
Solving the 3D container ship loading planning problem by representation by rules and meta-heuristics

Anibal Tavares de Azevedo*

Applied Science Faculty,
State University of Campinas,
Rua Pedro Zaccaria, 1300, Limeira, Brazil
E-mail: atanibal@gmail.com

*Corresponding author

Cassilda Maria Ribeiro and
Galenó José de Sena

Mathematics Department,
State of São Paulo University,
Av. Dr. Ariberto Pereira da Cunha,
333, Guaratinguetá, Brazil
E-mail: cassilda@feg.unesp.br
E-mail: gsena@feg.unesp.br

Antônio Augusto Chaves and
Luis Leduíno Salles Neto

Department of Science and Technology,
Federal University of São Paulo,
Rua Talim, 330, São José dos Campos,
São Paulo, Brazil
E-mail: antonio.chaves@unifesp.br
E-mail: luiz.leduino@unifesp.br

Antônio Carlos Moretti

Applied Science Faculty,
State University of Campinas,
Rua Pedro Zaccaria, 1300, Limeira, Brazil
E-mail: moretti@ime.unicamp.br

Abstract: This paper formulates the 3D containership loading planning problem (3D CLPP) and also proposes a new and compact representation to efficiently solve it. The key objective of stowage planning is to minimise the number of container movements and also the ship's instability. The binary formulation of this problem is properly described and an alternative formulation called Representation by Rules is proposed. This new representation is combined with three metaheuristics – genetic algorithm, simulated annealing, and beam search – to solve the 3D CLPP in a manner that

ensures that every solution analysed in the optimisation process is compact and feasible.

Keywords: 3D container ship stowage; combinatorial optimisation; meta-heuristics.

Reference to this paper should be made as follows: Tavares de Azevedo, A., Ribeiro, C.M., de Sena, G.J., Chaves, A.A., Neto, L.L.S. and Moretti, A.C. (2014) ‘Solving the 3D container ship loading planning problem by representation by rules and meta-heuristics’, *Int. J. Data Analysis Techniques and Strategies*, Vol. 6, No. 3, pp.228–260.

Biographical notes: Anibal Tavares de Azevedo received his BSc in Applied Computational Mathematics in 1999, as well as MSc and PhD in Electrical Engineering in 2002 and 2006 from the State University of Campinas (UNICAMP), Campinas, Brazil. Currently, he is a Professor at UNICAMP. His research interests are combinatorial optimisation, heuristics, logistics and transportation. Two of his recent publications are: ‘Automatic guided vehicle simulation in MATLAB by using genetic algorithm’, *MATLAB for Engineers – Applications in Control, Electrical Engineering, IT and Robotics*, ‘Problem of assignment cells to switches in a cellular mobile network via beam search method’, *WSEAS Transaction on Communications Systems*.

Cassilda Maria Ribeiro received her BSc in Mechanical Engineering from State University of São Paulo, Brazil, in 1982 and MSc in Electrical Engineering in 1985 from the State University of Campinas (UNICAMP), Campinas, Brazil. She received her PhD in Industrial Automation and Informatics from the Laboratoire D’analyse Et D’architecture Des Systèmes-LAAS-Toulouse-France. Her research interests are linear and non-linear programming, mixed and integer optimisation, industrial automation and production planning and control. Her recent publications are: ‘Problem of assignment cells to switches in a cellular mobile network via beam search method’, *WSEAS Transaction on Communications Systems*.

Galen José de Sena received his BSc in Mechanical Engineering in 1983, as well as MSc and PhD in Informatics in 1987 and 1992 from the Pontific Catholic University of Rio de Janeiro (PUC/RJ), Rio de Janeiro, Brazil. He was a Post-Doctoral Researcher at the National Institute of Multimedia Education – NIME at Chiba-shi, Japan from 2003 to 2004. Currently, he is a Professor at the State University of São Paulo (UNESP). His research interests are computation, learning based on projects, new learning methodologies and techniques and new technologies in education and web environment. His recently publications are: ‘Data information system to promote the organisation data collections modeling considerations by the unified modeling language’, *Revista de Gestão da Tecnologia e Sistemas de Informação*, and ‘Scientific data dissemination – a data catalogue to assist research organisation’, *Ciência da Informação*.

Antônio Augusto Chaves received his BSc in Applied Computation in 2003, as well as PhD in Applied Computation (2009) from the National Institute of Space Research (INPE), São José dos Campos, Brazil. Currently, he is a Professor at the Federal University of São Paulo (UNIFESP) São José dos Campos campus. His research interests are computation, learning based on operations research, combinatorial optimisation problems, heuristics and meta-heuristics. His recent publications are: ‘Hybrid evolutionary algorithm for the capacitated centered clustering problem’, *Expert Systems with Applications* and ‘Clustering search algorithm for the capacitated centered clustering problem’, *Computers & Operations Research*.

Luiz Leduíno de Salles Neto received his BSc in Applied Computational Mathematics in 1997, as well as MSc in Mathematics and PhD in Applied Mathematics in 2000 and 2005 from the State University of Campinas (UNICAMP), Campinas, Brazil. He was a Post-Doctoral Researcher at the Sevilla University during 2010. Currently, he is a Professor at the Federal University of São Paulo (UNIFESP). His research interests are operation research, cut and packing problems, combinatorial, non-linear, non-linear and multi-objective optimisation. Four of his recent publications are: ‘Application of genetic algorithm to minimise the number of objects processed and setup in a one-dimensional cutting stock problem’, *International Journal of Applied Evolutionary Computation*, ‘Non-linear cutting stock problem’, *Integración – UIS*, ‘A genetic symbiotic algorithm applied to the one-dimensional cutting stock problem’, *Pesquisa Operacional*, and ‘A genetic symbiotic algorithm applied to the cutting stock problem with multiple objectives’, *Advanced Modeling and Optimization*.

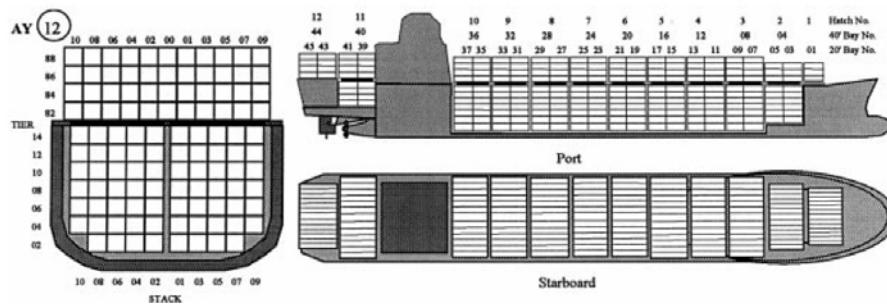
Antônio Carlos Moretti received his BSc in Computation Science in 1980, as well as MSc in Applied Mathematics in 1983 from the State University of Campinas (UNICAMP), Campinas, Brazil. He received his MSc and PhD in Industrial Engineering and Systems in 1989 and 1992 from the Georgia Institute of Technology, USA. His research interests are operation research, cut and packing problems, combinatorial, non-linear, non-linear and multi-objective optimisation. Three of his recently publications are: ‘Non-linear cutting stock problem’, *Integración – UIS*, ‘A genetic symbiotic algorithm applied to the one-dimensional cutting stock problem’, *Pesquisa Operacional*, and ‘A genetic symbiotic algorithm applied to the cutting stock problem with multiple objectives’, *Advanced Modeling and Optimization*.

1 Introduction

According to Vacca et al. (2007), there are three planning levels concerning terminal port operation optimisation: The strategic level, the tactical level and the operational level. The operational level involves real-time decisions for the following problems: berth allocation, quay crane scheduling, yard operations, transfer operations, and ship stowage planning.

The development of an efficient method for solving the containership stowage planning problem can be derived from observing the cellular structure of the containership as shown by Figure 1.

Figure 1 The containership cellular structure from Wilson and Roach (2000)



Containership capacity is measured in terms of TEU (*20-foot Equivalent Units*). For example, a ship with 8,000 TEUs is able to carry at least 8,000 20 foot deep containers. The containership cells are grouped in Sections, or Bays, where the containers are stacked in vertical stacks. A bay is a group of cells where a certain number of containers are organised in a series of vertical stacks. Generally, each bay can handle containers that are 40 or 20 feet deep. The bay can be organised in horizontal lines, or rows numerated with $r = 1, 2, \dots, R$, where row one represents the bottom of the stack and row R is on the top of the stack, and vertical lines, or columns numerated with $c = 1, 2, \dots, C$, where column 1 is the leftmost one.

As a consequence, the containers located on the top of the stack have to be moved in order to allow the unloading of the containers located underneath them. According to Dubrovsky et al. (2002), the charge for moving containers can be high, costing about \$200 per container to move. Thus, the objective of containership stowage planning is to minimise the number of unnecessary movements. It is also important to point out that other criteria must be observed such as containership stability (the weight of the containers), and type of containers (standard, hazardous, etc) (Ambrosino et al., 2006; Avriel et al., 1998; Wilson and Roach, 2000).

The containership's special structure allows ship stowage planning to be solved as a special case of the cutting and packing problem, in particular, as a three-dimensional (Orthogonal)-bin-packing problem (Dyckhoff, 1990; Wäscher et al., 2007).

Avriel et al. (1998) showed that the 2D containership loading planning problem (CLPP) can be formulated as a 0–1 binary linear model, whose optimal solution can be found by assuming that the number of containers to be shipped, along with their origin and the destination ports, are known in advance. However, this binary model is limited to small instances, since the number of binary variables and constraints grows with the number of ports and containership dimensions.

Avriel et al. (2000) also proved that the 2D CLPP is NP-Complete, which motivates the development of a series of heuristics and metaheuristics to obtain good solutions for this problem. Avriel et al. (1998) had proposed the suspensory heuristic procedure, which avoids the binary model problem by observing only the containership arrangement and the containers that must be loaded for each specific port. They also described how to generate instances for the CLPP.

Dubrovsky et al. (2002) developed a genetic algorithm (GA) with compact solution encoding, which consists of a string with sections equal to the number of ports, where each section is composed of four vectors related to the number of unload movements that must be done. Also, instability constraints were considered as a penalty function and the proposed approach was tested in instances with 11 ports and a containership with a 1,000 TEU capacity.

Wilson and Roach (1999, 2000) developed a methodology that solves stowage planning in two steps. The first step employs a branch-bound to solve the problem of assigning generalised containers to a bay. Then, the second step consists in using Tabu search to assign each container a specific position in each bay. This approach takes two hours to obtain a solution for a 688 TEU containership and 4 destinations.

Ambrosino et al. (2006, 2010) also introduced and tested the notion of stability by considering that each container has a weight in the model. Ambrosino et al. (2010) also tested ant colony optimisation (ACO) in a real problem for the SECH Terminal of Genova, Italy, and a containership whose capacity is 1,800 TEUs for instances with two

or three destinations and 4,920 or 5,510 TEUs for instances with four or five destinations, respectively.

Imai et al. (2006) presented a multi-criteria optimisation method to the ship stowage problem by considering two conflicting objectives: the ship's stability and the number of container re-handles. For the ship stability measurements, the GM, list, and trim were used and for the container re-handles, an estimated number of re-handles was used. Computational experiments were carried out with a proposed GA for instances of 504 and 2,016 TEUs, and three and four destinations, respectively.

Sciomachen and Tanfani (2007) formulated the problem as a three-dimensional bin packing problem and presented a heuristic solution method which was based on this relation. Objectives were to minimise the total loading time as well as to efficiently use the quay equipment. The approach was validated by using real test cases from the port of Genova (Italy), namely: a containership with a 198-TEU capacity, containers with different weights, one or two cranes, and two or three destinations.

Fan et al. (2010) provided stowage planning with a tradeoff between crane workload balance and ship stability and solved instances in which the containership had a 5,000-TEU capacity, five quay cranes, five types of containers, and five destination ports.

For such complex problem, this paper proposes a two-level solution to minimise the unnecessary container movements and also ship instability. In the first level, the meta-heuristic manipulates the encoded solutions. In the second level, the solutions are evaluated by a decoding algorithm that simulates the loading and unloading of a containership at each port. In this encoding and decoding procedure the size of the search space can be reduced and human knowledge may be properly incorporated. This encoding and decoding is called representation by rules.

This paper is organised as follows. Section 2 presents the CLPP mathematical model. Section 3 explains the encoding, using meta-heuristic methods, and their advantages. Section 4, Section 5, and Section 6 detail the GA, simulated annealing (SA), and the beam search (BS) method, respectively. Section 7 presents and discusses the computational results and Section 8 presents the conclusions and possible future work.

2 Mathematical model

For the sake of simplicity, without loss of generality, it will be considered that:

- a the containership has a rectangular format and can be represented by a matrix with rows ($r = 1, 2, \dots, R$), columns ($c = 1, 2, \dots, C$) and bays ($d = 1, 2, \dots, D$) with maximum capacity of $R \times C \times D$ containers
- b the containers are all the same size and weight
- c the ship starts to be loaded in Port 1, where it arrives empty
- d the ship visits Ports 2, 3, ..., N such that the containership will be empty in the last port, because the ship performs a circular route where port N , in fact, represents Port 1
- e in each port $i = 1, 2, \dots, N$, the containership can be loaded with containers whose destination are ports $i + 1, \dots, N$

- f the containership can always carry all the containers available in each port and this will never exceed the maximum containership capacity.

In addition to conditions (a) to (f), the number of containers that must be loaded at a certain port is given by a transportation matrix T of dimension $(N - 1) \times (N - 1)$, whose element T_{ij} represents the number of containers from port i that must be transported to the destination port j . This matrix is an upper triangular matrix, since $T_{ij} = 0$ for every $i \geq j$.

The mathematical model in terms of linear programming with binary variables for the 3D CLPP is given by (1) to (6).

- Min

$$f(x) = \alpha \phi(x) + (1 - \alpha) \phi(y) \quad (1)$$

$$i = 1, \dots, N - 1, j = i + 1, \dots, N;$$

- S.A.

$$\sum_{v=i+1}^j \sum_{r=1}^R \sum_{c=1}^C \sum_{d=1}^D x_{ijv}(r, c, d) - \sum_{k=1}^{i-1} \sum_{r=1}^R \sum_{c=1}^C \sum_{d=1}^D x_{kji}(r, c, d) = T_{ij} \quad (2)$$

$$i = 1, \dots, N - 1, j = i + 1, \dots, N;$$

$$\sum_{k=1}^i \sum_{j=i+1}^N \sum_{v=i+1}^j x_{kjv}(r, c, d) = y_i(r, c, d) \quad (3)$$

$$i = 1, \dots, N - 1, r = 1, \dots, R; \\ c = 1, \dots, C; d = 1, \dots, D;$$

$$y_i(r, c, d) - y_i(r + 1, c, d) \geq 0 \quad (4)$$

$$i = 1, \dots, N - 1, r = 1, \dots, R - 1; \\ c = 1, \dots, C; d = 1, \dots, D;$$

$$\sum_{i=1}^{j-1} \sum_{p=j}^N x_{ipj}(r, c, d) + \sum_{i=1}^{j-1} \sum_{p=j+1}^N \sum_{v=j+1}^p x_{ipv}(r + 1, c, d) \leq 1 \quad (5)$$

$$j = 2, \dots, N, r = 1, \dots, R - 1; \\ c = 1, \dots, C; d = 1, \dots, D;$$

$$x_{ijv}(r, c, d) = 0 \text{ or } 1; y_i(r, c, d) = 0 \text{ or } 1; \quad (6)$$

where the binary variable $x_{ijv}(r, c, d)$ is defined as follows: if, in port i , the compartment (r, c, d) has a container whose destination is port j and this container was moved in port v , then the variable assumes value 1; otherwise value 0 is assumed. The term compartment (r, c, d) represents row r , column c for the containership bay d . Similarly, variable $y_i(r, c, d)$ is defined as follows: if, in port i , the compartment (r, c, d) has a container; then the variable assumes value 1; otherwise value 0 is assumed.

The objective function (1) is composed of two terms: the first is the total cost of moving a container and, the second is the sum of instability measures for the containership configuration in each port. It is assumed that, for all ports, the container

movement costs the same and is equal to one. Constraint (2) is related to the container conservation flow. In other words, the total number of containers in the ship in a port i must be equal to the number of containers that have been loaded in all ports $p = 1, 2, \dots, i$ minus the total number of containers unloaded in all ports $p = 1, 2, \dots, i$. Constraint (3) requires that each compartment (r, c, d) of the containership is always occupied by at most one container. Constraint (4) is related to the physical storage of the containers in the ship, and it imposes that, for each container in row $r + 1$, there be another container in the row r for all $r = 1, \dots, R - 1$. Constraint (5) defines how a container can be unloaded from the ship in port j by requiring that, if a container occupies the position (r, c, d) at port j , and it will be unloaded, then, there are no containers above or the containers above have already been unloaded at previous ports.

The two terms which compose the objective function [equation (1)] define two optimisation criteria: the first term is a function of the number of containers moved, $\phi_1(x)$, and the second depends on how the containership is organised in each port, $\phi_2(x)$. The two criteria are combined by values given by each weight α and $(1 - \alpha)$ in a manner that forms a bi-objective optimisation framework.

The term $\phi_1(x)$ assumes that for all ports, the container movement cost is the same and is equal to one which may be translated as equation (7).

$$\phi_1(x) = \sum_{i=1}^{N-1} \sum_{j=i+1}^N \sum_{v=i+1}^{j-1} \sum_{r=1}^R \sum_{c=1}^C \sum_{d=1}^D x_{ijv}(r, c, d) \quad (7)$$

Term $\phi_2(y)$ refers to the containership's stability and assumes that every container has the same mass and is equal to one. This term measures the distance between the mass and the geometric centre of the containership for each bay in every port as described by equation (8).

$$\phi_2(y) = \sum_{i=1}^N \left(\sum_{d=1}^D (\Delta zcm_{di})^2 + \sum_{d=1}^D (\Delta xcm_{di})^2 \right) \quad (8)$$

where

$$\Delta zcm_{di} = zcm_{di} - R / 2,$$

$$\Delta xcm_{di} = xcm_{di} - C / 2,$$

$$zcm_{di} = \frac{\left(\sum_{r=1}^R \sum_{c=1}^C (y_i(r, c, d) \bullet (r - 0.5)) \right)}{\left(\sum_{r=1}^R \sum_{c=1}^C y_i(r, c, d) \right)},$$

$$xcm_{di} = \frac{\left(\sum_{c=1}^C \sum_{r=1}^R (y_i(r, c, d) \bullet (c - 0.5)) \right)}{\left(\sum_{r=1}^R \sum_{c=1}^C y_i(r, c, d) \right)}.$$

This mathematical model is the first that extends the 2D CLPP proposed by Avriel and Penn (1993). Avriel and Penn (1993) showed that the 2D CLPP is NP-complete; so, the

3D CLPP is also. Another drawback of the 2D or 3D model for real problems is related to solution encoding, which demands a large number of binary variables to represent a solution, so that they may only be used for small instances in practice. For example, a 3D CLPP solution that produces a complete stowage plan through N ports for a problem instance with a containership with a $(R \times C \times D)$ bay dimension demands an amount $(R \times C \times D \times N^3)$ of $x_{ijr}(r, c, d)$ variables and $(R \times C \times D \times N)$ of $y_i(r, c, d)$ variables. That means that with $D = 5$, $R = 6$, $C = 50$ and $N = 30$, there will be 40,500,000 $x_{ijr}(r, c, d)$ variables and 45,000 $y_i(r, c, d)$ variables to represent just one solution. This fact justifies not only the use of heuristics for the 3D CLPP, but also a different way to represent the solution.

The next section shows how to represent a solution without using a binary model that leads to a large number of variables. The Section describes a special encoding/decoding approach which combines rules that describe how to load and unload a containership, and a simulation procedure. This encoding will be called Representation by rules.

3 The representation by rules approach

An alternative to the binary mathematical model, presented in Section 2, is the representation by rules approach. It can be better explained if the containership is represented by a graph, as shown in Figure 2. In that Figure, the node represents Port p , where the containership is docked. The containership state changes when it arrives and leaves Port p and those changes are represented by the arcs labelled with x_p and x_{p+1} , respectively. The x_p state will turn into the x_{p+1} state by the following two decisions.

- a How the existing containers will be unloaded in port p . This decision is represented by u_p . The u_p decision can be seen as the result of two other decision variables:
 - the q_p variable represents the containers that must be unloaded, because their destination is Port p or because they are blocking containers with a Port p destination
 - the v_p variable represents the containers that can be unloaded for a better containership arrangement that minimises the number of unnecessary movements at future ports
- b How to reload the ship with containers from Ports $1, \dots, p - 1$, whose destinations are Ports $p + 1, \dots, P$, and how to load the containers from Port p . This decision is represented by y_p . Note that the variables q_p and v_p will have a great influence on how many containers will be reloaded into the ship.

Figure 2 The containership and possible decisions represented by a graph with a node and arcs

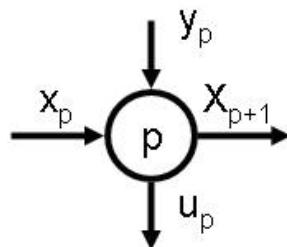


Figure 2 is important to make clear that the arrangement in the containership will be determined by two decisions at each port: how to unload and load the ship. In real life, containership unloading and loading is done according to experience or rules of thumb. In other words, the loading or unloading process follows existent rules. In the future, these rules can be translated into a computational program that simulates the required behaviour.

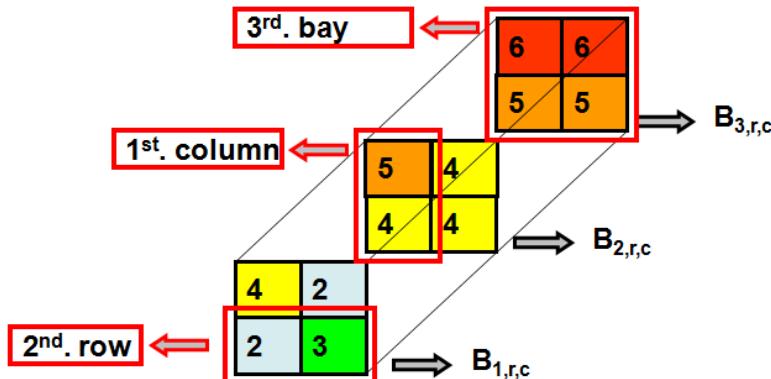
Representation by Rules is an interesting alternative to the binary mathematical model. The solutions with this encoding are always feasible since they must obey the proposed scheme given by Figure 2, and this ensures the feasibility of Constraint (2). In other words, it ensures that the arrangement in the containership when leaving each Port p (x_{p+1}) depends on the containership arrangement when just arriving at Port p (x_p), plus how many containers are unloaded (u_p) and loaded (y_p) at Port p . In fact, Constraint (2) represents a kind of container conservation law. The feasibility of constraints (3) to (6) will be ensured by incorporating them into the computational simulation related to every rule.

The computational implementation of representation by rules depends on the definition of two elements.

- Since the containership has a cellular structure, as shown in Figure 1, the containership arrangement state can be represented by a B matrix, called a state matrix, and this matrix represents the variable related to the containership state (x_p and x_{p+1}).
- The changes in the containership state made by the application of the unloading rules (UR) and loading rules (LR), represented by u_p and y_p , respectively, can be performed using a computational simulation procedure.

Matrix B represents the containership arrangement, since each element of B is represented by B_{drc} , and it describes whether there is a container whose destination is Port p in the cell located in row r , column c and Bay d , if $B_{drc} = p$; if B_{drc} is empty, then B_{drc} is 0. To better illustrate this, a B matrix where $D = 3$ and $R = C = 2$ is shown in Figure 3.

Figure 3 State matrix B representing the container arrangement in a 12-container containership to 6-ports (see online version for colours)

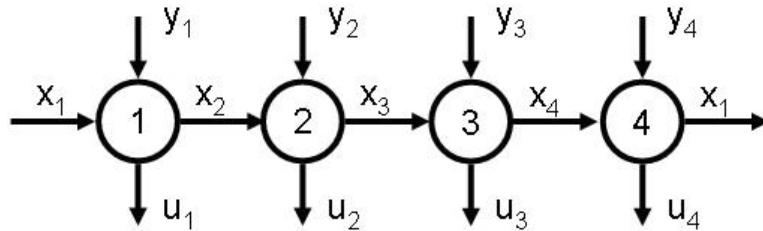


In Figure 3, row 2 represents the bottom of the containership and row 1 represents the top of the containership. Thus, element (1, 1, 1) is equal to 4, which means that this cell is occupied by a container whose destination is Port 4. Using the same criteria, element (3, 2, 2) is equal to 5 and it means that this cell is occupied by a container whose destination is Port 5.

Supposing that matrix B in Figure 3 represents the Containership in Port 2, in order to unload this Containership, it is necessary to move the containers located in cells (1, 2, 1), and (1, 1, 2). However, observe that the container in cell (1, 2, 1) can only be unloaded if the container located in cell (1, 1, 1) is unloaded too; even though the destination of this container is Port 4. The goal of the CLPP is to minimise the number of movements of this kind by performing an adequate arrangement of the containership bay at every port.

The representation by rules approach treats the CLPP as a problem where matrix B represents the Containership arrangement (x_p) before arriving in Port p . It will be modified at each port by deciding how to perform the unloading (u_p) and loading operations (y_p) by defining an unloading and a LR, respectively. It is also important to stress that Constraint (2) connects decisions from port to port. The choice of an UR or LR for Port 2 can indirectly influence the containership arrangement in Port 4, since the graph representation of the CLPP takes the form shown in Figure 4.

Figure 4 The graph representation for the CLPP with four-port destination



In the next sections, the LR and UR proposed to solve the CLPP will be presented. Also, a description of how the LR and UR affect the containership arrangement will be presented supposing the containership dimensions are the same used in Figure 3.

3.1 Loading rules

- *Loading rule LR1:* this rule fills matrix B row by row, from left to right, starting from the bottom row for each bay in a manner that the containers with the farthest destination are placed on the lowest rows and each bay is filled before the next. Figure 5 shows the application of the LR for the containership at Port 1.
- *Loading rule LR2:* this rule fills matrix B row by row, from left to right, starting from the first bay and filling only one row per bay in a manner that the containers with the farthest destination are placed on the lowest rows and distributed among the bays. Figure 6 shows the application of the LR for the containership in Port 1.

- *Loading rule LR3*: this rule is the reverse of *LR1* which means matrix B is filled row by row, from right to left, starting from the bottom row for each bay in a manner that the containers with the farthest destination are placed on the lowest rows and each bay is filled before the next is begun. Figure 7 shows the application of this LR for the containership in Port 1.
- *Loading rule LR4*: this rule is the reverse of *LR2* in the sense that it fills matrix B row by row, from right to left, one row per bay starting from the first bay until it reaches the last in a manner that the containers with the farthest destination are placed on the lowest rows and distributed among the bays. Figure 8 shows the application of this LR for the containership in Port 1.
- *Loading rule LR5*: this rule fills matrix B row by row from left to right with containers destined for the nearest port, starting from the first bay and continuing until the number of elements θ_p in a column is reached. The value θ_p is computed by equation (9).

$$\theta_p = \left\lceil \frac{\sum_{i=1}^p \sum_{j=p+1}^N T_{ij}}{D \times C} \right\rceil \quad (9)$$

Then another bay is filled in a manner that the containers with the nearest destination are placed first to form the stacks. Figure 9 shows this load rule with the containership in Port 2.

- *Loading rule LR6*: this rule is the reverse of *LR5* in the sense that it fills matrix B row by row from right to left with containers destined for the nearest port, starting from the first bay and continuing until the number of elements θ_p in a column is reached. The value θ_p is also computed by equation (9). Figure 10 shows this load rule with the containership in Port 2.

Figure 5 State matrix B representing the container arrangement after applying *LR1* (see online version for colours)

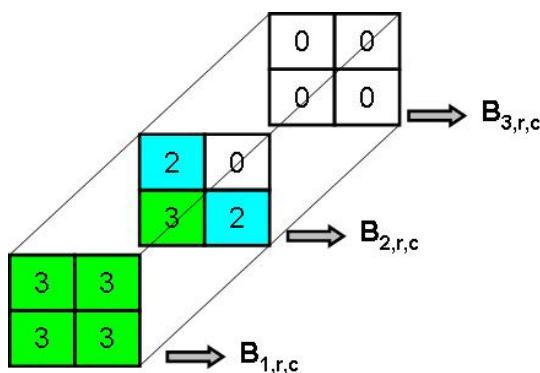


Figure 6 State matrix B representing the container arrangement after applying LR2 (see online version for colours)

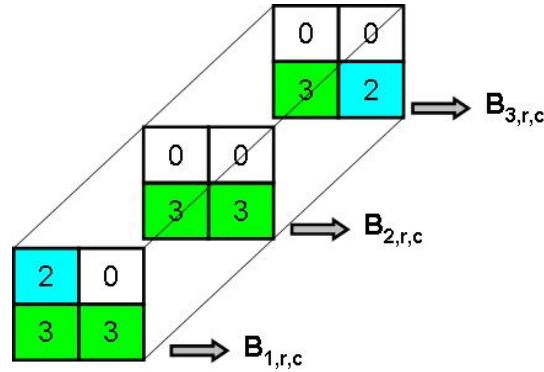


Figure 7 State matrix B representing the container arrangement after applying LR3 (see online version for colours)

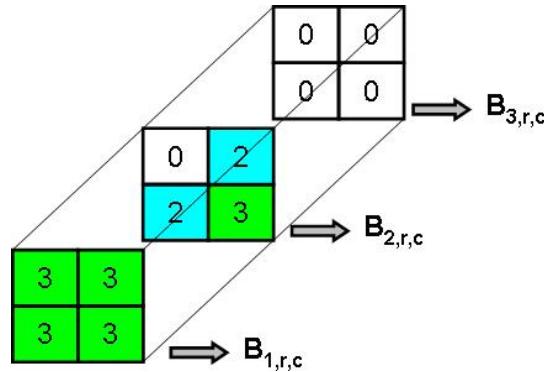


Figure 8 State matrix B representing the container arrangement after applying LR4 (see online version for colours)

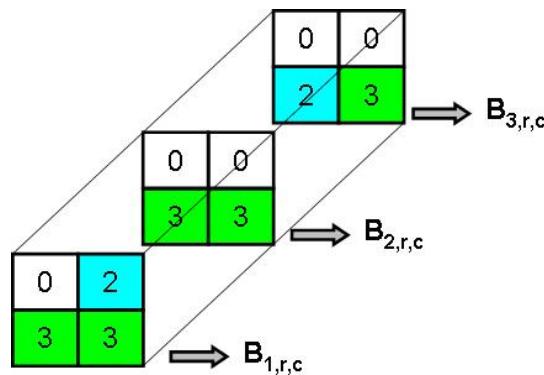


Figure 9 State matrix B representing the container arrangement after applying $LR5$ (see online version for colours)

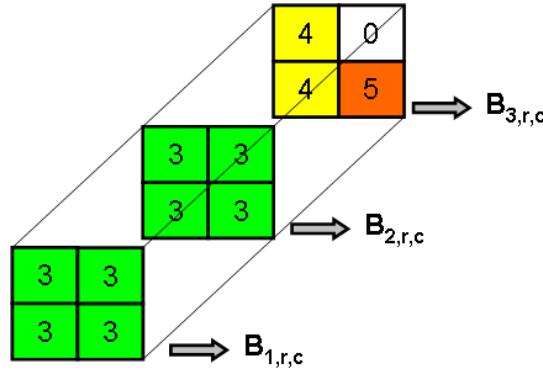
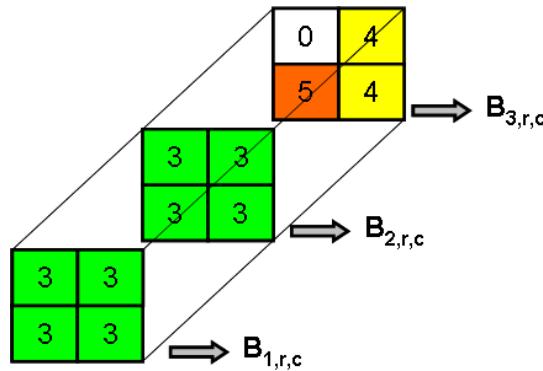


Figure 10 State matrix B representing the container arrangement after applying $LR6$ (see online version for colours)



3.2 Unloading rules

- *Unloading Rule UR1:* suppose that the containership arrives at Port p . This rule will only remove the containers whose destination is Port p , and all the ones that are blocking the stacks.
- *Unloading Rule UR2:* this rule states that the containership must unload every container when arriving at a specific Port p , in a manner that it allows a complete rearrangement of every stack.

3.3 Combining LR and UR

In order to avoid the need of using two values to indicate which LR and UR will be used for every port it is possible to simplify the encoding by defining the various combinations of the LR and UR. The specific combination of LR and UR for Port p is defined as a rule.

To better illustrate the rule concept, the six LR and the two UR described before can be combined to generate new 12 rules. Table 1 illustrates all the rules created as a result of these LR and UR combinations.

Table 1 Rules produced by the combination of LR and UR

<i>Load rules</i>	<i>Unload rules</i>	<i>Rule</i>
LR1	UR1	1
	UR2	2
LR2	UR1	3
	UR2	4
LR3	UR1	5
	UR2	6
LR4	UR1	7
	UR2	8
LR5	UR1	9
	UR2	10
LR6	UR1	11
	UR2	12

Table 1 shows the various combinations of LR and UR. For example, Rule 2 is a combination of the unloading rule *UR2* and the loading rule *LR1*. This encoding allows the representation of a solution for the CLPP that employs a vector whose number of elements is equal to the number of ports.

To adopt this encoding process it is necessary to define a translation scheme which simulates the unloading and loading actions proposed by the rules. This translation extracts the unloading and LR that should be applied for the containership arrangement simulation in every port, as described in Figure 11. Afterwards, the scheme gives the number of containers moved in every port and total instability measure.

The various rules for the loading and unloading functions, used in the algorithm described in Figure 11, are detailed in Section 3.1 and Section 3.2, respectively. The calcDXDZ procedure computes the sum of the distance between the mass and the geometric centre of the containership in each bay *d* in each port as shown in Figure 12.

Defining x_{cm3} as the mass centre row coordinate for Bay 3 and z_{cm3} as the mass centre column coordinate for Bay 3; then the x_{cm3} and z_{cm3} values are computed using equation (8) and the information given in Figure 12.

$$\begin{aligned}x_{cm3} &= (2 \times 0.5 + 2 \times 1.5 + 2 \times 2.5) / (2 + 2 + 2) \\&= 9 / 6 = 1.5\end{aligned}$$

$$\begin{aligned}z_{cm3} &= (3 \times 0.5 + 3 \times 1.5) / (2 + 2 + 2) \\&= 6 / 6 = 1.0\end{aligned}$$

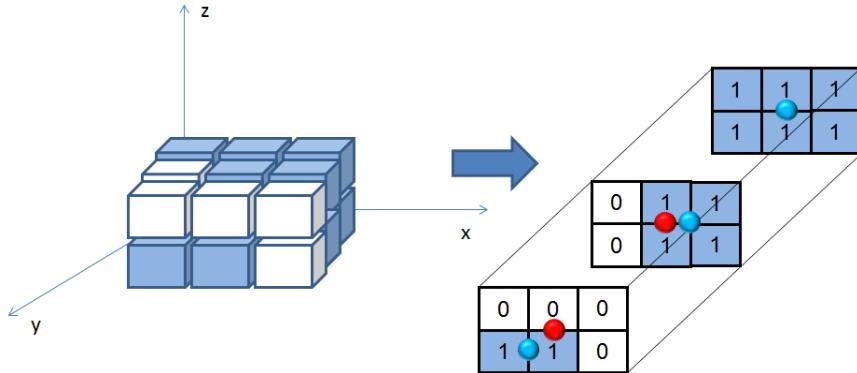
Figure 11 The translation scheme gives the number of containers moved through the ports and total instability measure combined with weights alpha and beta for a solution vector s

```

Translation Scheme
Begin
    p ← 1, nmov ← 0, instab ← 0
    start(B)
    while (p < N) do
        start
        [ur, lr] = extractRules(s(p))
        if (p > 1)
            [aux, B, T] = unloading(ur, B, p)
            instab ← instab + calcDXDZ(B)
            nmov ← nmov + aux
        end
        if (p < N-1)
            [aux, B, T] = loading(lr, B, T, p)
            instab ← instab + calcDXDZ(B)
            nmov ← nmov + aux
        end
        p ← p + 1
    end
    return alpha*nmov + beta*instab
End

```

Figure 12 The container arrangement and the corresponding mass centre (in blue) and geometric centre (in red) by Bay (see online version for colours)



The same computation is done for Bay 2 in order to obtain x_{cm2} and z_{cm2} values by using equation (8) and Figure 12.

$$\begin{aligned}
 x_{cm2} &= (0 \times 0.5 + 2 \times 1.5 + 2 \times 2.5) / (0 + 2 + 2) \\
 &= 8 / 4 = 2.0 \\
 z_{cm2} &= (2 \times 0.5 + 2 \times 1.5) / (4) \\
 &= 4 / 4 = 1.0
 \end{aligned}$$

Finally, for Bay 1 coordinates x_{cm1} and z_{cm1} may be computed using equation (8) and Figure 12.

$$\begin{aligned} xcm_l &= (1 \times 0.5 + 1 \times 1.5 + 0 \times 2.5) / (1 + 1 + 0) \\ &= 2 / 2 = 1.0 \end{aligned}$$

$$\begin{aligned} zcm_l &= (2 \times 0.5 + 0 \times 1.5) / (2) \\ &= 1 / 2 = 0.5 \end{aligned}$$

The instability measure, thus, becomes:

$$\begin{aligned} \phi(y) &= ((1.5 - 1.5)^2 + (2.0 - 1.5)^2 + (1.0 - 1.5)^2) \\ &\quad + ((1.0 - 1.0)^2 + (1.0 - 1.0)^2 + (0.5 - 1.0)^2) \\ &= 0.75 \end{aligned}$$

The translation scheme shown in Figure 11 will make it easy to employ meta-heuristic methods like the GA described in Section 4, since each solution will be represented by just one vector and each vector is always associated to a feasible solution. To better explain how this can be done, Subsection 3.4 explains how one solution is decoded through the translation scheme.

3.4 An evaluation of one solution using the translation scheme

Suppose the containership's dimensions are like the ones presented in Figure 10 (with three bays, two rows, and two columns) and the transportation matrix T has the information given in Table 2.

Table 2 The data for transportation matrix B

	$D2$	$D3$	$D4$
O1	2	4	3
O2	0	1	3
O3	0	0	4

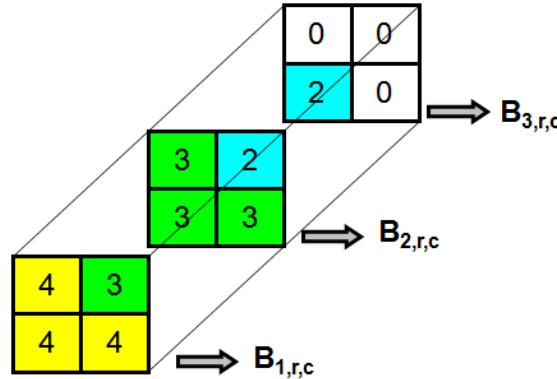
Each element $T(i, j) = nc$ of Table 2 indicates that the number of nc containers will be shipped from Port (row) i to Port (column) $j + 1$. For example, element $T(2, 3) = 3$ means that in the second Port, three containers whose destination is Port four must be loaded.

Now if a vector $v = [1 5 3]$ solution is proposed for this problem, it may be evaluated in terms of the number of movements and also the instability measure by using the translation scheme as follows.

3.4.1 Port 1

3.4.1.1 Loading

According to the solution vector v , $v[1] = 1$, for Port 1, Rule 1 should be applied which means *LRI* should be applied, according to Table 1, as shown in Figure 13.

Figure 13 State matrix B after applying LR1 (see online version for colours)

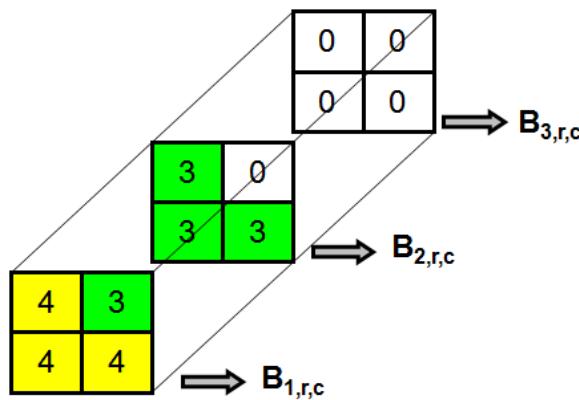
After this operation, the variables that measure the solution quality will be:

- instab = $2 * (0.5)^2 = 0.5$
- nmov = 9

3.4.2 Port 2

3.4.2.1 Unloading

According to the solution vector v , $v[2] = 5$, for Port 2 Rule 5 should be applied, which means *UR1* should be applied, according to Table 1, as shown in Figure 14.

Figure 14 State matrix B after applying UR1 (see online version for colours)

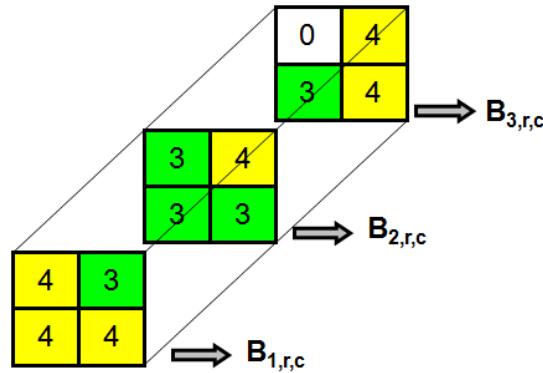
After this operation, the variables that measures solution quality will be:

- instab = $0.5 + (2 * 0.166^2 + 2 * 1.0) = 0.5 + (0.055 + 2.0) = 2.555$
- nmov = 9 + 2 = 11

3.4.2.2 Loading

According to the solution vector v , $v[2] = 5$, for Port 2 Rule 5 should be applied, which means *LR3* should be applied, according to Table 1, as shown in Figure 15.

Figure 15 State matrix B after applying *LR3* (see online version for colours)



After this operation, the variables that measure solution quality will be:

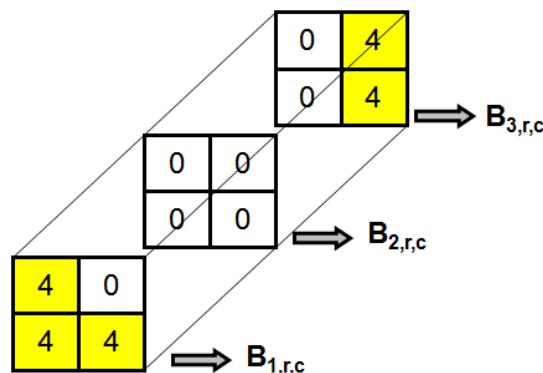
- $\text{instab} = 2.555 + (2 * 0.166^2) = 2.555 + 0.055 = 2.610$
- $\text{nmov} = 11 + 4 = 15$

3.4.3 Port 3

3.4.3.1 Unloading

According to the solution vector v , $v[3] = 3$, for Port 3 Rule 3 should be applied, which means *UR1* should be applied, according to Table 1, as shown in Figure 16.

Figure 16 State matrix B after applying *UR1* (see online version for colours)



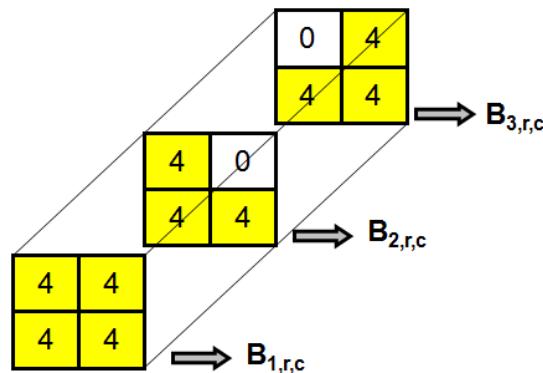
After this operation, the variables that measure solution quality will be:

- $\text{instab} = 2.610 + (2 * 0.166^2 + 2 * 1.0 + 0.5^2) = 4.91$
- $\text{nmov} = 15 + 6 = 21$

3.4.3.2 Loading

According to the solution vector v , $v[3] = 3$, for Port 3 Rule 3 should be applied, which means $LR2$ should be applied, according to Table 1, as shown in Figure 17.

Figure 17 State matrix B after applying $LR2$ (see online version for colours)



After this operation, the variables that measure solution quality will be:

- $\text{instab} = 4.91 + (2 * 0.166^2 + 2 * 0.166^2) = 5.57$
- $\text{nmov} = 21 + 5 = 26$

3.4.4 Port 4

For the last port, there is only one option, which is to unload the containers from the ship, and this is equal to the number of containers whose destination is Port 4. This will produce the following values for instab and nmov :

- $\text{instab} = 5.57$
- $\text{nmov} = 26 + 10 = 36$

This evaluation scheme for each solution will be used by the meta-heuristics explained in the next sections.

4 The GA

The GA uses a population of individuals, represented by: $A(t) = \{A_1^t, \dots, A_n^t\}$ for each generation (iteration) t , in which each individual represents a vector of rules. In the computational implementation adopted here, the population is stored in a matrix $A(t)$ and each line A_i^t represents this vector of rules. Each vector A_i^t is evaluated according to the number of movements and to the instability measure. Then, *fitness*, a measure of how successful this individual is in the problem, is calculated. *Fitness* is calculated for the entire population and is based on this new population that combines the most capable individuals that will form generation $t + 1$. During the formation of the new population, some individuals from generation t are submitted to a transformation process by genetic operators in order to form new rules. These transformations include unary operators m_i

(mutation), that allow the creation of new rules by small changes in the individual attributes ($m_i: A_i \rightarrow A_i$), and superior order transformation c_j (crossover) that produces new individuals by combining one or more individuals ($c_j: A_j \times \dots \times A_k \rightarrow A_j$). This process is carried out until a previous specified maximum number of generations is reached (Holland, 1975; Michalewicz, 1996).

The implementation details adopted for the GA are as follows:

4.1 Data structure codification for each individual

Each GA individual is associated to a set of rules by using a vector v with, for example, four elements. The values inside the elements correspond to each combination of unloading and LR (1 to 12) that will be applied at each Port (1 to 4). The set of rules for various individuals is stored in a matrix and each column represents an individual, as shown in Figure 18. In the example shown in Figure 18, the first column contains a vector v_1 with four elements, in which each value corresponds to Table 1.

Figure 18 Relation between GA individual encoding and a set of rules for each individual (column vector)

Port	Ind. 1	Ind. 2
1	6	3
2	1	5
3	12	10
4	2	7

Note: Matrix A has two vectors v_1 and v_2 .

Once the individual is defined, the population which consists of $numpop$ individuals stored in a matrix A of dimension $10 \times numpop$ can also be defined. For every instance, the value of $numpop$ equals 100 was adopted. Since each column of matrix A represents individuals/solutions, every element $A(i, j) = k$ defines what rule k (k equals 1 to 12) in Port i will be used, if the individual j is chosen. For example, $A(1, 1) = 6$ means that the individual/solution 1 in Port 1 is to apply Rule 6, which means unloading the containership with *UR2* and loading using *LR3*.

4.2 Fitness evaluation and assignment

The fitness evaluation or *fitness* is responsible for ranking the individual within the population in a generation t . Thus, fitness is constructed in a manner that the solutions with less credit measure have the higher fitness values. This explanation justifies why fitness for each individual A_i is calculated with equation (10), given as follows:

$$Fit(A_i) = f(A_i) \quad (10)$$

where $f(A_i)$ is the fitness evaluation, according to the combination of number of movements and the instability measure as defined by equation (1), equation (7), and equation (8). In other words, it translates the solution proposed by the set of rules stored

in vector A_i in terms of number of movements [equation (7)] and the instability measure [equation (8)] through a simulation of the container arrangement in the Containership as it travels from port to port.

4.3 Individual selection for the next generation

The population formation process employed is the ‘roulette wheel’, which is a random raffle, where the probability $Q(A_i)$ is used to choose the individual A_i for the next generation in a population with b individuals. The $Q(A_i)$ value can be obtained by using equation (11).

$$Q(A_i) = \text{Fit}(A_i) / \sum_{i=1}^{\text{numpop}} (\text{Fit}(A_i)) \quad (11)$$

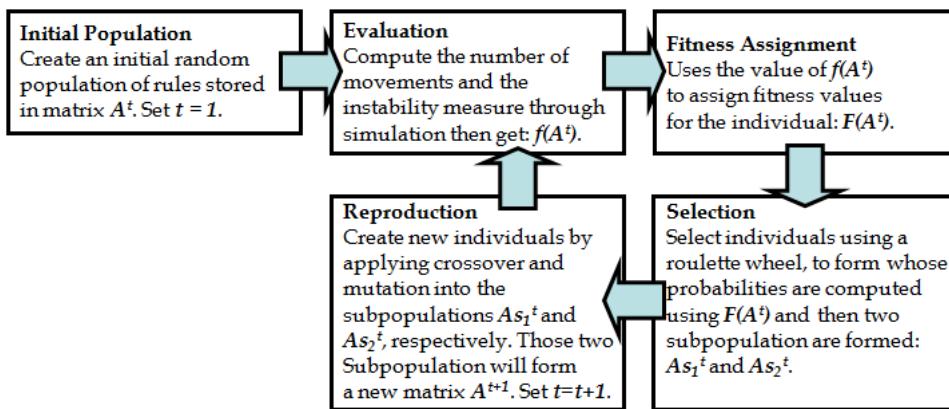
The best individual of the current generation is always kept to be in the next generation.

4.4 Crossover OX

The Crossover operators try to generate new individuals to form the next population by combining information from past generations that is present in the individuals. Here, two crossover operators are used which is described as follows.

Two individuals, A_1 and A_2 , with N elements are randomly chosen from a population in generation t . Then, an integer number δ in the interval $[1, N-1]$ is drawn and elements of A_1 that are in positions δ until N are exchanged with the elements of A_2 that are in positions δ until N . This exchange will produce two new individuals, nA_1 and nA_2 , that can appear in the next generation.

Figure 19 How the GA elements are combined into a unique algorithm (see online version for colours)



4.5 Mutation operator

The mutation operator modifies $10 * \text{numpop} * pm$ elements of matrix A , where pm is the percentage of total bits to be mutated. The selection of which element A_{ij} to be mutated

consists in randomly selecting the line index and the column index, and then changing them.

Figure 19 shows how the GA components are combined.

One important observation is that the size of subpopulations As_1^t and As_2^t are the same and are equal to 5% of the total population ($numpop$).

5 The BS method

The BS method was first used by the community of Artificial Intelligence to treat problems of speech recognition (Lowerre, 1976). The literature gives a lot of applications of this method in scheduling problems (Della Croce and T'kindt, 2002; Valente and Alves, 2005; Ow and Morton, 1988).

The BS method is an implicit enumeration method for solving combinatorial optimisation problems. It can be said that it is an adaptation of the Branch and Bound method, where only a predetermined number of best partial solutions (nodes) are evaluated at each level of the search tree while the others are discarded permanently. Since most of the tree search nodes are discarded, only a few nodes are kept for further use. Consequently, the method execution time is polynomial.

Thus, the BS is a tree search technique in which a fixed number of nodes is analysed at each level of the tree, producing a fixed number of solutions. The number of nodes analysed at each level is called beam width and is denoted by β .

Before building the search tree, it is necessary to establish the following definitions:

- D.1 The tree is constructed by level and at each level i , the assignment of Rule r is made for every Port i .
- D.2 The nodes that are in Level 1 of the tree are called seed nodes because each of them will generate a sub-tree of decisions.
- D.3 For Level i , when making the assignment of a Rule r to a Port i , two values are considered: the number of movements to unload and load, and the instability measure. Once again the rule assignments performed in previous ports will affect containership arrangement in Port i which will, in turn, indirectly affect the number of movements in Port i .

Considering that the tree has N levels (D.1), where N is the number of ports, a complete problem solution, with attribution of all R rules to all the N ports is only obtained when defining the assignments until level N .

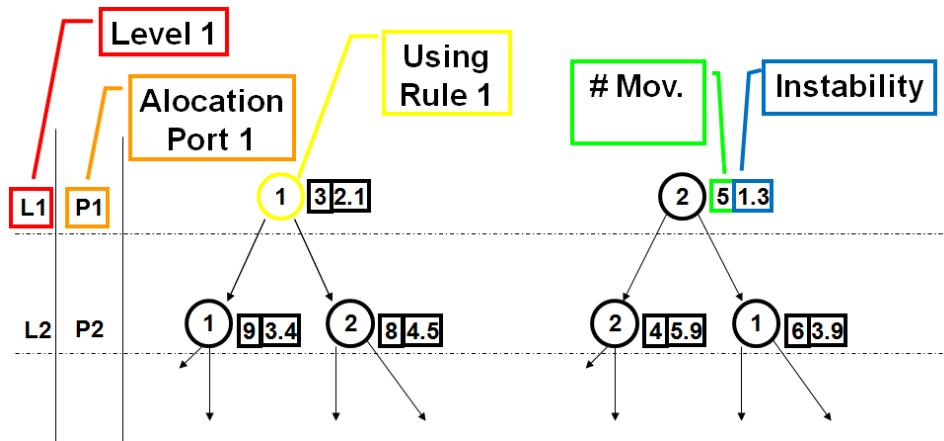
To avoid the exponential growth of the tree, the proposed algorithm does not create all the nodes of the tree. The nodes are created according to some rules. In each node of the tree, the following information is stored as:

- 1 rule identification
- 2 port identification
- 3 number of movements to unload and load containers, and the instability measure extracted from the simulation (Figure 11) which is related to the allocation made in the node

- 4 partial cost of the solution, i.e., total number of movements performed and instability measure to the current level.

The Figure 20 shows how that information is used and combined in the search tree.

Figure 20 The BS method elements (see online version for colours)



Normally, during the construction process of the solution tree, the first criterion must be applied to check whether to create a node at Level i . One node will only be created (assignment of Rule r) in Level i , if the constraints (2) to (6) hold. However, this is not the case for Representation by Rules integrated with simulation because Rule r always produces a feasible arrangement for the Containership in Port (Level) i .

After going through the test to check whether the constraints hold, the node will only remain in the solution tree if it passes in the search width criterion β . For example, if the search width chosen by the user is $\beta = 2$ in each level, only the nodes that generate the two minor solutions, computed by the technique of the greedy algorithm, remain in the tree.

Thus, for $\beta = 2$, each node will only generate two branches, and they will be those in which the greedy solution computation produces the lowest cost.

The creation process of the greedy solution is similar to the process of the tree of solutions creation, except that here, in each step, the allocation that generated the best partial solution of the node is chosen, i.e., the nodes with less costs will be part of the solution. A more complete description of the BS method and also the greedy best-first search can be found in Ribeiro et al. (2010).

6 The SA meta-heuristic

The SA meta-heuristic is a popular local search method which helps avoid getting trapped in local minima by allowing some hill-climbing movements that produce worse objective function values. The original ideas come from the Statistical Mechanics process, in particular the metal or glass annealing process, which starts with a low energy configuration when it is heated and then is slowly cooled (Blum and Roli, 2003). The SA algorithm was first proposed to be applied to solving combinatorial problems by

Kirkpatrick et al. (1983) and the following basic SA algorithm is adapted from McMullen and Frazier (2000) as shown in Figure 21.

Figure 21 Detailed SA algorithm

```

Algorithm: Simulated Annealing
begin
  //s0: initial solution;
  //T0: initial temperature;
  //α: cooling factor;
  // SAmax: maximum number of iterations to
  // achieve the thermal equilibrium;
  s* ← s0; { s* : better solution}
  s ← s0; { s : current solution}
  T ← T0; {T: current temperature}
  cont ← 0; {cont: total number of solution
  evaluations}
  IterT ← 0; {IterT: number of evaluation in
  the temperature T}
  while (T > 0.001) and (cont <= 30000) do
    begin
      while (IterT < SAmax) and (cont <= 30000) do
        begin
          IterT ← IterT + 1;
          cont ← cont + 1;
          Generate a neighbor s' from Nk (s);
          Δ = f (s') - f (s);
          if (Δ ≤ 0) then
            begin
              s ← s';
              if (f (s') < f (s*)) then s* ← s';
            end
          else
            begin
              be x ∈ [0, 1];
              if (x < e-Δ / T) then s ← s';
            end;
          end; // while IterT and cont.
          T ← α × T;
          IterT ← 0;
        end; // while T and cont
        return(s*);
      end. // SA
    
```

This SA algorithm solves the 3D CLPP by starting from a random initial solution with $N-1$ elements, where N is the number of ports. Each element in a position i has a value k where range is from 1 to 12 and it represents that Rule k , following the description given in Table 1, will be applied at the Port i . For each solution the objective function given by equation (1) will be computed using the translation scheme presented in Figure 11 and corresponds to the temperature T .

This SA algorithm follows the same steps of the traditional SA algorithm: given an initial temperature T , the algorithm randomly selects a neighbour from the neighbourhood

of the current solution. In the 3D CLPP, the neighbourhood structure is the random selection of the ports. Once the port is selected, then a new Rule k is also randomly selected and should be different from the previous one. The next step is to compute the new neighbour objective function value. If this value is lower than the previous solution it will be the new current solution. Otherwise, if the objective value is higher, the new neighbour solution may be accepted according to the probability of acceptance given by ($x < e^{-\Delta/T}$) that decreases as the temperatures decrease.

The SA control parameters are the cooling factor α , the number of iterations in each temperature (SA_{max}), the initial temperature T_0 and the total number of evaluations for one solution (*cont*). Here, the initial temperature is fixed at $T_0 = 100,000$ and *cont* = 30,000. For the other parameters, a variation scheme of the values: $\alpha = 0.98$ and $SA_{max} = 1,000$ (for $T > 1,000$); $\alpha = 0.95$ and $SA_{max} = 500$ (for $100 < T \leq 1000$); $\alpha = 0.92$ and $SA_{max} = 200$ ($T \leq 100$) is used. In this way, there is more slower cooling and a larger number of iterations for higher temperatures. For lower temperatures there is faster cooling and a lower number of neighbours.

A reheating scheme to increase the temperature value T (only one time for each algorithm execution) was also used for a new value that is 20% of the initial temperature value, T_0 , when T became lower than 0.01.

7 Results

The bi-objective function formulation (1) to (8) helps to find a solution according to user preferences. For example, if the user wants to obtain the best solution in terms of number of movements, then the weight alpha should be set to value 1 and the instability measure for each port is ignored. Now, if the user wants to know how much the increasing cost is in the objective function when the number of movements is ignored, this can be found by setting the beta value to 1 and the correspondent solution is obtained. Thus, the users can set their preferences in terms of alpha and beta values and also can measure the resulting trade-offs between extreme solutions.

To verify the GA, 15 instances were automatically and randomly generated to test this approach. These instances were classified according to the number of ports, the type of transportation matrix T , and containership dimensions. For each instance, a transport matrix T was generated which ensures that the containership capacity will not be exceeded for any Port p . In other words, the value given by equation (9) must be lower or equal to R for every Port p . It means, in mathematical terms, that the inequality (12) holds for every port.

$$\sum_{i=1}^p \sum_{j=p+1}^N T_{ij} \leq R \times C \quad (12)$$

for all $p = 1, \dots, N$

According to Avriel et al. (1998), it is possible to generate three types of transportation matrix:

- 1 mixed
- 2 long
- 3 short.

This classification for the transportation matrix expresses how long a container must be carried by the containership from port to port. The short transportation matrix indicates that the majority of the containers will remain for a small number of ports until unloading. The long transportation matrix indicates that most containers will remain onboard the containership from port to port before being unloaded at one of the last ports. The mixed is created in a manner that mixes short and long instance characteristics. The ship dimensions adopted for the instances presented in this article are $D = 5$, $R = 6$, $C = 50$. All instances are available at the following site:

- <https://sites.google.com/site/3dcontainershipproject/home>

In Table 3 and Table 4, the column index I corresponds to instance number; N corresponds to how many ports the Containership has to pass through; the column index M refers to the type of transportation matrix; N_{Min} refers to the lower bound on the number of movements [the lower bound value can be obtained by multiplying the sum of T_{ij} , by two which comes from equation (10)]; $FO1$ is the total number of movements performed [equation (7)]; $FO2$ is the total measure of instability [equation (8)]; and T is the computational time spent in seconds. The results presented in Table 2 and 3 were obtained with a program created in a MATLAB 7.0, a machine with a 1.66 GHz Core Duo Intel Processor, RAM memory of 2 GB and Windows Vista Operational System with Service Pack 2.

Table 3 Results obtained for GA and ($\alpha = 1$, $\beta = 0$)

I	N	M	N_{min}	$FO1$	$FO2$	$T(s)$
1	10	1	6,994	7,072	564.45	146.63
2	10	2	4,172	4,214	542.88	139.57
3	10	3	17,060	17,116	566.89	149.63
4	15	1	9,974	10,584	237.62	234.00
5	15	2	4,824	5,030	514.54	210.34
6	15	3	24,908	25,046	578.56	221.73
7	20	1	10,262	10,802	229.31	300.11
8	20	2	4,982	5,500	439.95	277.82
9	20	3	32,602	32,638	792.20	279.01
10	25	1	11,014	11,848	252.34	353.65
11	25	2	5,002	5,466	573.02	322.62
12	25	3	43,722	44,082	659.70	387.19
13	30	1	11,082	12,580	188.38	476.60
14	30	2	4,720	5,312	156.04	381.72
15	30	3	53,592	54,398	427.08	461.90

Table 4 Results obtained for GA and ($\alpha = 0, \beta = 1$)

<i>I</i>	<i>N</i>	<i>M</i>	<i>Nmin</i>	<i>FO1</i>	<i>FO2</i>	<i>T(s)</i>
1	10	1	6,994	10,106	39.12	152.09
2	10	2	4,172	6,374	13.88	145.87
3	10	3	17,060	18,018	63.31	153.14
4	15	1	9,974	12,766	30.01	241.33
5	15	2	4,824	9,078	36.12	232.31
6	15	3	24,908	25,844	113.35	258.25
7	20	1	10,262	14,658	77.21	360.50
8	20	2	4,982	9,784	17.13	317.05
9	20	3	32,602	33,700	488.88	330.84
10	25	1	11,014	17,302	19.02	558.95
11	25	2	5,002	10,564	98.93	440.53
12	25	3	43,722	44,964	161.29	455.14
13	30	1	11,082	16,556	32.45	513.86
14	30	2	4,720	7,098	10.43	402.42
15	30	3	53,592	56,290	166.05	472.50

The results presented in Table 3 and Table 4 show that the GA combined with the representation by rules approach was able to provide solutions for instances with 30 ports and a containership with five bays, six rows, and 50 columns in less than nine minutes. According to equations (1) to (8), each solution for this instance must have 40,545,000 binary variables ($30^3 \times 5 \times 6 \times 50 + 30 \times 5 \times 6 \times 50$).

From Table 3 to Table 8 it can be said that each five increments in the number of ports will result in a corresponding increment of about two minutes in the computational time spent for the GA, about eight minutes for the SA meta-heuristic, and almost double the time for the BS.

The results also show that when the objective function is to minimise the number of movements ($\alpha = 1, \beta = 0$), for the Short transportation matrix (type 3), representation by rules and the GA produce results near the optimal solution in terms of the number of movements, indicating the adequacy of the proposed rules. They produce results near the lower bound for instances with 10, 15 and 20 ports whose solutions are 0.33%, 0.55%, and 0.11%, above the lower bound. The SA algorithm produces better results than the GA and for two instances (*I* = 1 and 14), it produces the best results of all methods. But, this implies in an additional computational effort when compared with the time used by the GA. The BS produces the best results in almost all instances and for the Short transportation matrix, the results are near the optimal solution in terms of the number of movements, indicating the adequacy of the proposed rules. The BS produces results near the lower bound in the instances with 10, 15 and 20 ports whose solutions are 0.08%, 0.34%, and 0.02% above the lower bound. Since the best solutions produced by the BS result in a higher increase in the total computational time spent, due to time limits, GA or SA may be chosen.

For instances with the mixed (type 1) and long (type 2) transportation matrix, the GA, SA, and BS presented results which were distant from the lower bound. One possible explanation is the need to create more adequate rules for these kinds of instances or an improvement in the computation on the lower bound. This can be done by creating other rules based on the experience of skilled crane operators.

When the objective function is set to minimise the instability measure ($\alpha = 0, \beta = 1$), the solutions given by all methods have a significant increment in the number of movements. For example, for the instance $I = 14$ and GA method the instability measure is reduced about 11 times, but the number of movements (FOI) increased in 1,778 units.

This shows tradeoffs must be made to ensure stability. Figure 22, Figure 23, and Figure 24 show such tradeoffs for the container arrangements for the best solutions found for instance $I = 1$ when the weights are set as ($\alpha = 1, \beta = 0$) and ($\alpha = 0, \beta = 1$), respectively.

Figure 22 The container arrangement in Port 2 for the best solution found with *GA*, *SA*, and *BS* using ($\alpha = 1, \beta = 0$) (see online version for colours)

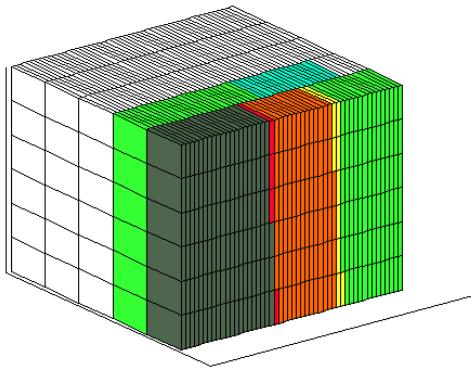


Figure 23 The container arrangement in Port 2 for the best solution found with *GA*, and *SA* using ($\alpha = 0, \beta = 1$) (see online version for colours)

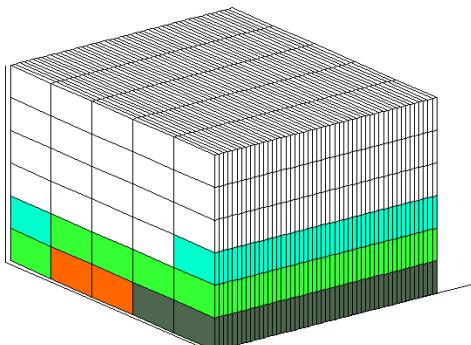


Figure 24 The container arrangement in Port 2 for the best solution found with BS using ($\alpha = 0, \beta = 1$) (see online version for colours)

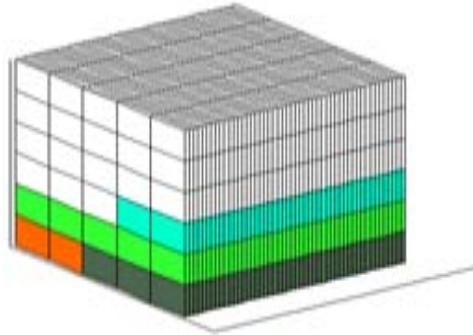


Figure 22 shows that the container arrangements for the first Port, when the objective function is to minimise the number of movements, is very unstable. On the other hand, solutions with very stable container arrangements, as shown in Figure 23 and Figure 24, can be obtained if the objective function observes only the minimisation of the instability measure. For the instance $I = 1$ this results in an increase in the number of movements ($FO1$) by 4,054 units.

Table 3 to Table 8 show that there are some instances where the minimisation of the number of movements also produces a solution that minimises the instability measure. For example, in Table 2 the best solution found for instance $I = 2$ is given by BS and it also minimises the number of movements ($FO1 = 4202$) and the instability measure ($FO2 = 478.22$). However, the same behaviour does not happen with $I = 5$ where the best solution in terms of number of movements is given by BS ($FO1 = 4,936$), but the best solution for instability is given by GA ($FO2 = 514.54$).

Table 5 Results obtained for SA and ($\alpha = 1, \beta = 0$)

I	N	M	N_{min}	$FO1$	$FO2$	$T(s)$
1	10	1	6,994	7,068	507.14	1,654.91
2	10	2	4,172	4,208	492.45	1,531.98
3	10	3	17,060	17,088	583.18	1,431.31
4	15	1	9,974	10,420	221.40	2,054.83
5	15	2	4,824	5,082	532.23	2,044.66
6	15	3	24,908	24,998	595.22	1,887.36
7	20	1	10,262	10,740	202.09	2,485.62
8	20	2	4,982	5,458	451.71	2,482.72
9	20	3	32,602	32,632	1,000.11	2,554.71
10	25	1	11,014	11,590	200.89	2,866.11
11	25	2	5,002	5,430	561.69	2,495.45
12	25	3	43,722	44,078	543.47	2,624.41
13	30	1	11,082	12,146	183.97	3,158.39
14	30	2	4,720	5,246	168.66	2,687.99
15	30	3	53,592	54,454	519.62	3,076.42

Table 6 Results obtained for *SA* and ($\alpha=0, \beta=1$)

<i>I</i>	<i>N</i>	<i>M</i>	<i>Nmin</i>	<i>FO1</i>	<i>FO2</i>	<i>T(s)</i>
1	10	1	6,994	11,122	14.20	1,385.63
2	10	2	4,172	6,672	11.55	1,318.55
3	10	3	17,060	17,554	92.85	1,094.48
4	15	1	9,974	13,910	30.10	1,850.03
5	15	2	4,824	8,098	34.49	1,761.20
6	15	3	24,908	25,896	154.01	1,607.45
7	20	1	10,262	16,396	13.60	2,286.52
8	20	2	4,982	10,022	14.57	2,258.52
9	20	3	32,602	33,378	575.36	2,299.17
10	25	1	11,014	14,674	12.69	2,767.76
11	25	2	5,002	9,456	106.50	2,499.26
12	25	3	43,722	45,120	218.75	2,727.52
13	30	1	11,082	18,180	17.99	3,158.18
14	30	2	4,720	8,604	13.99	2,708.06
15	30	3	53,592	55,754	299.82	3,082.48

Table 7 Results obtained for *BS* and ($\alpha=1, \beta=0$)

<i>I</i>	<i>N</i>	<i>M</i>	<i>Nmin</i>	<i>FO1</i>	<i>FO2</i>	<i>T(s)</i>
1	10	1	6,994	7,072	588.45	429.46
2	10	2	4,172	4,202	478.22	403.39
3	10	3	17,060	17,074	577.54	453.83
4	15	1	9,974	10,234	194.24	1,752.34
5	15	2	4,824	4,936	603.19	1,550.85
6	15	3	24,908	24,992	420.96	1,808.89
7	20	1	10,262	10,432	163.64	4,275.94
8	20	2	4,982	5,152	447.15	3,946.33
9	20	3	32,602	32,610	1,146.09	4,102.06
10	25	1	11,014	11,154	285.08	8,217.97
11	25	2	5,002	5,156	553.64	7,576.42
12	25	3	43,722	43,942	613.83	9,506.44
13	30	1	11,082	11,430	198.93	15,756.15
14	30	2	4,720	5,598	126.63	14,468.23
15	30	3	53,592	53,896	701.56	17,335.38

Table 8 Results obtained for *BS* and ($\alpha = 0, \beta = 1$)

<i>I</i>	<i>N</i>	<i>M</i>	<i>Nmin</i>	<i>FO1</i>	<i>FO2</i>	<i>T(s)</i>
1	10	1	6,994	10,432	17.59	507.34
2	10	2	4,172	7,172	12.52	522.42
3	10	3	17,060	17,642	15.83	618.38
4	15	1	9,974	15,058	12.59	2,872.77
5	15	2	4,824	9,560	9.93	2,274.02
6	15	3	24,908	25,952	17.38	2,022.22
7	20	1	10,262	16,374	6.68	5,723.36
8	20	2	4,982	8,652	10.21	4,863.09
9	20	3	32,602	33,544	431.16	4,651.98
10	25	1	11,014	14,058	4.59	9,684.85
11	25	2	5,002	10,240	92.11	9,455.98
12	25	3	43,722	45,304	32.45	10,397.03
13	30	1	11,082	18,972	6.04	18,073.31
14	30	2	4,720	7,544	6.59	15,066.07
15	30	3	53,592	55,062	29.81	18,301.89

8 Conclusions and future work

This Article shows a new approach for the 3D Containership loading planning problem (3D CLPP) that has three advantages:

- it allows compact encoding by representing the solution as a vector with P elements instead of $(R \times C) \times (P + P^3)$ binary variables, which enables large-scale problem solving
- the experience of skilled personnel can be incorporated into the optimisation process under the form of a rule and a computational simulation
- the solutions with this encoding are always feasible, which avoids the problem of extended processing time to make solutions feasible.

This new encoding approach saves considerable computational time and produces quality solutions. In some instances the optimality is very close to the lower bound value, showing that this approach may eventually have practical commercial value. Different or more complex rules would be needed to ensure stability and to address specific unloading/loading constraints at a particular port.

Actually, this should not be a problem since the proposed approach can deal with this practical situation by only changing the set of created and considered rules. Moreover, this strategy enables the use of the proposed approach for other similar storage problems, e.g., AS/RS.

One promising idea for future work is to codify the previous approaches proposed for the containership problem in the form of rules or interview crane operators to produce rule from crew experience. Future work must address this 3D CLPP with methods that deal with the pareto-optimal frontier in order to choose the solutions.

Acknowledgements

This research was supported by the Foundation for the Support of Research of the State of São Paulo (FAPESP) under the process 2010/51274-5 and by the Brazilian Council for the Development of Science and Technology (CNPq). Also, the authors would like to thank the anonymous reviewer for their valuable comments, which helped to clarify and improve the contents of this paper.

References

- Ambrosino, D., Anghinolfi, D., Paolucci, M. and Sciomachen, A. (2010) 'An experimental comparison of different heuristics for the master bay plan problem', *Lecture Notes in Computer Science*, Vol. 6049, pp.314–325.
- Ambrosino, D., Sciomachen, A. and Tanfani, E. (2006) 'A decomposition heuristics for the containership stowage problem', *J. Heuristics*, Vol. 12, No. 3, pp.211–233.
- Avriel, M. and Penn, M. (1993) 'Containership stowage problem', *Computers and Industrial Engineering*, Vol. 25, Nos. 1–4, pp.271–274.
- Avriel, M., Penn, M. and Shpirer, N. (2000) 'Containership stowage problem: complexity and connection to the coloring of circle graphs', *Discrete Applied Mathematics*, Vol. 103, Nos. 1–3, pp.271–279.
- Avriel, M., Penn, M., Shpirer, N. and Wittenboon, S. (1998) 'Stowage planning for containerships to reduce the number of shifts', *Annals of Operations Research*, Vol. 76, No. 1, pp.55–71.
- Blum, C. and Roli, A. (2003) 'Metaheuristics in combinatorial optimization overview and conceptual comparison', *ACM Computing Surveys*, Vol. 35, No. 3, pp.268–308.
- Della Croce, F. and T'kindt, V. (2002) 'A recovering beam search algorithm for the one-machine dynamic total completion time scheduling problem', *Journal of the Operational Research Society*, Vol. 53, No. 11, pp.1275–1280.
- Dubrovsky, O., Levitin, G. and Penn, M. (2002) 'A genetic algorithm with a compact solution encoding for the containership stowage problem', *Journal of Heuristics*, Vol. 8, No. 6, pp.585–599.
- Dyckhoff, H. (1990) 'A typology of cutting and packing problems', *European Journal of Operational Research*, Vol. 44, No. 2, pp.145–159.
- Fan, L., Low, M.Y.H., Ying, H.S., Jing, H.W., Min, Z. and Aye, W.C. (2010) 'Stowage planning of large containership with tradeoff between crane workload balance and ship stability', *Proceedings of the International MultiConference of Engineers and Computers Scientists*, Vol. III, pp.1–7.
- Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI, USA.
- Imai, A., Sasaki, K., Nishimura, E. and Papadimitriou, S. (2006) 'Multi-objective simultaneous stowage and loading planning for a container ship with container rehandle in yard stacks', *European Journal of Operational Research*, Vol. 171, No. 3, pp.373–389.
- Kirkpatrick, S., Gellat, D.C. and Vecchi, M.P. (1983) 'Optimizations by simulated annealing', *Science*, Vol. 220, No. 4598, pp.671–680.

- Lowerre, B.T. (1976) 'The HARPY speech recognition system', PhD thesis, Carnegie-Mellon University, USA.
- McMullen, P.R. and Frazier, G.V. (2000) 'A simulated annealing approach to mixed-model sequencing with multiple objectives on a JIT line', *IIE Transactions*, Vol. 32, No. 8, pp.679–686.
- Michalewicz, Z. (1996) *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed., Springer-Verlag, London, UK.
- Ow, P.S. and Morton, T.E. (1988) 'Filtered beam search in scheduling', *International Journal of Production Research*, Vol. 26, No. 1, pp.35–62.
- Ribeiro, C.M., Azevedo, A.T. and Teixeira, R.F. (2010) 'Problem of assignment cells to switches in a cellular mobile network via beam search method', *WSEAS Transactions on Communications*, Vol. 9, No. 1, pp.11–21.
- Sciomachen, A. and Tanfani, E. (2007) 'A 3D-BPP approach for optimizing stowage plans and terminal productivity', *European Journal of Operational Research*, Vol. 183, No. 3, pp.1433–1446.
- Vacca, I., Bierlaire, M. and Salani, M. (2007) 'Optimization at container terminals: status, trends and perspectives', *7th Swiss Transportation Research Conference*, September, pp.1–21.
- Valente, J.M.S. and Alves, R.A.F.S. (2005) 'Filtered and recovering beam search algorithm for the early/tardy scheduling problem with no idle time', *Computers & Industrial Engineering*, Vol. 48, No. 2, pp.363–375.
- Wäscher, G., Haußner, H. and Schumann, H. (2007) 'An improved typology of cutting and packing problems', *European Journal of Operational Research*, Vol. 183, No. 3, pp.1109–1130.
- Wilson, I. and Roach, P.A. (1999) 'Principles of combinatorial optimization applied to container-ship stowage planning', *Journal of Heuristics*, Vol. 5, No. 4, pp.403–418.
- Wilson, I. and Roach, P.A. (2000) 'Container stowage planning: a methodology for generating computerised solutions', *Journal of the Operational Research Society*, Vol. 51, No. 11, pp.1248–1255.