{n\}} \sum_{j \in N^{+}(i)} \sum_{h \in M} \sum_{k \in M} t_{ij} d_{hk} x_{ih} x_{jk} \tag{1}$$

subject to

$$\sum_{k \in M(i)} x_{ik} = 1 \qquad \forall i \in N, \tag{2}$$

$$\sum_{i \in N} q_i x_{ik} \leq Q_k \qquad \forall k \in M, \tag{3}$$

$$\sum_{i \in N} c_i x_{ik} \leq C_k \qquad \forall k \in M, \tag{4}$$

$$x_{ik} = x_{\tilde{n}+i,k-1} \qquad \forall i \in D, \forall k \in M(i) \setminus \{1\}, \tag{5}$$

$$x_{i9} = 0 \qquad \forall i \in D, \tag{6}$$

$$x_{ik} \in \{0, 1\} \qquad \forall i \in N, \forall k \in M. \tag{7}$$

The objective function to minimize is the sum of traffic intensities between services weighted by the distances resulting from the assignment decisions. We therefore minimize the housekeeping, i.e. the handling operations resulting from yard to yard container transfers occurring when the area of the incoming and outgoing services differ. Constraints (2) state that each service must be assigned to one and only one bay and this bay must belong to the feasible set of the service. Constraints (3) and (4) ensure that the total space and the total crane moves required by the services assigned to a bay do not exceed its availability. Constraints (5) force the split pair $(i, \tilde{n}+i)$ to be assigned to positions k and $k-1$, with $k \in M(i) \setminus \{1\}$, and constraints (6) forbid the assignment of a split pair to bays 8 and 9 which present a discontinuity.

2.2 Linearization of the SAP formulation

To take advantage of the linear layout of the Gioia Tauro yard (see Figure 3) we introduce a position variable, y_i, $i \in N$, equal to a_k whenever $x_{ik} = 1$. A formulation of the SAP using y_i variables but still with a non-linear objective function follows:

$$\text{minimize} \sum_{i \in N \setminus \{n\}} \sum_{j \in N^+(i)} t_{ij} \, |y_i - y_j| \tag{8}$$

subject to (2) - (7) and

$$y_i = \sum_{k \in M(i)} a_k x_{ik} \qquad \forall i \in N. \tag{9}$$

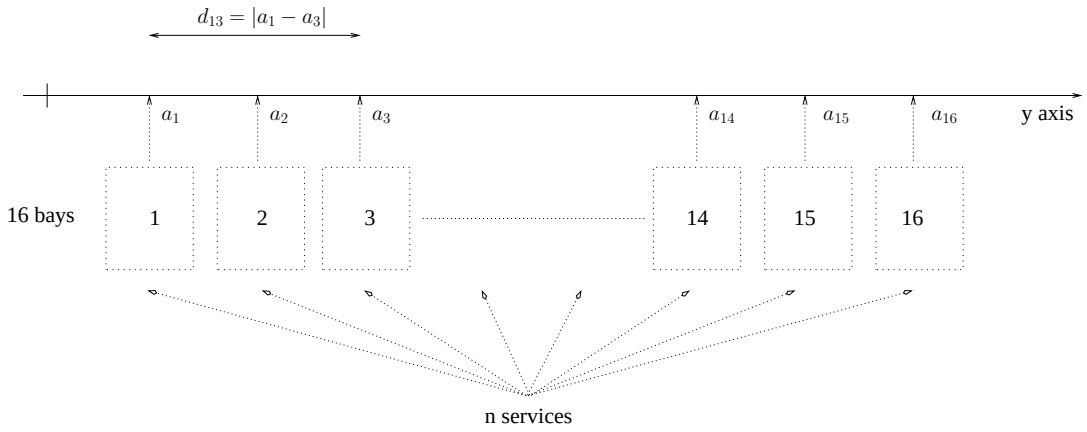

Figure 3: Linear disposition of the bays modelled by the SAP

Position variables y_i are consistent with the assignment variables x_{ik} because of constraints (9). This formulation can easily be linearized through the introduction of continuous non-negative variables ϕ_{ij} and ψ_{ij} and additional constraints, yielding the L_1SAP formulation:

$$\text{minimize} \sum_{i \in N \setminus \{n\}} \sum_{j \in N^+(i)} t_{ij}(\phi_{ij} + \psi_{ij}) \tag{10}$$

subject to (2) - (7), (9) and

$$y_i - y_j \leq \phi_{ij} \qquad \forall i \in N \setminus \{n\}, \forall j \in N^+(i), \tag{11}$$

$$y_j - y_i \leq \psi_{ij} \qquad \forall i \in N \setminus \{n\}, \forall j \in N^+(i), \tag{12}$$

$$\phi_{ij} \geq 0 \qquad \forall i \in N \setminus \{n\}, \forall j \in N^+(i), \tag{13}$$

$$\psi_{ij} \geq 0 \qquad \forall i \in N \setminus \{n\}, \forall j \in N^+(i). \tag{14}$$

Because the objective function is minimized, it follows by constraints (11) and (13) that $\phi_{ij} = y_i - y_j$ whenever $y_i - y_j \geq 0$. Similarly, by constraints (12) and (14), $\psi_{ij} = y_j - y_i$ whenever $y_i - y_j \leq 0$. If the inter-bay distance is a constant, as it is in our case, we can, through the y_i variables, formulate the adjacency of the split services with only n_d constraints instead of $n_d(m-1)$ as in (5):

$$y_i - y_{\tilde{n}+i} = 1 \qquad \forall i \in D. \tag{15}$$

In Section 4 we will discuss the opportunity of integrating constraints (15) in the L_1SAP formulation, along with constraint (16) which, while redundant, strengthens the relaxation:

$$\sum_{i \in N} y_i = \sum_{i \in N} \sum_{k \in M} a_k x_{ik}. \tag{16}$$

2.3 Extension of the linearized model to a two dimensional grid

The L_1SAP formulation, which exploits the L_1 norm used to compute distances between the bays, can be extended to handle the case of a two-dimensional grid. Let a generic bay $k \in M$ be identified by a pair (a_k, b_k), and let w_i, $i \in N$, be a position variable equal to b_k whenever $x_{ik} = 1$ (see Figure 4). The model of Section 2.2 then becomes:

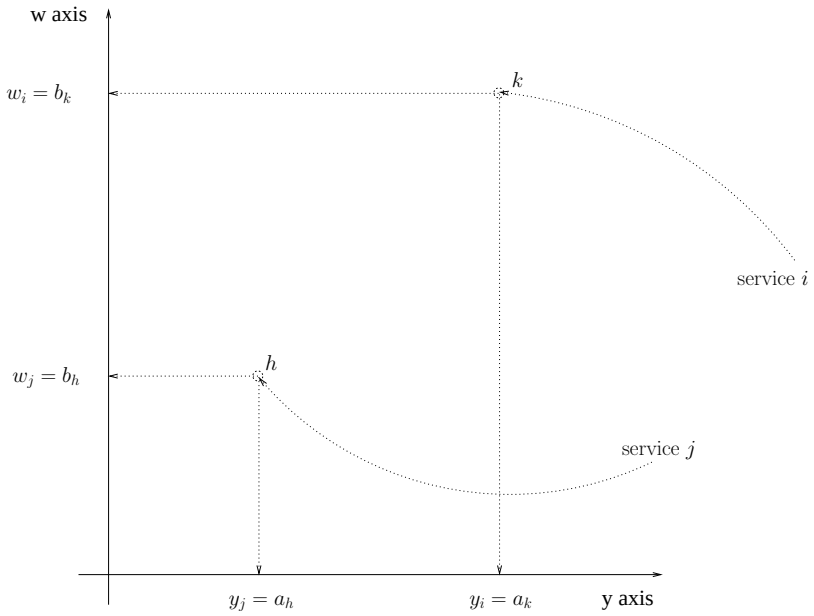

Figure 4: Two-dimensional case with L_1 norm

$$\text{minimize} \sum_{i \in N \setminus \{n\}} \sum_{j \in N^+(i)} t_{ij}(\phi_{ij} + \psi_{ij} + \beta_{ij} + \gamma_{ij}) \tag{17}$$

subject to (2) - (7), (9), (11) - (14) and

$$w_i = \sum_{k \in M} b_k x_{ik} \qquad \forall i \in N, \tag{18}$$

$$w_i - w_j \leq \beta_{ij} \qquad \forall i \in N \setminus \{n\},\ \forall j \in N^+(i), \tag{19}$$

$$w_j - w_i \leq \gamma_{ij} \qquad \forall i \in N \setminus \{n\},\ \forall j \in N^+(i), \tag{20}$$

$$\beta_{ij} \geq 0 \qquad \forall i \in N \setminus \{n\},\ \forall j \in N^+(i), \tag{21}$$

$$\gamma_{ij} \geq 0 \qquad \forall i \in N \setminus \{n\},\ \forall j \in N^+(i). \tag{22}$$

A similar approach can be used for a problem with the L_∞ norm, but since it is straightforward and not relevant for our problem we omit it.

2.4 Linearization of the GQAP

Lee and Ma (2003) have introduced a linearization for the GQAP, based on QAP linearizations proposed by Frieze and Yadegar (1983) and by Padberg and Rijal (1996). Adapting this formulation to the our problem is simple. The formulation, called LGQAP, is as follows:

$$\text{minimize} \sum_{i \in N \setminus \{n\}} \sum_{j \in N^+(i)} \sum_{k \in M} \sum_{h \in M} t_{ij} d_{hk} z_{ikjh} \tag{23}$$

subject to (2) - (7) and

$$\sum_{h \in M} z_{ikjh} = x_{ik} \qquad \forall i \in N \setminus \{n\},\ \forall k \in M,\ \forall j \in N^+(i), \tag{24}$$

$$\sum_{k \in M} z_{ikjh} = x_{jh} \qquad \forall i \in N \setminus \{n\},\ \forall j \in N^+(i),\ \forall h \in M, \tag{25}$$

$$0 \leq z_{ikjh} \leq 1 \qquad \forall i \in N \setminus \{n\},\ \forall k \in M,\ \forall j \in N^+(i),\ \forall h \in M. \tag{26}$$

LGQAP uses additional variables z_{ikjh} equal to 1 if and only if service i is assigned to bay k and service j is assigned to bay h. In Section 4 we will present comparative computational results obtained with the L_1SAP and LGQAP formulations.

3 M-SAP, a Memetic Heuristic for the SAP

We now present a memetic heuristic called M-SAP, based on a previously developed heuristic for the GQAP (Cordeau et al., 2005a) refereed to as M-GQAP. While M-GQAP considers only one set of capacity constraints, M-SAP must be capable of handling two sets of capacity constraints, as well as dedicated constraints for split pairs of services, specific berth characteristics like the berth discontinuity, and infeasible assignments resulting from the draft constraint. We will describe the components of M-SAP and highlight its differences with respect to M-GQAP.

A memetic heuristic (Moscato and Cotta, 2003) combines genetic search (Holland, 1992) and tabu search (Glover, 1986). The algorithm randomly generates an initial *population* P of feasible solutions, and a cycle of genetic iterations is then applied to P. At each iteration a merging procedure creates an offspring which is then improved trough at most η iterations of tabu search. If the resulting improved offspring is better than the worst member of the population and does not already belong to P, it then replaces the worst solution; otherwise the offspring is discarded. To enhance the exploration of different areas of the solution space, a multi-start approach is also used, meaning that the algorithm is started ξ times and the best solution is retained. We now present the components of the heuristic in more detail.

3.1 Initial population

Denote by S the set of solutions satisfying constraints (2) - (7) and by $\tilde{S}$ the set of solutions satisfying (2) - (4) and (7). A first solution $s \in \tilde{S}$ is randomly generated by assigning services to bays while attempting to respect the capacity constraints (3) and (4). Reaching feasibility with respect to these constraints is not always possible, particularly in the case of tightly constrained instances. Furthermore, because constraints (5) and (6) are not explicitly considered during the random assignment, these are seldom satisfied in the initial solution. Therefore, an attempt to reach feasibility is made through the application of the TABUFIX search procedure described in Section 3.4. If TABUFIX fails, or if the solution thus produced is already in P, a new solution is generated until a desired population size $|P|$ or a time limit is reached. If the time limit is reached the algorithm stops with an error message, since it failed in generating the desired population size.

3.2 Genetic procedure

The genetic procedure attempts to improve the population P by merging, at each iteration ω, two solutions to create an offspring. The basic merging rule used is inspired by the work of Drezner (2003) for the QAP. A modified merging procedure that considers capacity constraints was developed for the GQAP (Cordeau et al., 2005a). Because the SAP contains additional constraints with respect to the GQAP, the merging procedure we have developed for the SAP is new. The additional constraints relate to crane moves (4), the split pair continuity requirements (5), and the berth discontinuity (6). Constraints (2) are also different form those of the GQAP because the draft requirement forbids some assignments.

For each bay $k \in M$ the median value of the distances between bay k and the other $m-1$ bays is first computed. Bays whose distance from k is less than the median value are called *cohesive* with respect to k. All bays that are at the same distance as the median value are called *neutral*, and the remaining bays are called *non-cohesive*. For each bay k it is then possible to partition M into a subset M_1^k of cohesive bays, a subset M_2^k of neutral bays, and a subset M_3^k of non-cohesive bays. Bay k is considered as belonging to M_1^k. A bay k selected to partition M into subsets is called *a pivot*. The construction of the partitions of M for each pivot is performed only once at the beginning of the algorithm.

The merging procedure randomly selects two solutions of the population pool called *parents*. The parent A_1 having the smaller solution value is called *first parent* and the other parent A_2 is called *second parent*. Ties are broken arbitrarily. Like all elements of P, these two solutions satisfy constraints (2) - (7). The merging procedure operates as a cycle of iterations applied to all bays $k \in M$. Each iteration attempts to generate a new solution $s^k \in \tilde{S}$ which, as we have mentioned, is feasible with respect to constraints (2) - (4) and (7). Therefore the merging procedure relaxes constraints (5) - (6) related to the adjacency requirement for split pairs and to the berth discontinuity. We denote by $p_1(s^k)$ the amount of violation of constraints (5) - (6) in solution s^k and we compute it as follows: $p_1(s^k) = \sum_{i \in D}\{||\nu(i) - \nu(\tilde{n}+i)| - 1| + x_{i9}\}$, where $\nu(i)$ is the assigned bay for services $i \in N$, and $p_1(s^k) = 0$ if $s^k \in S$. In the merging cycle the solutions generated are compared by means of a penalized cost function $f(s^k) = c(s^k)(1 + p_1(s^k) \times m \times \zeta)$, where $c(s^k)$ is the cost function as stated in (1). In this expression the factor m is used to penalize more heavily the violation $p_1(s^k)$ when the instance becomes larger and hence more difficult. The factor ζ was set equal to 0.1 to reduce the impact of m.

In s^k a bay h belonging to the cohesive subset M_1^k is assigned all services previously assigned to that bay h in the first parent A_1. Similarly a bay in the non-cohesive subset M_3^k is assigned all services from the second parent A_2, if these services are not already assigned to the cohesive subset. After this phase some services may still be unassigned. An attempt is then made to randomly assign these to bays while preserving feasibility on capacity constraints (4) - (5), but if no feasible assignment can be reached the solution is then discarded. If a solution $s^k \in \tilde{S}$ is obtained, it is then compared, using $f()$, to the best known solution s^*_ω found at iteration ω among those corresponding to all pivot bays; if s^k improves upon s^*_ω, it then replaces it. As a result of this merging cycle we can obtain a solution $s^*_\omega \in \tilde{S}$ that will not necessarily satisfy constraints (5) and (6). In this case the TABUFIX search procedure is called to reach feasibility. Merging two parents can be unsuccessful if no solution $s^*_\omega \in \tilde{S}$ is encountered during the m iterations or if TABUFIX fails to restore feasibility. In this case another set of parents is randomly selected and the process is repeated until a feasible offspring is generated. After every successful or unsuccessful merging, the two parents are labelled as *forbidden*, and no other merging is attempted with them. The number of forbidden couples is updated during the search. When this number reaches its maximum possible value of $|P| \times (|P| - 1)/2$, the genetic cycle is stopped. We refer to Cordeau et al. (2005a) for an example illustrating the merging process with capacity constraints.

After the merging procedure a post-optimization tabu search heuristic (POTS) is called to improve the best offspring $s^*_\omega \in S$. The resulting solution is compared with the worst solution in P and is discarded if it does not improve upon it. Otherwise the improved offspring is compared with the other solutions in P. If the offspring is not original, meaning that the same solution already exists in P, it is discarded; otherwise, it is inserted in the population P to replace the worst solution. The information about the forbidden coupling are therefore updated.

At the end of an iteration the genetic iteration counter ω is incremented and another iteration is performed until a limit ϖ is reached. The genetic cycle may terminate prematurely if the number of forbidden couples reaches the maximum value of $|P| \times (|P| - 1)/2$. When this event occurs the population P becomes exhausted, no coupling is possible, and a new random population is required. Whenever $\omega > \varpi$ or premature termination occurs, the genetic procedure gives the control to the multi-start cycle which compares the best solution found during the current genetic cycle with the preceding ones, if any. The genetic procedure can be summarized as follows:

- for $\omega = 1, ..., \varpi$ { /* genetic cycle */

- Do {
  - randomly select a non-forbidden couple (A_1, A_2) in P
  - for $k = 1, ..., m$ { /* merging cycle */
    - merge (A_1, A_2) using k as pivot and obtain $s^k \in \tilde{S}$, if any
    - using the penalized cost function $f()$, compare s^k to the best known offspring s^*_ω, if any}
  - if $s^*_\omega \notin S$ call TABUFIX
- } until an offspring is generated
- Apply POTS to the offspring s^*_ωwhich eventually is improved
- Compare s^*_ω with the worst and the best solutions in P
- If s^*_ω is better than the worst and not better than the best solution, check whether it is original
- Replace the worst solution with s^*_ω if it is original and better than the worst solution
- Update the information about forbidden couples and eventually decide to terminate the genetic cycle}

3.3 Post-optimization tabu search

The POTS procedure allows temporary violations of the capacity constraints (3) - (4) and of the adjacency and berth discontinuity constraints (5) - (6). Denote by S^A the set of solutions satisfying the assignment constraints (2) and (7). As before, let $c(s)$ denote the cost of a solution $s \in S^A$. Let also $p_2(s)$ be the violation of the yard capacity constraints (3) and $p_3(s)$ be the violation of the crane moves capacity constraints (4):

$$p_2 = \sum_{k \in M} \max\{0, \sum_{i \in N} q_i x_{ik} - Q_k\},$$
$$p_3 = \sum_{k \in M} \max\{0, \sum_{i \in N} c_i x_{ik} - C_k\}.$$

Solutions are then evaluated by means of a penalized cost function $g(s) = c(s) + [1 + (p_2(s) + p_3(s))/m]\alpha_1 p_1(s) + \alpha_2 p_2(s) + \alpha_3 p_3(s)$, where $\alpha_l,\ l \in \{1, 2, 3\}$ are positive parameters. Since usually

$p_1(s) << p_2(s) + p_3(s)$, the term $[1 + (p_2(s) + p_3(s))/m]$ acts as a scaling factor. By dynamically adjusting the value of the α_l parameters the relaxation mechanism facilitates the exploration of the search space and is particularly useful to escape local minima. Parameter α_l is updated at each iteration as follows: if the new current solution s satisfies $p_l(s) = 0$, then α_l is divided by a factor $1 + \delta$, where δ is a positive parameter; otherwise α_l is multiplied by $1 + \delta$.

The heuristic explores the solution space by moving at each iteration from the current solution $s \in S^A$ to the best solution in its neighborhood $H(s)$. The tabu search method is based on the definition of attributes used to characterize the solutions of S^A. These attributes also control tabu tenures. With each solution $s \in S^A$ is associated an attribute set $B(s) = \{(i, k)$: facility i is assigned to bay $k\}$. The neighborhood $H(s)$ of a solution s is defined by applying a simple operator that removes an attribute (i, k) from $B(s)$ and replaces it with another attribute (i, k'), where $k \neq k'$. We note that $k' \in M(i)$, since constraints (2) and (7) are satisfied for $s \in S^A$. When facility i is removed from bay k, its reassignment to k is declared forbidden for the next θ iterations by assigning a tabu status to attribute (i, k). An aspiration criterion allows the revocation of the tabu status of an attribute if that would allow the search process to reach a solution of smaller cost than that of the best solution identified possessing that attribute.

Every time an improved feasible solution is reached a counter ζ^* is set to ζ, the number of the current iteration. When the difference $\zeta - \zeta^*$ becomes larger than a preset value μ, POTS terminates and the current best solution is returned to the genetic cycle. However, the maximum number of iterations allowed is set equal to η.

3.4 Recovering feasibility: procedure TABUFIX

TABUFIX is an auxiliary procedure called by two of the procedures just described (initial population and genetic) when a feasible solution cannot be reached. TABUFIX is a variant of POTS. It starts from an infeasible solution s with some violation $p_l(s) > 0, l \in \{1, 2, 3\}$. The cost function $g(s)$ is replaced with $g(s) = [1 + (p_2(s) + p_3(s))/m]p_1(s) + p_2(s) + p_3(s)$ and thus the penalization mechanism driven by the parameters α_l is no longer necessary. The value of the tabu tenure parameter, θ_f, can be different from that of θ used by POTS. The neighborhood structure and all the other tabu search mechanisms are the same as those of POTS, but TABUFIX employs also a diversification scheme. A solution $\overline{s} \in N(s)$ such that $f(\overline{s}) \geq f(s)$ is penalized by a factor proportional to the addition frequency of its distinguishing attribute, and by a scaling factor. More precisely, let

ρ_{ik} be the number of times attribute (i,k) has been added to the solution during the process and let ζ be the number of the current iteration. If attribute (i,k) is added to s to obtain $\overline{s}$, a penalty $p(\overline{s}) = \lambda\sqrt{nm}\, c(\overline{s})\rho_{ik}/\zeta$ is added to $f(\overline{s})$, where $p(\overline{s}) = 0$ if $f(\overline{s}) < f(s)$. The scaling factor $c(\overline{s})$ introduces a correction to adjust the penalties with respect to the total solution cost. The use of $\sqrt{nm}$ as a scaling factor was first suggested in the context of the VRP by Taillard (1993) after extensive computational experiments. It can be explained as follows: since there are $O(nm)$ possible attributes, the frequency of addition of a given attribute into the solution is inversely related to problem size. Using the factor $\sqrt{nm}$ seems to adequately compensate for problem size. Finally, a parameter λ is used to control the intensity of the diversification. These penalties have the effect of driving the search process toward less explored regions of the search space. TABUFIX ends as soon as a feasible solution is reached, meaning that $\sum_{l=1}^{3} p_l(s) = 0$. The maximum number of iterations allowed is set equal to η_f.

3.5 Differences between M-SAP and M-GQAP

With respect to what is done in M-GQAP, the penalized cost function used in POTS was modified to accommodate violations of new constraints present in SAP, namely: the adjacency requirement for split pairs, the berth discontinuity, and capacity constraints relative to quay crane moves. The neighborhood structure $H(s)$ must also consider feasibility for the draft requirement. The major impact of the SAP structure is on the genetic procedure, where the merging cycle alone infrequently generates a feasible offspring. Therefore an appropriate relaxation was developed for the merging phase and the procedure TABUFIX was also used in this context. Other differences between the SAP and the GQAP, like the absence of installation costs in the objective function and use of an L_1 norm, have led to other smaller variations between the two heuristics.

4 Computational results

We first describe how test instances were generated, and then provide results obtained with our heuristic compared with those obtained with the two exact formulations of Sections 2.2 and 2.4.

4.1 Test instances

Sixty-two services are listed in the Gioia Tauro port operation database. Ordering the services according to their expected yard space requirement q_i shows that the first 33 contribute to 95% of the total space requirement. This situation is illustrated in Figure 5.

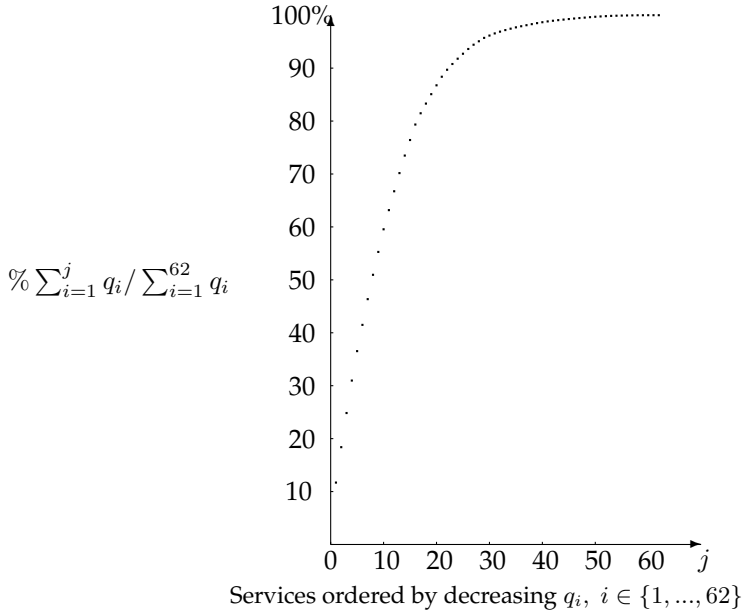

Figure 5: Relative importance of the Gioia Tauro's services

A similar observation can be made by ordering the services according to the c_i values. Among these 33 services, six require a double assignment, i.e., $n_d = 6$ and $n = 39$. Therefore the full SAP problem contains 39 services to be assigned to 16 bays, if the less important services are neglected. The yard space and crane moves required by each service and the interactions between services are derived by a statistical assessment of the terminal database. The instance thus defined is called "39-16". As we will show, this instance is too large to be solved by applying CPLEX to an exact formulation. To test the heuristic versus an exact formulation we have therefore created some easier instances. An instance called "20-8" is obtained from the original "39-16" data, by selecting a subset of services which are assigned, under the current SAP decisions at Gioia Tauro, to the bay set $\{9, ..., 16\}$. These services are representative because they all belong to the same and most important shipping company operating at Gioia Tauro. Furthermore, we have designed a new set R of ten randomly generated instances. These are labelled with three letters: n, m, and a parameter $f \in \{1, ..., 100\}$ which controls the tightness of the capacity constraints. For given n, m and f, the

input data were generated as follows:

- t_{ij} was generated according to a discrete uniform distribution in $[1, ..., 100]$,
- q_i and c_i were generated according to a discrete uniform distribution in $[1, ..., 100]$,
- C_k was generated according to a discrete uniform distribution in $[1, ..., 100 \times \sum_{i \in N} c_i/(fm)]$,
- Q_k was generated according to a discrete uniform distribution in $[1, ..., 100 \times \sum_{i \in N} q_i/(fm)]$.

Given a service $i \in N$, each $k \in M$ belongs to $M(i)$ with probability 90%. The number of split pairs n_d was taken equal to $\lfloor 0.2n \rfloor$ and, after randomly selecting n_d pairs in N, their data were adjusted according to the description given for this set of services (a split pair $(i, \tilde{n} + i)$ must have the same draft requirement, and $t_{ij} = t_{\tilde{n}+i,j}$ $\forall i \in D$ and $\forall j \in N$).

4.2 Implementation details and results

The M-SAP procedures were coded in ANSI C and the two exact formulations were implemented in CPLEX 8.1. Computational experiments were performed on a SUN workstation (1.2 GHz).

For the L_1SAP formulation, let LB1 be the linear relaxation value with constraints (15) and (16), and let LB2 be the linear relaxation value without these. Figure 6 plots the lower bounds LB1 and LB2 at various levels of the branch-and-bound enumeration tree for the "20-08" instance. Similar patterns can be observed on other instances. Constraints (15) and (16) are therefore effective in improving the linear relaxation. In the following the CPLEX implementation of L_1SAP refers to the one with constraints (15) and (16).

The parameters used in the various M-SAP procedures are as follows:

- Initial Population:
  - $|P|$: population size, equal to 150.
- Genetic Procedure:
  - ϖ: maximum number of genetic iterations, equal to 8000.
- Post-Optimization Tabu Search:
  - θ: tabu duration, equal to 64;
  - μ: limit when POTS is terminated prematurely, equal to 20;

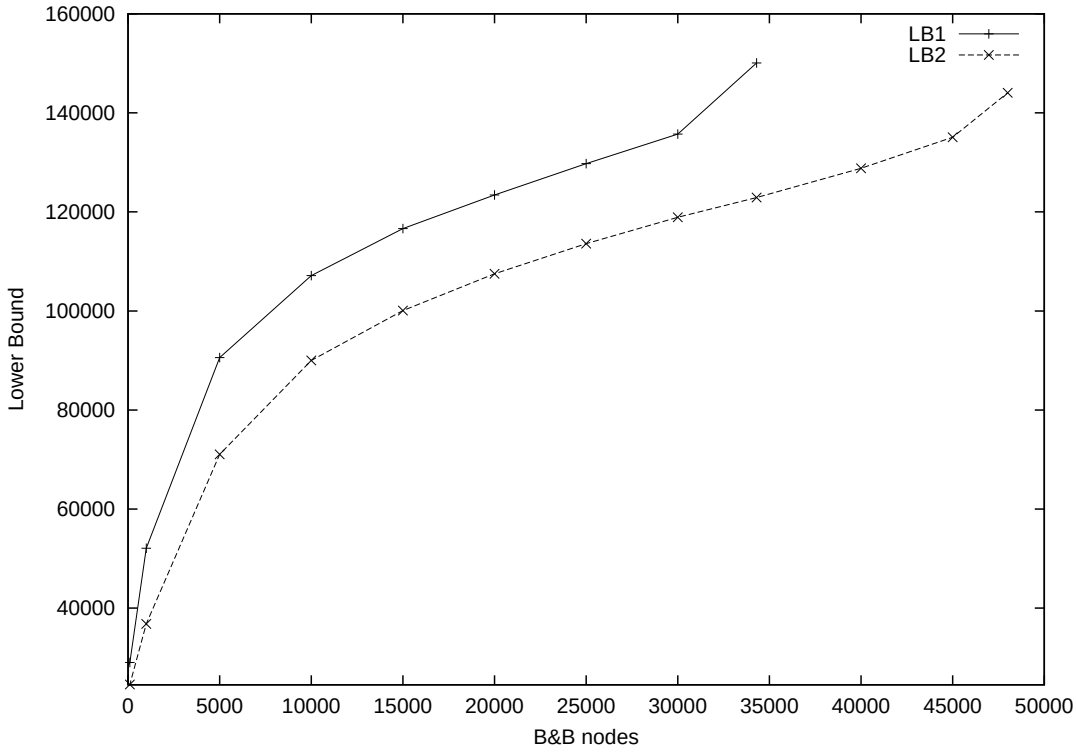

Figure 6: Lower Bounds of two implementations of L_1SAP.

- δ : penalty adjustment parameter, equal to 1;
- η: maximum number of POTS iterations, equal to 1000.

• TABUFIX:

- θ_f : tabu duration, equal to 128;
- λ : diversification intensity parameter, equal to 1;
- η_f: maximum number of iterations, equal to 3200.

These parameter values are the same as those used for the M-GQAP heuristic, i.e. we did not attempt specific parameter settings for the instances of this study. The computation time of the heuristic is influenced by the parameter ξ (the number of times the algorithm is started) and this is the only parameter that we will allow to vary during our computational experiments for the SAP.

Table 1 reports the results of the heuristic and the two exact formulations implemented in CPLEX on the real data instances. Here the multi-start mechanism was disabled, i.e. $\xi = 1$. The small instance "20-08" is easily solved to optimality by all three algorithms, but LGQAP is the

slowest one. On the large instance "39-16" we set a 60 hour time limit on the two CPLEX implementations. M-SAP obtains a better solution in far less computational time than these truncated branch-and-bound algorithms.

	M-SAP - $\xi = 1$		L_1SAP-CPLEX			LGQAP-CPLEX		
Instance code	Solution	Time (min.)	Solution	Gap[a] (%)	Time (min.)	Solution	Gap[a] (%)	Time (min.)
20-08	**159994**	0.2	**159994**	0.0	4.6	**159994**	0.0	129.1
39-16	**377606**	5.2	379846	74.7%	3600.0	559858	83.7%	3600.1

Table 1: Real data instances, best values in bold

a - The gap is computed with respect to the value of the linear relaxation as (upper bound - lower bound)/upper bound.

Table 2 illustrates how solution quality improves with the number of M-SAP iterations for instance "39-16". However, from $\xi = 80$ upwards there are no additional improvements. The corresponding CPU time of 463 minutes is still far less than the time limit allowed to CPLEX in the tests reported above.

	ξ					
	1	10	20	40	80	160
Solution value	377606	372346	365110	359232	**357018**	**357018**
100 × solution value / best solution	105.8	104.3	102.3	100.6	100.0	100.0
CPU time (min.)	5.2	59.9	118.6	233.4	463.0	926.4

Table 2: M-SAP on instance "39-16" varying ξ, best values in bold

Table 3 provides the computation results for the instance set R. The first three instances were easily solved by all the three algorithms. The fourth instance, while solved to optimality by L_1SAP, proved to be difficult for LGQAP, and reached the imposed time limit of two hours before fully exploring the branch-and-bound tree. M-SAP with $\xi = 10$ always obtains the optimum when L_1SAP does, but requires shorter average computational times and yields better results on the instances where an optimal solution was not found.

In Table 4 we report results produced by the heuristic with five different ξ values in the set $\Xi = \{10, 20, 40, 80, 160\}$. The heuristic exhibits a stable behavior since the quality of the solutions does not improve significantly when longer computational times are allowed. The solution quality index was computed as follows: let be c_i^* the minimum solution value obtained for a given instance i over all runs for $\xi \in \Xi$, and let c_i^χ be the solution value of instance i with $\xi = \chi$; the average of the values $100 \times c_i^\chi / c_i^*$ is our solution quality index for the choice $\xi = \chi$.

	M-SAP - $\xi = 10$		L_1SAP-CPLEX			LGQAP-CPLEX		
Instance code	Solution	Time (min.)	Solution	Time (min.)	Gap[a] (%)	Solution	Time (min.)	Gap[a] (%)
15-10-75	**14688**	1.1	**14688**	**0.6**	0.0	**14688**	19.1	0.0
20-05-65	**7812**	**1.2**	**7812**	1.2	0.0	**7812**	4.5	0.0
20-10-65	**18804**	**3.0**	**18804**	8.4	0.0	**18804**	46.9	0.0
20-10-75	**19476**	**2.8**	**19476**	61.2	0.0	21034	120.0	26.0
22-11-85	**33456**	**4.1**	34764	120.0	36.0	38254	120.0	32.1
25-15-60	**36782**	**6.0**	40834	120.0	75.9	49728	120.2	83.8
25-15-70	**43182**	**5.9**	45250	120.0	69.3	49600	120.2	74.7
25-15-80	**49480**	**6.8**	52432	120.0	74.2	59842	120.2	75.0
25-15-90	**62438**	**11.3**	63644	120.0	54.0	78232	120.2	69.4
30-15-70	**63178**	**8.6**	69940	120.0	82.7	91470	120.3	88.9
Average		**5.1**		*79.2*			*91.1*	

Table 3: Random generated instances, best values in bold
a - The gap is computed with respect to the value of the linear relaxation as (upper bound - lower bound)/upper bound.

	ξ				
Instance code	10	20	40	80	160
15-10-75	**14688**	**14688**	**14688**	**14688**	**14688**
20-05-65	**7812**	**7812**	**7812**	**7812**	**7812**
20-10-65	**18804**	**18804**	**18804**	**18804**	**18804**
20-10-75	**19476**	**19476**	**19476**	**19476**	**19476**
22-11-85	**33456**	**33456**	**33456**	**33456**	**33456**
25-15-60	36782	36782	**36568**	**36568**	**36568**
25-15-70	43182	43012	**41876**	**41876**	**41876**
25-15-80	49480	**49276**	**49276**	**49276**	**49276**
25-15-90	62438	**61886**	**61886**	**61886**	**61886**
30-15-70	63178	63178	61646	61646	**61442**
Average CPU time (min.)	5.1	10.1	20.2	40.0	79.9
Solution quality	100.78	100.61	100.03	100.03	100.00

Table 4: M-SAP sensitivity to ξ on instance set R, best values in bold

5 Conclusions

We have described, formulated and solved the service allocation problem, a difficult combinatorial problem arising in the tactical management of container ports. We have developed a memetic heuristic for the problem. To test the heuristic two linearizations of the SAP were implemented in CPLEX. The solution values provided by the heuristic were compared on small instances to the optimal values generated by the exact formulations. On these instances our heuristic always yields optimal solutions. On larger instances it always outperforms the truncated CPLEX branch-and-bound algorithm applied to the two exact formulations. These computational results confirm that the heuristic can be used to handle real-life SAP instances. Therefore, the heuristic fills the gap between the large amount of data that can be gathered by the terminal information system and the current practice where the SAP decisions are made by rules of thumb.

Acknowledgment

This work was supported by the Canada Research Chair in Distribution Management, by the HEC Centre for International Business Studies, by the CN Chair in Transport Economics and Intermodality, and by a strategic research grant from HEC Montréal. The authors thank the Gioia Tauro port managers Carmine Crudo and Vincenzo Perri for their kind availability. Thanks are also due to the referees for their valuable comments.

References

Christiansen, M., Fagerholt, K., and Ronen, D. (2004). Ship routing and scheduling: Status and perspectives. *Transportation Science*, 38:1–18.

Cordeau, J.-F., Gaudioso, M., Laporte, G., and Moccia, L. (2005a). A memetic heuristic for the generalized quadratic assignment problem. *INFORMS Journal on Computing*. Forthcoming.

Cordeau, J.-F., Laporte, G., Legato, P., and Moccia, L. (2005b). Models and tabu search heuristics for the berth allocation problem. *Transportation Science*. Forthcoming.

Daganzo, C. F. (1990). The productivity of multipurpose seaport terminals. *Transportation Science*, 24:205–216.

De Castilho, B. and Daganzo, C. F. (1993). Handling strategies for import containers at marine terminals. *Transportation Research*, 27B:151–166.

Drezner, Z. (2003). A new genetic algorithm for the quadratic assignment problem. *INFORMS Journal on Computing*, 15:320–330.

Frieze, A. and Yadegar, J. (1983). On the quadratic assignment problem. *Discrete Applied Mathematics*, 5:89–98.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13:533–549.

Holland, J. (1992). *Adaptation in Natural and Artificial Systems*. The MIT Press.

Lee, C. and Ma, Z. (2003). The generalized quadratic assignment problem. Technical report, Department of Mechanical and Industrial Engineering, University of Toronto.

Moscato, P. and Cotta, C. (2003). A gentle introduction to memetic algorithms. In Glover, F. and Kochenberger, G. A., editors, *Handbook of Metaheuristics*, pages 105–144. Kluwer, Boston.

Padberg, M. W. and Rijal, M. P. (1996). *Location, Scheduling, Design and Integer Programming*. Kluwer, Boston.

Steenken, D., Voss, S., and Stahlbock, R. (2004). Container terminal operation and operations research - a classification and literature review. *OR Spectrum*, 26:3–49.

Taleb-Ibrahimi, M., Castilho, B. D., and Daganzo, C. F. (1993). Storage space versus handling work in container terminals. *Transportation Research*, 27B:13–32.

UNCTAD (2004). Review of maritime transport. Technical report, United Nations, New York and Geneva.

Vis, I. F. A. and Koster, R. D. (2003). Transshipment of containers at a container terminal: An overview. *European Journal of Operational Research*, 147:1–16.