WILEY

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 23 (2016) 147–172
DOI: 10.1111/itor.12099

The pallet loading problem: a review of solution methods and computational experiments

Elsa Silva[a], José F. Oliveira[b] and Gerhard Wäscher[c]

[a]*INESC TEC, Campus da FEUP, Rua Dr Roberto Frias, 4200-465 Porto, Portugal*
[b]*INESC TEC, Faculdade de Engenharia, Universidade do Porto, Portugal*
[c]*Faculty of Economics and Management, Otto-von-Guericke-University Magdeburg, Germany*
E-mail: emsilva@inescporto.pt [Silva]

Received 9 May 2013; received in revised form 22 April 2014; accepted 23 April 2014

Abstract

The manufacturer's pallet loading problem (MPLP) has been widely studied during the past 50 years. It consists of placing a maximum number of identical rectangular boxes onto a single rectangular pallet. In this paper, we have reviewed the methods that have been proposed for the solution of this problem. Furthermore, the various problem instances and data sets are analyzed that have been used in computational experiments for the evaluation of these methods. The most challenging and yet unsolved methods are identified. By doing so, areas of future research concerning the MPLP can be highlighted.

Keywords: loading problems; packing problems; cutting problems; combinatorial optimization

1. Introduction

The loading problem, as considered in this paper, consists of placing a maximum number of identical rectangular boxes onto a single rectangular pallet. This problem is also occasionally, and more precisely, called the *manufacturer's pallet loading problem* (MPLP; Birgin et al., 2010, 2012; Hodgson, 1982; Morabito and Farago, 2002; Morabito and Morales, 1998; Pureza and Morabito, 2006; Wu and Ting, 2007; Yi et al., 2009). It has been studied since the 1960s (Barnett and Kynch, 1967); in recent years, however, research on pallet loading has shifted from the MPLP to more complex problems (Wäscher et al., 2007), for example, to the distributer's pallet loading problem (Birgin et al., 2012; Bischoff and Ratcliff, 1995; Morabito and Arenales, 1994). This shift of the research focus was probably motivated by the successful development of exact algorithms (e.g. Martins and Dell, 2008; Wu and Ting, 2007) that appear to solve every instance of the MPLP to optimality in a reasonable computing time. In extensive numerical experiments that included 196,557 problem instances from the literature, Birgin et al. (2010) were able to obtain optimal solutions

in 99.5% of cases, whereas the average computing time per instance amounted to four seconds only.

Considering the evidence from these experiments, the purpose of this paper is to assess if there is still room for research concerning the MPLP and, if so, identify areas of future research. In order to examine these issues, we will first take a theoretical perspective. The problem will be defined and represented in a formal model in Section 2. In Section 3 its computational complexity will be discussed. It will be shown that already the current knowledge concerning this aspect is not really satisfying. Then, the state-of-the-art concerning (upper) bounds and exact and heuristic solution approaches to the MPLP will be presented in Sections 4 and 5, respectively. In Section 6 we will analyze what kind of individual problem instances and sets of problem instances (data sets) have been used in numerical experiments and how the described methods managed to deal with them. This will allow us to identify problem categories that have not been solved satisfactorily so far. Research in future could concentrate on the development of methods for these categories, and instances from these categories should be used in numerical experiments. Finally, some general conclusions will be presented and discussed in Section 7.

2. Problem definition and representation

As for the MPLP, when placing a maximum number of boxes onto the pallet, certain basic geometric feasibility conditions must hold. The boxes have to be arranged orthogonally, that is, their edges have to be positioned in parallel to the edges of the pallet, and, as with other cutting/packing problems, they must not overlap. They may be rotated by 90°. Their vertical orientation, however, is fixed. Since the boxes have to be loaded in layers, the original three-dimensional (3D) MPLP is reduced to a two-dimensional (2D) one in which a maximum number of identical small rectangles have to be assigned orthogonally to a single given large rectangle such that all small rectangles lie entirely within the large rectangle and the small rectangles do not overlap. Any formal description of a solution to the MPLP will be called a loading pattern.

In the typology of cutting and packing (C&P) problems proposed by Wäscher et al. (2007), this problem is categorized as a 2D, rectangular identical item packing problem (IIPP) and considered as a pure layout problem regarding the arrangement of identical small items. Unlike for other output minimization problem types in C&P, no decision has to be made with respect to which items are to be loaded.

Based on a model proposed by Beasley (1985) for a more general 2D C&P problem in which the small items can be of different types and the associated values may be nonproportional to the size (area) of the items, Dowsland (1987b) developed a model for the MPLP that has widely served as a basis for the development of bounds and solution methods. In order to define the problem under consideration more precisely, this model will be presented in the following. Other models, for example, the binary model with disjunctive constraints of Tsai et al. (1993) for the 3D version of the problem, could also be adapted for the MPLP. However, it is hardly ever referred to in the literature related to the MPLP; therefore, we do not present any further details here.

Let L and W represent the length and width of the pallet, and a and b the length and width of the box. Without loss of generality, it can be assumed that L, W, a, and b can be represented by integer numbers and $L \geq W \geq a \geq b$ holds. Since rotations are permitted, each box can be placed in two

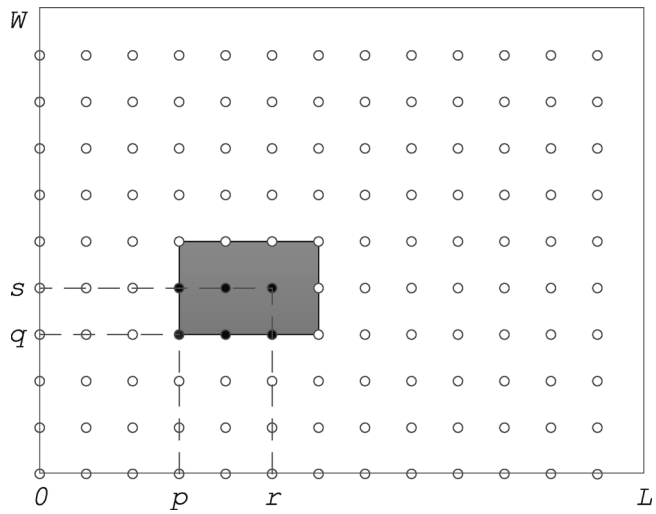

Fig. 1. Infeasible assignment positions (*r*,*s*).

modes onto the pallet, namely horizontally ($i = 1$) and vertically ($i = 2$). A horizontally placed box will be represented by $(a_1, b_1) = (a, b)$, a vertically placed box by $(a_2, b_2) = (b, a)$. In other words, (a_i, b_i), $i = 1, 2$ indicates the length and width of the small item considering orientation i.

Each decision variable of the model is related to the question of whether or not the lower left corner of a small item with orientation i, $i = 1, 2$, is assigned to a particular point (position) (p, q) given by the corresponding coordinates on the pallet. All integer combinations of coordinate values are potential assignment positions, that is, the set P of all potential assignment positions is given by $P = \{(p, q) | p \in X, q \in Y\}$, where $X = \{0, 1, 2, \ldots, L - 1\}$ and $Y = \{0, 1, 2, \ldots, W - 1\}$. The decision variables are then defined as follows:

$$x_{ipq} = \begin{cases} 1, & \text{if a small item with orientation } i \ (i = 1, 2) \text{ is assigned to position } (p, q), \\ & \text{where } p \in X \wedge p \leq L - a_i \text{ and } q \in Y \wedge q \leq L - b_i, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Furthermore, when assigning boxes to assignment positions, the overlapping of boxes must be prohibited. Figure 1 demonstrates that boxes cannot be assigned to positions (r, s) indicated by the black dots if a box is assigned horizontally ($i = 1$) to position (p, q). Otherwise, overlapping will occur.

In order to allow for the formulation of constraints that prohibit the overlapping of boxes in the loading pattern to be generated, parameters u_{ipqrs} are introduced as follows:

$$u_{ipqrs} = \begin{cases} 1, & \text{if} \quad 0 \leq p \leq r \leq p + a_i - 1 \leq L - 1 \quad \text{and} \quad 0 \leq q \leq s \leq q + b_i - 1 \leq W - 1, \\ & p, r \in X \wedge p, r \leq L - a_i, \ q, s \in Y \wedge q, s \leq L - b_i, \ \text{and } i = 1, 2, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The parameter u_{ipqrs} is equal to 1 if a small item with orientation i ($i = 1, 2$) assigned to position (p, q), where $p \in X, p \leq L - a_i; q \in Y, q \leq W - b_i$, overlaps with a small item in position (r, s), such that $r \in X, 0 \leq p \leq r \leq p + a_i - 1 \leq L - 1, s \in Y, 0 \leq q \leq s \leq q + b_i - 1 \leq W - 1$.

Then the following binary optimization problem can be formulated as a representation of the MPLP:

$$\text{Maximize} \sum_{i=1}^{2} \sum_{p \in X \wedge p \leq L - a_i} \sum_{q \in Y \wedge q \leq W - b_i} x_{ipq} \tag{3}$$

$$\text{subject to} \sum_{i=1}^{2} \sum_{p \in X \wedge p \leq L - a_i} \sum_{q \in Y \wedge q \leq W - b_i} u_{ipqrs} x_{ipq} \leq 1, \qquad \forall r \in X, s \in Y; \tag{4}$$

$$x_{ipq} \in \{0, 1\}, \qquad \forall i = 1, 2, \quad p \in X \wedge p \leq L - a_i, \quad q \in Y \wedge q \leq W - b_i. \tag{5}$$

In the objective function (3), the small items are counted that are assigned to the pallet. Their number has to be maximized. Constraints (4) ensure that no overlapping of small items occurs. In the final solution, the values of the decision variables (5) indicate where small items are located on the pallet and what their orientation is.

We note that, according to the definition of the set P, the number of decision variables in the above models (1)–(5) may become quite large. In order to reduce their number, instead of X and Y as defined above, one may use the normal sets proposed by Herz (1972) and Christofides and Whitlock (1977) without losing the optimal solution. In these sets, only coordinates are to be considered, which represent non-negative integer combinations of the length and width of the small item type, that is,

$$X' = \{l | l = \alpha \cdot a + \beta \cdot b, 0 \leq l \leq L - b, \alpha, \beta \in Z_+ \cup \{0\}\}, \tag{6}$$

$$Y' = \{w | w = \alpha \cdot a + \beta \cdot b, 0 \leq w \leq W - b, \alpha, \beta \in Z_+ \cup \{0\}\}. \tag{7}$$

Dowsland (1984) showed that instances can be divided into classes in a way that all instances belonging to the same class have the same optimal loading pattern, that is, the instances are equivalent. The consideration of these *equivalence classes* allows for the reduction of the size of solution space since by generating an optimal solution for one member instance of an equivalence class, an optimal solution for each instance of the entire set is obtained.

The definition of equivalence classes is based on the concept of *feasible partitions, efficient partitions, perfect partitions and perfect partition reductions.* A feasible partition of a pallet edge (length, width) S into box dimensions a and b is an ordered pair of non-negative integers (n, m) satisfying $S \geq na + mb$. An efficient partition is a feasible partition that cannot be increased by the addition of another box, that is, an ordered pair of non-negative integers (n, m) satisfying $0 \leq S - na - mb < b$.

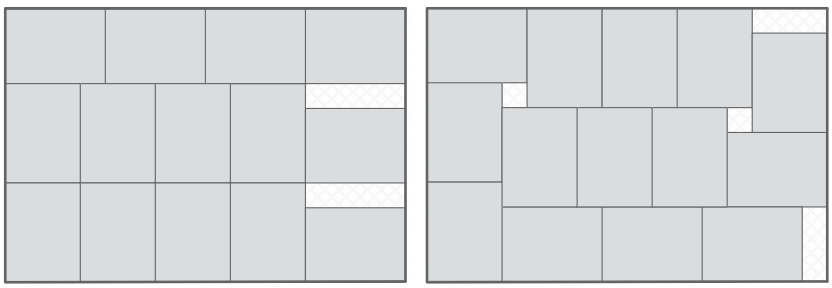

Fig. 2. Loading patterns: (a) Guillotinable and (b) nonguillotinable.

A perfect partition is a feasible partition for which the equality holds, that is, an ordered pair (n, m) satisfying $S = na + mb$. The perfect partition reduction of a pallet edge S in a and b is the largest integer less than or equal to S that has a perfect partition a and b, that is, $\max\{S' : S' \leq S, S' = na + mb, n, m \text{ non-negative integers}\}$. Given an instance with dimensions (L, W, a, b), its perfect partition is (L', W', a, b) with $L' \leq L$ and $W' \leq W$.

3. Computational complexity

Despite its simple structure, based on the work of Fowler et al. (1981), the MPLP has early been classified as an NP-complete problem by Dowsland (1985b). Authors of subsequent publications, for example, Dowsland (1987a, 1987b), Herbert and Dowsland (1996), Bhattacharya and Roy (1998), Pureza and Morabito (2006), have adopted this classification.

When considering the decision version of the pallet loading problem, that is, answering the question if n items can be packed, most of the existing algorithms perform a search over a tree of possible loading patterns, which can have 2^n nodes (Nelissen, 1995a), or in other words, they have an exponential worst-case time complexity.

On the other hand, however, doubts concerning the NP-completeness of the MPLP still exist. Such doubts, among others, are supported by the fact that even problem instances in which pallets can be loaded with a large number of small items can be solved optimally in reasonable computing times.

Apart from this empirical observation, also some more theoretical arguments can be brought forward. For example, Nelissen (1995a) and Alvarez-Valdes et al. (2005a) have pointed out that the encoding of any instance of the MPLP only requires four integers, namely the lengths and width of the pallet and the length and the width of the box, which is not common for NP-complete problems.

Furthermore, Tarnowski et al. (1994) have shown that the MPLP can be solved in polynomial time if the guillotine constraint is imposed, that is, only guillotinable loading patterns have to be considered. A loading pattern is called guillotinable if it can be obtained by dividing the pallet successively into two subrectangles (see Fig. 2). The time complexity of the algorithm presented in their paper is $O(\log_2 L \cdot \log_2^2 a)$.

Nelissen (1993) even doubted that the MPLP is included in the problem class NP. If the problem is transformed into the problem of deciding whether it is possible to load n boxes on the pallet, assuming that it is only possible to process a fixed percentage of the n boxes in constant time, then the time needed by a Turing machine for checking the input string is not polynomially bounded. Thus, Nelissen concluded that the complexity of the MPLP is unknown. In fact, this is still the current state of knowledge (Arslanov, 2000; Birgin et al., 2005, 2010; Letchford and Amaral, 2001; Lins et al., 2003; Martins and Dell, 2008; Morabito and Morales, 1998; Young-Gun and Maing-Kyu, 2001).

4. Upper bounds

Bounds are of great significance for the improvement of exact methods and for the evaluation of the performance of heuristics. A trivial upper bound for the number of items in an optimal solution for the MPLP is the *area bound* ($\left\lfloor \frac{L \cdot W}{a \cdot b} \right\rfloor$), which results from rounding down to the nearest integer the ratio of the pallet area to the item area. However, Smith and De Cani (1980) and Dowsland (1985a) state that this bound is only accurate in 15% of the cases they have tested.

Due to the poor quality of the area bound, studies on upper bounds have dominated the literature over the years. For the special case in which the item has unit width and integer length, a lower bound on the waste (the unused area of the pallet) was presented in Barnett and Kynch (1967). An extension of this bound is proposed in Barnes (1979), where also an item with unit length and integer width is considered. This bound is based on packing strips onto the pallet. Dowsland (1985a) improved Barnes' bound by considering the perfect partitions of the pallet edges instead of referring to the original pallet dimensions.

Dowsland (1984) showed that instances having equal efficient partitions also have the same optimal loading patterns. Thus, such instances are equivalent. Consequently, the definition of an efficient partition is used to divide a set of instances into equivalence classes, allowing the reduction of the size of the solution space. Solving optimally one member instance of an equivalence class, results in solving the entire set of instances of the class to an optimum, the same being valid for the value of the upper bound of the class. A modification of the upper bound proposed in Dowsland (1984) was presented in Daniels and Ghandforoush (1990). However, some errors and a counterexample are pointed out in George et al. (1992), who propose a correction for some of the cases.

Nelissen (1995a) proposes an upper bound based on a linear programming model, which is solved for different cost functions. The upper bound is proven by contradiction, that is, it is assumed that a given upper bound is valid and it is tried to reach a contradiction by gradually tightening the structural constraints by different arguments. Computational experiments were performed for two data sets, which only differ with respect to the maximal number of items that can be loaded on the pallet. For the data set with a maximal number of 50 items, the upper bound provided an accuracy of 99.73% for 5210 equivalence classes. For the data set with a maximal number of 100 items, the upper bound was equal to the optimal solution in 98.41% of the 11,832 equivalence classes, requiring an average computing time of 1.7 seconds per instance. The concepts of feasible and efficient partitions are used also by Exeler (1991), who presents a problem formulation with linear constraints and a nonconvex objective function that is transformed into a piecewise linear problem.

The linear relaxation of models (1)–(5) was used for the determination of an upper bound for the MPLP in the computational experiments of Morabito and Morales (1998) and Ribeiro and Lorena (2007). Morabito and Farago (2002) develop an upper bound based on a Lagrangean relaxation of this model.

A review of upper bounds, including a theoretical dominance analysis, was presented in Letchford and Amaral (2001). The authors compared four upper bounds: the area bound, Barnes' bound, upper bound obtained by the linear relaxation of Beasley's integer programming model, and the upper bound proposed by Isermann (1987). The authors concluded that the upper bound resulting from the linear relaxation of Beasley's integer programming model dominates the other upper bounds considered. In computational experiments including representatives of 3176 equivalence classes, the upper bound resulting from the linear relaxation of Beasley's integer programming model obtained a value equal to the optimal objective function value in 99.78% of the tested cases, confirming the quality of the bound. Although it was not proven formally, they also concluded that this bound is incomparable to the upper bound from Nelissen (1993), since neither dominates the other.

5. Solution methods

5.1. Exact methods

The main disadvantage of models (1)–(5) is that for large problems the number of variables can be very high (Alvarez-Valdes et al., 2005b; Nelissen, 1995a). In order to overcome this difficulty, different strategies have been proposed in the literature.

In Alvarez-Valdes et al. (2005a) a branch-and-cut algorithm is presented for solving the MPLP to optimality. Beasley's model is improved by adding new constraints and by using variable reduction procedures. Wu and Ting (2007) propose a two-phase algorithm, in which an integer programming model is solved first in order to obtain the maximum number of items that can be horizontally and vertically packed onto the pallet. Then this value is used for the generation of a loading pattern by means of a constraint programming model that is based on Beasley's model. In Ribeiro and Lorena (2007, 2008), a Lagrangean relaxation for Beasley's model is introduced and solved by column generation with clusters.

An exact algorithm based on a graph-theoretic model of the MPLP is presented in Dowsland (1987b). The problem is solved by determining the maximum stable set in a finite graph, in which the nodes represent the possible positions of the box on the pallet and two nodes are connected by an arc if the respective boxes overlap. The model was based on the one-to-one correspondence between the maximum stable sets in the graph and optimal layout. By combining the model with variable reduction and bounding conditions, optimal solutions were obtained for 6000 instances with a maximum of 50 boxes on the pallet.

Exact tree search procedures are proposed by Bhattacharya and Roy (1998) and Martins and Dell (2008). In tree search procedures at each node a partial layout of the boxes on the pallet has been built. The methods differs with respect to the way in which boxes are added to the partial solution and the different bounding procedures.

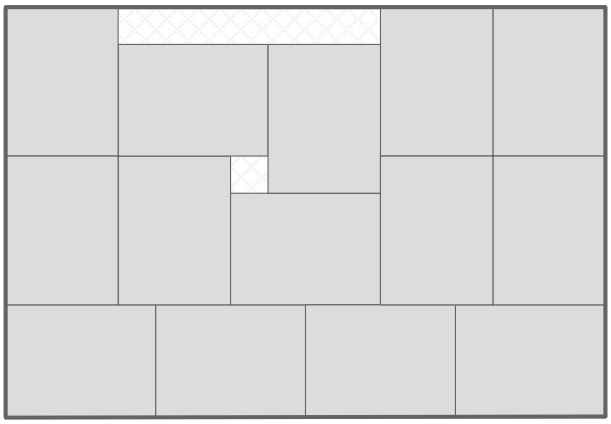

Fig. 3. G4 loading pattern.

In Arenales and Morabito (1995), an AND/OR-graph method is proposed to solve the problem of packing items of different sizes onto a single pallet. The MPLP can be considered as a special case of this problem, in which only two items have to be taken into account, the original item and the item rotated by 90°. The loading patterns result from combining successive guillotine and/or nonguillotine cuts. When performing a nonguillotine cut on a rectangle, five new rectangles are obtained. These new rectangles are considered blocks because they can be formed by a set of items.

5.2. Heuristics

Heuristics and metaheuristics have been also used to solve the MPLP. They can serve as stand-alone solution procedures or they can be integrated in exact methods in order to provide good initial solutions or good lower bounds.

Constructive heuristic for the MPLP are based on the concept of blocks. These methods, called block heuristics, consider regions on the pallet and fill them with well-defined arrangement of items. The first heuristic of this type, a 4-block heuristic, was proposed by Steudel (1979). Later, similar algorithms have been proposed by Steudel (1984) and Smith and De Cani (1980). These methods have in common that the pallet is divided into, at most, four rectangular blocks that are composed of items having the same orientation. They differ with respect to how the size of the blocks is actually been determined. All these and other block heuristics can be applied recursively, meaning that once a block is defined, another block can be obtained from the first block and so on, until all the boxes are placed. A so-called "G4 heuristic" was proposed by Scheithauer and Terno (1996). It consists of recursively dividing the pallet into "standard" blocks, that is, blocks consisting of items that have the same orientation, or 4-block structures that consist of a single item or four standard blocks. An example of a G4 loading pattern is provided in Fig. 3.

Bischoff and Dowsland (1982) were the first to introduce a 5-block heuristic, in which they introduced a fifth block in order to fill the inner hole created by the 4-block heuristics. Morabito and Morales (1998) proposed an extension to this procedure, known as the recursive 5-block

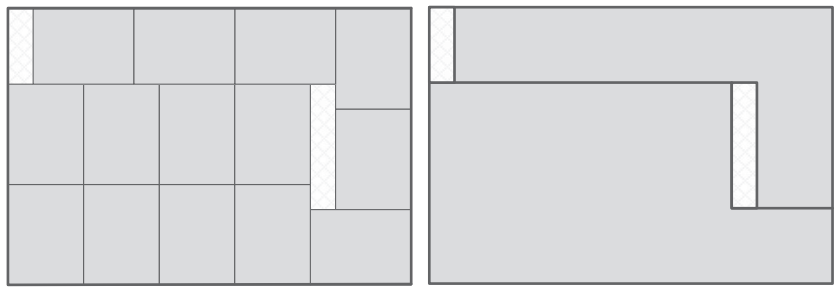

Fig. 4. Loading pattern representation: (a) L-shaped and (b) L-block.

heuristic. A recursive heuristic, in which the pallet is divided into five blocks, was also presented in Young-Gun and Maing-Kyu (2001).

The recursive 5-block heuristic was further extended to the division of the pallet into L-shaped blocks in Lins et al. (2003). A representation of a pattern consisting of such blocks, that is, an L-shaped pattern, is shown in Fig. 4, and its division into two L-shaped blocks is presented in Fig. 4. Based on computational experiments, the authors conjectured that their approach (called L-shaped-block heuristic) is an exact one, since the obtained solutions were all optimal. Long computing times turned out to be the main disadvantage of this method. Later, in Birgin et al. (2005) this disadvantage was overcome by the utilization of a different data structure. In Birgin et al. (2010) the recursive 5-block heuristic and the L-algorithm were combined giving rise to the recursive partitioning algorithm.

Three other heuristics are presented in Martins and Dell (2008): the hollow block heuristic, G-5 heuristic, and higher order nonguillotine heuristic. The hollow block heuristic considers blocks that cover the whole pallet. The dimension of the blocks are defined by selecting compatible pairs of perfect partitions for both the length and width. Similarly to the recursive 5-block heuristic proposed by Morabito and Morales (1998), the G5 heuristic divides the pallet into five blocks, however, uses the hollow block heuristic to identify partial solutions. The higher order nonguillotine heuristic is an extension of the G5 heuristic, in which five additional loops are considered and the pallet is divided into at most eight blocks.

Metaheuristics have also been suggested for solving the MPLP. In Dowsland (1993), simulated annealing is applied. The neighborhood of a given solution is defined by the set of solutions that can be obtained by moving any piece that is not overlapping with another piece as any other position.

A genetic algorithm is presented by Herbert and Dowsland (1996). The solution space includes both feasible and infeasible solutions, and different fitness functions are used to penalize infeasibility. A repair operator ensures the feasibility of the final population.

In Amaral and Wright (2001), a variant of tabu search, the strategic oscillation algorithm, is used to solve the MPLP. The method is composed of five stages: at first an initial feasible solution is generated; then, in a second stage, by the addition of boxes, the initial solution becomes infeasible; in the third stage, the solution is turned into a feasible one by randomly removing overlapping boxes; in the fourth stage boxes are removed in order to create a large gap, and in the last stage, through a greedy procedure, boxes are added to the solution, filling the gap created in the previous stage.

In Alvarez-Valdes et al. (2005b), a tabu search algorithm is proposed. The solution space contains only feasible solutions, and the moves are related to blocks and G4 structures. The evaluation of the moves is based on weighting different characteristics of the loading patterns. A tabu search algorithm is also suggested by Pureza and Morabito (2006). The solution space contains only feasible solutions and the moves are also related to blocks. Here, however, the moves refer to the expansion of blocks in specific directions.

A different approach is presented by Yi et al. (2009). The authors define so-called "pinwheel loading" patterns and propose a generation method. The pinwheel pattern concept integrates all the pattern structures discussed previously.

6. Analysis of data used in numerical experiments

The use of appropriate test problems is crucial for the evaluation of newly developed algorithms. In this section, a survey of the data sets commonly used in numerical experiments for the evaluation of algorithms for the MPLP is presented. Basically, two types of data can be used in such experiments:

- *Benchmark instances*: These are single problem instances that have proven themselves being particular hard to solve. Such instances may have arisen from practical applications, but they may have also been systematically designed with respect to aspects that are likely to make an instance difficult. Furthermore, they may have been identified as particular challenging members of a particular class of problem instances in numerical experiments.
- *Testbeds*: Testbeds consist of several classes of problem instances, each of which can be identified by a specific set of problem characteristics. Testbeds are used in the systematic evaluation of solution methods, in particular in order to determine how computing times and/or solution quality react to variations of problem parameters. The instances of the problem classes are usually generated randomly.

Obviously, there can be interactions between these two types of data. As a consequence from having identified a particular hard instance within a class of problem instances one may only consider this particular instance in future experiments. Likewise, a single proven benchmark instance may lead to the generation of a class of problem instances with identical or similar characteristics.

6.1. Benchmark instances

Table 1 presents information on instances that have been used as benchmark instances in at least two numerical experiments described in the literature. Instances that have only been used once have not been accommodated in our list because we assumed that such instances have immediately been solved in a reasonable computing time. In Table 1, each instance is distinguished with respect to an instance number (#), the length (L) and the width (W) of the pallet, the length (a) and the width (b) of the box, and the optimal objective function value (z), that is, the maximum number of boxes that can be loaded on the pallet. Furthermore, we mention all papers describing numerical experiments

Table 1
Benchmark instances

#	L	W	a	b	z	Papers where the instances were used	Year
1	1000	1000	205	159	30	Bischoff and Dowsland (1982), Young-Gun and Maing-Kyu	1982
2	1000	1000	200	150	33	(2001), Wu and Ting (2007)	
3	14	10	3	2	23	Dowsland (1984), Young-Gun and Maing-Kyu (2001), Wu and Ting (2007)	1984
4	22	16	5	3	23	Dowsland (1984), Arenales and Morabito (1995),	1984
5	86	82	15	11	42	Bhattacharya and Roy (1998), Morabito and Morales (1998), Young-Gun and Maing-Kyu (2001), Pureza and Morabito (2003), Wu and Ting (2007), Martins and Dell (2008)	
6	30	22	7	4	23	Dowsland (1984), Arenales and Morabito (1995),	1984
7	46	34	11	6	23	Bhattacharya and Roy (1998), Young-Gun and Maing-Kyu	
8	50	36	11	7	23	(2001), Wu and Ting (2007)	
9	53	51	9	7	42		
10	63	60	11	8	42		
11	76	73	13	10	42		
12	87	47	7	6	97	Nelissen (1995a), Bhattacharya and Roy (1998), Morabito and Morales (1998), Amaral and Wright (2001), Young-Gun and Maing-Kyu (2001), Pureza and Morabito (2003, 2006) Wu and Ting (2007), Martins and Dell (2008)	1995
13	57	44	12	5	41	Nelissen (1995a), Scheithauer and Terno (1996), Morabito and Morales (1998), Young-Gun and Maing-Kyu (2001), Pureza and Morabito (2003, 2006), Wu and Ting (2007), Martins and Dell (2008)	1995
14	40	33	7	4	46	Nelissen (1995a), Bhattacharya and Roy (1998), Young-Gun and Maing-Kyu (2001), Letchford and Amaral (2001), Ribeiro and Lorena (2007), Wu and Ting (2007)	1995
15	3750	3063	646	375	46	Nelissen (1995a), Bhattacharya and Roy (1998), Young-Gun	1996
16	1200	800	176	135	38	and Maing-Kyu (2001), Wu and Ting (2007)	
17	34	23	5	4	38		
18	300	200	21	19	149	Scheithauer and Terno (1996), Morabito and Morales (1998), Martins and Dell (2008)	1996
19	40	25	7	3	47	Scheithauer and Terno (1996), Morabito and Morales (1998),	1996
20	52	33	9	4	47	Pureza and Morabito (2003, 2006) Wu and Ting (2007),	
21	56	52	12	5	48	Martins and Dell (2008)	
22	43	26	7	3	53	Morabito and Morales (1998), Lins et al. (2003), Pureza and Morabito (2003, 2006), Birgin et al. (2005), Wu and Ting (2007), Martins and Dell (2008)	1998
23	153	100	24	7	90	Morabito and Morales (1998), Martins and Dell (2008)	1998
24	42	39	9	4	45	Morabito and Morales (1998), Pureza and Morabito (2003, 2006)	1998
25	124	81	21	10	47	Wu and Ting (2007), Martins and Dell (2008)	

(*Continued*)

Table 1
Continued

#	*L*	*W*	*a*	*b*	*z*	Papers where the instances were used	Year
26	100	64	17	10	36	Letchford and Amaral (2001), Ribeiro and Lorena (2007)	2001
27	100	82	22	8	45		
28	100	83	22	8	45		
29	32	22	5	4	34		
30	32	27	5	4	42		
31	40	26	7	4	36		
32	53	26	7	4	48		
33	37	30	8	3	45		
34	81	39	9	7	49		
35	61	38	6	5	77	Lins et al. (2003), Pureza and Morabito (2003, 2006), Birgin et al. (2005), Wu and Ting (2007), Ribeiro and Lorena (2007), Martins and Dell (2008)	2003
36	63	44	8	5	69	Lins et al. (2003), Pureza and Morabito (2003, 2006), Birgin et al. (2005), Wu and Ting (2007), Martins and Dell (2008)	2003
37	61	35	10	3	71		
38	61	38	10	3	77		
39	93	46	13	4	82		
40	106	59	13	5	96		
41	141	71	13	8	96		
42	108	65	10	7	100		
43	86	52	9	5	99	Lins et al. (2003), Pureza and Morabito (2003, 2006), Alvarez-Valdes et al. (2005b), Birgin et al. (2005), Wu and Ting (2007)	2003
44	74	46	7	5	97		
45	67	44	6	5	97	Lins et al. (2003), Pureza and Morabito (2003, 2006), Wu and Ting (2007)	2003
46	49	28	8	3	57	Lins et al. (2003), Birgin et al. (2005), Martins and Dell (2008)	2003
47	57	34	7	4	69		
48	67	37	11	3	75		
49	67	40	11	3	81		
50	74	49	11	4	82		
51	2296	1230	136	94	219	Ribeiro and Lorena (2008), Birgin et al. (2010)	2008
52	2536	1312	144	84	273		
53	2252	1470	144	84	271		
54	1470	1458	144	84	175		
55	2296	1230	135	92	226		
56	1804	1230	137	95	169		
57	2466	1230	137	95	231		
58	1804	1750	137	95	241		
59	2426	1230	137	95	227		

in which the respective instance has been used, and the year when it appeared in the literature for the first time.

It is interesting to note that some instances appear under different names in the literature, which is the case of instance #5. It is named D8 in Arenales and Morabito (1995) and Bhattacharya and Roy (1998), D2 in Morabito and Morales (1998) and Martins and Dell (2008), and D11 in Young-Gun and Maing-Kyu (2001).

Instances #51 to #59 stem from a wood pulp stowage problem in the Brazilian ports, while the other instances, with the exception of #1, #3, #11, #12, and #15 to #18, are from the data sets "*Cover I*" and "*Cover II*," which were created using the experience from real pallet loading applications (see next subsection). The instances from "*Cover I*" have optimal solutions with less than 51 items and the instances from "*Cover II*" have optimal solutions of between 51 and 100 items. The instances from the real application have the highest number of items per pallet.

Some of the instances presented in Table 1 are equivalent, meaning that they have the same optimal loading pattern. This is the case with instances #4, #6, #7, and #8 with an optimal objective function value of 23; instances #5, #9, #10, and #11 with an optimal objective function value of 42; instances #14 and #15 with an optimal objective function value of 46; and instances #16 and #17 with an optimal objective function value of 38.

In Table 2, in order to sketch the current state of the art with respect to how the benchmark instances have been solved by exact and heuristic algorithms, we report the objective function value (z) obtained and the computing time (Time) needed by the respective most successful algorithm. Since computing times are not necessarily comparable, we also mention the computing platform on which the results were obtained.

As for the exact algorithms, in general we mention the paper in which the up-to-date fastest algorithm for an instance was presented. However, whenever several exact algorithms exist, which solved the instance in less than one second, then we simply mention the most recent paper corresponding to the algorithm that achieved such computing time. As for the heuristic algorithms, the entries of Table 2 refer to the (paper presenting the) algorithm that has obtained the best objective function value. That value is stated in column z. If there were several algorithms that provided solutions with this objective function value, we have listed the paper presenting the fastest one. If several algorithms provided those solutions in less than one second, then we refer to the most recent paper presenting such an algorithm.

In numerical experiments, exact algorithms have only been applied to a subset of the benchmark instances so far, namely to 36 of 59 instances (61.0%). All those 36 instances could be solved optimally. It is worth noting that the state of the art is represented by just two algorithms, the two-phase algorithm of Wu and Ting (2007) and algorithm of Martins and Dell (2008). Twenty (55.6%) of the 36 instances could be solved optimally in less than one second. Three instances, namely #18, #43, and #45, have proven to be particularly hard to solve. For instances #43 and #45, high computing times have also been reported for other exact solution approaches. Alvarez-Valdes et al. (2005b) have explicitly generated the above-described version of Beasley's optimization model for instance #43, to which they applied Cplex that required 89 hours for generating a solution and proving its optimality. Wu and Ting (2007) used the exact method based on Beasley's model that took 3.88 hours to provide an optimal solution. Lins et al. (2003) chose the same approach for solving instance #45, for which Cplex 7 needed 105 hours. According to Wu and Ting (2007) Cplex 9.1 solved this instance to a proven optimum in 1.38 hours. We conclude that instances #18, #43, #45, and probably #40, #42, and #44 (which required more than 100 seconds of computing time) still provide some challenge for exact solution algorithms.

Regarding the application of heuristic algorithms, we recognize that for each of the 36 instances that have been solved optimally by the exact algorithms, the same objective function value was

Table 2
Results for benchmark instances

#	Exact algorithms				Heuristic algorithms			
	Paper	z	Time (seconds)	Computing platform	Paper	z	Time (seconds)	Computing platform
1	Wu and Ting (2007)	30	<1	Pentium IV 3.0 GHz	Young-Gun and Maing-Kyu (2001)	30	<1	k6-350 MHz processor,
2		33	<1	processor, 1 GHz		33	<1	64 MB of RAM
3		23	<1	RAM—Cplex 9.1		23	<1	
4	Martins and Dell (2008)	23	<1	Pentium III 600 MHz	Martins and Dell (2008)	23	<1	Pentium III 600 MHz
5		42	<1	processor		42	<1	processor
6	Wu and Ting (2007)	23	<1	Pentium IV 3.0 GHz	Young-Gun and Maing-Kyu (2001)	23	<1	k6-350 MHz processor,
7		23	<1	processor, 1 GHz		23	<1	64 MB of RAM
8		23	<1	RAM—Cplex 9.1		23	<1	
9		42	<1			42	<1	
10		42	<1			42	<1	
11		42	<1			42	<1	
12	Wu and Ting (2007)	97	10.6	Pentium IV 3.0 GHz processor, 1 GHz RAM—Cplex 9.1	Martins and Dell (2008)	97	<1	Pentium III 600 MHz processor
13	Martins and Dell (2008)	41	<1	Pentium III 600 MHz processor	Martins and Dell (2008)	41	<1	Pentium III 600 MHz processor
14	Wu and Ting (2007)	46	2.7	Pentium IV 3.0 GHz	Young-Gun and Maing-Kyu (2001)	46	<1	k6-350 MHz processor,
15		46	7.6	processor, 1 GHz		46	<1	64 MB of RAM
16		38	1.4	RAM—Cplex 9.1		38	<1	
17		38	<1			38	<1	

(Continued)

Table 2
Continued

#	Exact algorithms				Heuristic algorithms			
	Paper	z	Time (seconds)	Computing platform	Paper	z	Time (seconds)	Computing platform
18	Martins and Dell (2008)	149	2247.8	Pentium III 600 MHz	Martins and Dell (2008)	149	<1	Pentium III 600 MHz
19		47	<1	processor		47	<1	processor
20		47	<1				47	<1
21		48	2.4				48	<1
22		53	<1				53	<1
23		90	<1				90	<1
24		45	<1				45	<1
25		47	<1				47	<1
26					Ribeiro and	36	27.0	Pentium IV
27					Lorena	45	335.0	processor,
28					(2007)	45	331.1	512 MB of
29						34	17.0	RAM—
30						42	45.2	Cplex 7.5
31						36	62.0	
32						48	287.0	
33						45	194.0	
34						49	2167.0	
35	Wu and Ting (2007)	77	1.9	Pentium IV 3.0 GHz	Martins and Dell (2008)	77	<1	Pentium III 600 MHz
36		69	5.7	processor, 1 GHz		69	3.8	processor
37		71	1.2	RAM—Cplex 9.1		71	2.6	
38		77	<1			77	3.7	
39		82	1.9			82	15.3	
40		96	101.8			96	17.8	
41		96	3.9			96	18.1	
42		100	311.2			100	2.4	

(Continued)

Table 2
Continued

#	Exact algorithms				Heuristic algorithms			
	Paper	z	Time (seconds)	Computing platform	Paper	z	Time (seconds)	Computing platform
43	Wu and Ting (2007)	99	2251.15	Pentium IV 3.0 GHz processor, 1 GHz RAM—Cplex 9.1	Birgin et al. (2005)	99	191.3	1533 MHz AMD Athlon (TM) MP 1800+ processor, 512 MB of RAM—Linux operating system
44		97	501.88			97	57.7	
45	Wu and Ting (2007)	97	3477.0	Pentium IV 3.0 GHz processor, 1 GHz RAM—Cplex 9.1	Pureza and Morabito (2006)	97	4.9	AMD Athlon XP 2600, 1.92 GHz processor
46					Martins and Dell (2008)	57	< 1	Pentium III 600 MHz processor
47						69	1.7	
48						75	3.4	
49						81	7.5	
50						82	<1	
51					Birgin et al. (2010)	219	<1	2.4 GHz Intel Core2 Quad Q6600 processor, 4.0 GB of RAM—Linux operating system
52						273	1039.7	
53						271	1344.0	
54						175	145.4	
55						226	<1	
56						169	<1	
57						231	<1	
58						241	909.4	
59						227	1706.0	

obtained by the (best) heuristic algorithm. In other words, for all these instances an optimal solution has been generated. In fact, the optimality has actually also been proven by the respective heuristic algorithms themselves since in every case a solution could be identified that met an incorporated upper bound. Unlike in the case of exact algorithms, not just two methods contribute to the current state of the art but a larger variety of methods does, namely the recursive 5-block heuristic (instances #1 to #3, #6 to #10, #14 to #17) by Young-Gun and Maing-Kyu (2001), the G-5 heuristic (#4, #5, #12, #13, #18 to #25) and the higher order nonguillotine heuristic (#35 to #42, #46 to #50) by Martins and Dell (2008), the heuristic (#26 to #34) by Ribeiro and Lorena (2007), the L-shaped-block heuristic (#43 to #44) and the recursive partitioning algorithm (#51 to #59) by Birgin et al. (2010) and the tabu search algorithm (#45) by Pureza and Morabito (2006).

Not unexpectedly for heuristic algorithms, computing times were significantly shorter than for the corresponding exact algorithms for most of the 36 instances. Where this appeared not to be the case (i.e. for instances #37, #38, #39, #41), it can be attributed to differences of the processors that have been used. For 26 of the 36 instances (72.2%), the computing times were negligible (less than one second), and for the instances that took exact methods particularly long to solve (instances #18, #40, #42 to #45, see above) computing times could be reduced dramatically (see instances #18, #43 and #45, in particular). By means of implemented upper bounds, it could also be shown that the heuristic algorithms have provided optimal solutions to those instances that have not been tackled by exact algorithms so far (instances #26 to #34, #46 to #59), that is, the values given in the z-column represent optimal objective function values. From the computing times, it can be taken that instances #52 to #54, #58, and #59 in particular represent some challenges for heuristic algorithms and may serve as benchmark instances in future. The same could also hold for instances #26 to #34, even though computing times may not look so problematic if more recent computer hardware would be used. The long computing times for heuristic algorithms indicate that these instances should also serve as benchmark instances for exact algorithms in future.

6.2. Testbeds

In this section, we analyze what kinds of data sets have been used in numerical experiments for the evaluation of exact and heuristic methods and how they have been generated. Table 3 gives an overview of data sets that have been referred to in numerical experiments for the evaluation of exact and/or heuristic algorithms in at least two publications. In particular, we provide additional information on how the problem parameters have been set for the generation of problem classes and the respective problem instances.

The first column identifies the data set, with the following four columns presenting the length and width of the pallet and the length and width of the item. The following columns provide the area ratio, the ratios of the length to width for the pallet and the item as criteria for the characterization of the data set. Finally in the last column, we reference the papers where the data sets have been used.

For data set A (Steudel, 1979), the pallet data (length L, width W) is fixed to (48, 40). The dimensions (a, b) of the small item is determined deterministically by considering explicitly all lengths from the interval [7, 15] and all widths from the interval [5, 14] in steps of 0.5 and combining them

Table 3
Data sets used often in the literature

Set	L	W	a	b	$\frac{WL}{ab}$	$\frac{L}{W}$	$\frac{a}{b}$	References
A	48	40	[7, 15]	[5, 14]				Steudel (1979, 1984)
B	1060	813	[260, 379]	[100, 219]				Smith and De Cani (1980), Dowsland (1987b)
C	1200	1000	[200, 600]	[150, 450]				Dowsland (1984, 1987b), Morabito and Morales (1998), Morabito et al. (2000)
D					[6, 50]	[1, 2]	[1, 4]	Dowsland (1984, 1985a)
E	1200	1000			[1, 51[		[1, 4]	Dowsland (1987b), Bhattacharya and Roy (1998)
F	1100	1100			[1, 51[		[1, 4]	Dowsland (1987b), Bhattacharya and Roy (1998)
G					[1, 51[	[1, 2]	[1, 4]	Dowsland (1987a, 1987b), Morabito and Morales (1998), Amaral and Wright (2001)
H					[1, 51[	[1, 2]	[1, 4]	Scheithauer and Terno (1996), Morabito and Morales (1998), Morabito and Farago (2002), Alvarez-Valdes et al. (2005a, 2005b), Birgin et al. (2005, 2010)
I					[51, 100]	[1, 2]	[1, 4]	Nelissen (1995a), Scheithauer and Terno (1996), Morabito and Morales (1998), Lins et al. (2003), Alvarez-Valdes et al. (2005a, 2005b), Birgin et al. (2005, 2010)
J					[101, 150]	[1, 2]	[1, 4]	Alvarez-Valdes et al. (2005b), Birgin et al. (2010)
K					$\leq$100			Martins and Dell (2007, 2008)

with each other. From all potential combinations, Steudel (1979, 1984) selected 182 as dimensions of the small item. In Steudel (1979), the instances were solved by means of a 4-block heuristic and the solutions were compared to the ones used in practice by the U.S. Navy. Improvements were obtained for 64 cases. Average computing times per problem instance were 1.25 seconds. Later, in Steudel (1984), it was reported that another 14 instances have been solved better than in practice. It has to be noted that both for the solutions provided by the heuristic but also for the solutions encountered in practice limited overhang was permitted.

Data set B, introduced by Smith and De Cani (1980), is similar to set A. Here, the items are generated by unit increments within the respective intervals, resulting in 14,400 instances. The authors applied their 4-block heuristic to these instances and solved them in 23 minutes (total computing time for all instances). By a comparison of the obtained objective function values to the values of the area bound, it could be shown that 14.1% of the instances had been solved optimally. Dowsland (1987b), instead of considering all integer lengths and widths explicitly, generated a set of instances randomly by assuming uniformly distributed lengths and widths in the respective intervals. One thousand instances were generated and optimally solved in less than 20 seconds (total computing time) by the exact tree search algorithm proposed by the author.

In data set C, originally introduced by Dowsland (1984), the dimensions of the pallet are fixed to the length (1200 mm) and the width (1000 mm) of the so-called "EURO pallet." The lengths and widths of the small items are assumed uniformly distributed within the intervals depicted in Table 3. In Dowsland (1987b), 1000 instances were generated randomly and optimally solved by

Dowsland's exact algorithm in less than 10 seconds. Morabito and Morales (1998) generated 100 instances. Their recursive 5-block heuristic solved all instances in an average computing time of 0.1 seconds per instance. By referring to the bound obtained from the linear relaxation of Beasley's model, they could prove that for all instances an optimal solution had been obtained.

Unlike in the data sets A, B, and C, in the following data sets (except for set K), ratios regarding the (relative) size of the small items in relation to the size of the large object ($\frac{L \cdot W}{a \cdot b}$; area ratio), the (relative) width of the pallet in relation to its length (L/W; pallet length/width ratio), and the (relative) width of the small item in relation to its length (a/b; item length/width ratio) are used in order to specify relevant problem properties.

As for data set D of Dowsland (1984), the instances are obtained by randomly selecting the values of the above-mentioned parameters from uniform distributions in the intervals [6, 50] (for $\frac{L \cdot W}{a \cdot b}$), [1, 2] (for $\frac{L}{W}$), and [1, 4] (for $\frac{a}{b}$). We would like to point out that the area ratio imposes a limit of 50 small items on the optimal objective function value of each instance. Dowsland (1985a) also fixed the length of the pallet to $L = 1000$ and generated 5000 problem instances from this set. The author combined the 5-block heuristic by Bischoff and Dowsland (1982) with the perfect-partition bound and managed to prove (by verifying the identity of upper and lower bound) that 72% of the instances had been solved to an optimum. The total computing time for solving all instances amounted to 266.3 seconds.

Sets E and F are designed rather similarly and do only differ with respect to the dimensions of the pallet. Length and width are fixed to the dimensions of the EURO pallet, that is, $(L, W) = (1200, 1000)$, in set E, and to $(L, W) = (1100, 1100)$, that is, a square pallet, in set F. In both cases the problem instances are obtained by randomly selecting the values of $(\frac{L \cdot W}{a \cdot b})$ and $(\frac{a}{b})$ from uniform distributions in the intervals [6, 51[and [1, 4]. Again, as a consequence from the assumed area ratio, for each instance the number of small items that can be packed on the pallet cannot exceed 50. Dowsland (1987b) generated 1000 instances for each data set. By means of her exact tree search algorithm, all instances except for three in set E and 10 in set F could be solved within a time limit of 100 seconds per instance. Bhattacharya and Roy (1998) also generated 1000 instances for each data set and applied their exact tree search procedure to these instances. They managed to solve all of them optimally with an average computing time of 0.99 seconds for the instances of set E and 1.44 seconds for the instances of set F.

Also for data set G, the problem instances are obtained by randomly selecting the values of the above-mentioned parameters, namely from uniform distributions in the intervals [1, 51[(for $\frac{L \cdot W}{a \cdot b}$), [1, 2] (for $\frac{L}{W}$), and [1, 4] (for $\frac{a}{b}$), allowing at most 50 small items to be packed on the pallet. Dowsland (1987b) generated 1000 instances, to which she applied her exact algorithm. Within 100 seconds, all but three instances were solved optimally. In Dowsland (1987a) the same exact algorithm was combined with an improved upper bound derived from graph coloring. This algorithm solved all 1000 instances to an optimum in less than one minute. Morabito and Morales (1998) generated 100 instances from set G and solved all of them optimally by means of their recursive 5-block heuristic. The average computing time per instance was 0.4 seconds. Amaral and Wright (2001) differentiated their experiments with respect to three classes of instances, namely instances in which the optimal number of items which can be packed is between (i) 10–20, (ii) 21–30, and (iii) 31–40. For each class, 100 instances were generated and solved by their strategic oscillation algorithm. Within a time limit (per instance) of 600 seconds they could prove—by means of the area bound—that 94.8% (class i), 84.3% (class ii), and 37% (class iii) had been solved optimally.

The generation of test problem instances from all the data sets dealt with in this section so far suffers from the drawback that some or even many instances may belong to the same equivalence class, that is, test problem sets will contain subsets of practically identical problems. Thus, results from numerical experiments carried out with such test problem sets can be severely biased.

In order to overcome this drawback, one may restrict the instances included in the data set G to one representative of each equivalence class. In Table 3, this set of representatives is depicted as data set H. Nelissen (1995b), according to Morabito and Farago (2002, p. 73), has provided such a data set in which each equivalence class of set G is represented by a single instance. This set, which was named "*Cover I*" by Nelissen (1995b), consists of 8274 instances. From this set, Morabito and Morales (1998) considered only instances with $(L, W) \leq (1000, 1000)$. Their recursive 5-block heuristic solved all 3183 instances to a proven optimum. The average computing time per instance was 0.2 seconds. In another numerical experiment, Morabito and Farago (2002) applied the same filtering criteria to Nelissen's data set but only received 3179 problem instances. Birgin et al. (2005) solved all these instances to a proven optimum by means of their L-shaped-block heuristic. In Birgin et al. (2010), the authors applied their heuristic recursive partitioning algorithm to the entire set and solved all 8274 instances optimally in 28.94 seconds (total computing time for all instances).

Alvarez-Valdes et al. (2005a, b) argued that the generation of a complete set of representatives for the equivalence classes of a data set (which is defined by fixing the range for specific problem parameters as described above) may not be unique. First, it has to be decided which member of an equivalence class should serve as its representative. Second, the generation of an instance according to given parameter ranges may result in an equivalence class of which some members do not fall into the prespecified parameter ranges. This gives rise to the question whether this equivalence class should be included in the set of representatives. The authors decided to have each equivalence class represented by its *minimum size instance* (MSI). An instance (L^*, W^*, a^*, b^*) is called an MSI of an equivalence class of instances if for all instances (L, W, a, b) in the equivalence class the following holds: $L^* \leq L$, $W^* \leq W$, $a^* \leq a$, $b^* \leq b$. Furthermore, the authors also included the respective ambiguous equivalence classes in the set of representatives. On the basis of these decisions, they identified a set of 7827 problem instances as representatives of all the equivalence classes determined by the parameter ranges set for data set H. We note that the size of this set is obviously different to the size of Nelissen's set, which happens because some equivalence classes have elements in different ranges. That is, there are instances of the same class where $\frac{L}{W} \leq 2$ and other instances where this ratio is greater than 2. The set provided by Alvarez-Valdez et al. was named "*Cover IB*" by Birgin et al. (2010), while Nelissen's set was renamed "*Cover IA*." The authors solved all instances of this set to a proven optimum by applying their heuristic recursive partitioning algorithm. In total, 150.54 seconds of computing time were needed. All instances of this set have also been solved to a proven optimum by Alvarez-Valdes et al. (2005b), who applied their tabu search algorithm with an imposed limit on the number of iterations which kept the computing time per instance below one second.

The definition of data set I differs from data set H only with respect to the range ([51, 100]) from which the area ratio is taken from. The set parameter values imply that up to 100 small items may be packed on the pallet, and, thus, it may be assumed that the instances of this set may be harder to solve than those of set H. In order to avoid the inclusion of equivalent problem instances in test problem sets, it makes sense again to consider in set I the representatives of the corresponding

equivalence classes only. Nelissen has provided a set of 41,831 problem instances, each of which representing a particular equivalence class of instances. At first, this set was named "*Cover II*" by Nelissen (1995b), but later renamed "*Cover IIA*" by Birgin et al. (2010). Of this set, Morabito and Morales (1998) confined themselves to instances with $(L, W) \leq (1000, 1000)$, which defines a subset of 16,938 instances. Their recursive 5-block heuristic solved 99.9% of the instances to a proven optimum and left only 18 instances for which no optimal solution was found or for which the optimality could not be verified. The average computing time per instance amounted to 3.7 seconds. Lins et al. (2003) considered 16 of these instances (for the remaining two instances the optimality was proven by the optimal solution derived from Beasley's model) and obtained proven optimal solutions for all of them by means of their L-shaped-block heuristic. The hardest instance required almost 369 minutes of computing time. Birgin et al. (2005) applied their version of the L-shaped-block heuristic to the same 16 instances and solved them all optimally, for which they needed 91.06 seconds of computing time on average. The hardest instance required was 327.88 seconds. Besides the 16 instances, in Birgin et al. (2005) all 16,938 instances are solved to a proven optimum with an average computing of 113.10 seconds. The heuristic recursive partitioning algorithm of Birgin et al. (2010) solved all 41,831 instances to a proven optimum, for which 0.16 seconds were needed on average per instance.

For the problem parameters of data set I, Alvarez-Valdes et al. (2005a, b) also generated—according to the principles presented above—a set of representatives for the respective equivalence classes. This set consists of 40,609 instances and has been named *"Cover IIB"* by Birgin et al. (2010). Alvarez-Valdes et al. (2005a) found that, by means of well-established heuristics (5-block heuristic, G4 heuristic), Nelissen's upper bound, and solving the linear relaxation of Beasley's model, already 40,368 could be solved to an optimum in a rather straightforward way. To the remaining 241 "hard" instances they applied their branch-and-cut algorithm. Within a time limit of 15 hours, only five instances could not be proven of having been solved to an optimum. Birgin et al. (2010) solved all 40,609 instances to a proven optimum by application of their heuristic recursive partitioning algorithm, for which an average computing time per instance of 0.71 seconds was needed. Alvarez-Valdes et al. (2005b) solved all instances except for one to a proven optimum by their tabu search algorithm, where they had imposed a computing time limit of five minutes per instance.

Data set J of Table 3 was proposed by Alvarez-Valdes et al. (2005b) who named it *"Cover III*." It differs from the previously discussed set only with respect to the number of items that can be placed maximally on the pallet. According to the range of its area ratio, up to 150 small items may be accommodated. The set includes 98,016 instances, each of which is a representative (the smallest instance) from a corresponding equivalence class. The authors applied their tabu search algorithm with an imposed limit on the number of iterations that kept the computing per instance below five minutes and solved 98.96% of the instances of *Cover III* to a proven optimum. Birgin et al. (2010) also applied their heuristic recursive partitioning algorithm to the instances of this set, which they named *"Cover IIIB*." The authors could show that they obtained optimal solutions for 97,333 instances, which corresponds to 99.30% of the tested cases. For the remaining 683 instances optimality is still unproven.

In Martins and Dell (2007), it is proven that each such class has one and only one MSI. Data set K contains the MSI for all equivalence classes for pallet loading problems characterized by an area ratio not larger than 100, that is, $\frac{L \cdot W}{a \cdot b} \leq 100$. Its size amounts to 3,080,730 instances. In Martins and Dell (2008), 3,073,724 of these instances (99.77%) were immediately solved optimally by the G4

heuristic and another 54 by other heuristics. The remaining 6952 instances were solved optimally by their exact HVZ algorithm. However, 226 instances required more than one hour, and 23 instances even more than 24 hours of computing time.

Benchmark instances #51 to #59 and data sets *Cover IA*, *Cover IB*, *Cover IIA*, *Cover IIB*, and *Cover IIIB*, together with the corresponding numerical results presented in Birgin et al. (2010) are available at http://lagrange.ime.usp.br/~lobato/packing/. Also the remaining instances from *Cover IIIB* for which the solutions obtained by the recursive partitioning algorithm could not be proven as having been solved optimally can be found on this web page. Furthermore, also the recursive 5-block heuristic from Morabito and Morales (1998), the L-shaped-block heuristic of Lins et al. (2003), and the recursive partitioning algorithm of Birgin et al. (2010) can be run. The respective source codes are available for download.

7. Conclusions

This paper dealt with a review of methods proposed in the literature for the solution of the MPLP. Furthermore, also an analysis of specific problem instances (benchmark instances) and more substantial sets of instances (testbeds) that have been used in computational experiments was presented. Even though this study has been rather comprehensive, it is not straightforward to identify exact or heuristic methods that are superior to others of their kind. In general, not only different hardware that has been used in numerical experiments, but also different operating systems, different programming languages, different individual encoding styles, etc., make such conclusions difficult. What complicates matters here is the fact that the methods under discussion have not always been tested on the same data sets. In the light of these preliminary remarks, the following conclusions should be drawn particularly carefully.

As for exact algorithms, the methods of Wu and Ting (2007) and Martins and Dell (2008) have been identified as the methods that solved the benchmark instances best. The excellent performance of the latter is also reflected by the results from its application to the instances of data set K. Unfortunately, nothing is known about its performance on instances of other problem classes. Nevertheless, we still consider this algorithm as a state-of-the-art exact solution method. Due to its excellent performance on the instances of problem classes H and I, we would also include the branch-and-cut algorithm proposed by Alvarez-Valdes et al. (2005a) here. The method of Wu and Ting (2007) could also be a candidate; however, no evidence is available on how it performs on the instances of the testbeds.

As far as heuristic methods are concerned, five algorithms provided extraordinary results for the benchmark instances. Of these algorithms, only the recursive partitioning algorithm of Birgin et al. (2010) has been evaluated on the instances of one of the problem classes (class J). Due to its excellent performance on this class, we consider this algorithm as one of the state-of-the-art heuristic solution methods. We further include the tabu search algorithm of Alvarez-Valdes et al. (2005a) in this class of methods as it also gave excellent results for the instances of the problem classes J, and also for the classes H and I.

The performance of the reviewed method, in particular of those which have been characterized as presenting the current state of the art, seems to indicate that almost every instance of the MPLP has already been solved or, at least, can be solved to optimality in reasonable computing time. Thus, the

question arises: Is there still room for research on the MPLP? In the remaining part of the paper, we will take a closer look at some details of our review and point out some shortcomings and the corresponding challenges.

First, we remark that among the benchmark instances we have identified some cases that still require a relatively high computational effort for the best exact and/or heuristic methods. This is the case for instances #18, #40, and #42 to #45 for which the computing times of the best exact methods amounted to more than 100 seconds. Thus, these instances, as well as instances #26 to #34 and #46 to #59 to which only heuristics have been applied so far, should be considered in future numerical experiments with exact methods. Particularly interesting is the fact that a subset of these instances (i.e. instances #51 to #59) represents data from a practical application. We note that these instances are characterized by optimal solutions that contain more than 200 items. Apart from the respective study (Birgin et al., 2010), instances of that size have not been considered in other numerical experiments. Thus, research on instances with more than 200 items would be a straightforward consequence, not just from a theoretical point of view but also because it could provide results that give better solutions to a practical problem.

Second, it has to be noted that not all classes of problem instances have been solved satisfactorily so far. This is particularly true for data set J (i.e. *Cover III*, *Cover IIIB*) with optimality still unproven for 683 equivalence classes. Interestingly, in this set the items are assumed to be relatively small in relation to the size of the pallet, and the number of items accommodated in the optimal solutions can be larger than in data sets previously used in numerical experiments. As has been observed for the benchmark instances from the practical application discussed above, the results for the instances of data set J suggest again that the MPLP gets harder to solve when the number of items in the optimal solution increases. This observation may necessitate additional research on algorithm development in order to improve solution quality and/or computing time and allow for solving instances with a number of items larger than what can currently be tackled successfully.

Third, we further notice that—for the challenging data sets that have recently been considered in numerical experiments—the ratio of the length to the width of the items is chosen from the range [1, 4]. Hardly anything is known for instances that are characterized by other ratios. If these values do not reflect the data structure of a wide range of practical problems, the current methods cannot necessarily be expected to solve real-world instances efficiently. Thus, we suggest that in future also other length-width ratios are considered in numerical experiments.

Finally, we would like to mention again that also the computational complexity of the MPLP still is an open problem. So also from a theoretical point of view, the MPLP continues to provide some challenge. Therefore, in order to sum up, we conclude that there are still several severe shortcomings with respect to what we know about the MPLP, which necessitate, ongoing research in the area.

Acknowledgments

This research was partially supported by ERDF through the Programme COMPETE and by the Portuguese Government through FCT (CPackBenchFrame-PTDC/EIA-CCO/115878/2009) and the Project BEST CASE-SAESCTN-PIIC\&DT/1/2011, which is co-financed by the

North Portugal Regional Operational Programme (ON.2-O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF).

References

Alvarez-Valdes, R., Parreño, F., Tamarit, J.M., 2005a. A branch-and-cut algorithm for the pallet loading problem. *Computers & Operations Research* 32, 3007–3029.

Alvarez-Valdes, R., Parreño, F., Tamarit, J.M., 2005b. A tabu search algorithm for the pallet loading problem. *OR Spectrum* 27, 43–61.

Amaral, A.R.S., Wright, M., 2001. Experiments with a strategic oscillation algorithm for the pallet loading problem. *International Journal of Production Research* 39, 2341–2351.

Arenales, M., Morabito, R., 1995. An AND/OR-graph approach to the solution of two-dimensional non-guillotine cutting problems. *European Journal of Operational Research* 84, 599–617.

Arslanov, M.Z., 2000. Continued fractions in optimal cutting of a rectangular sheet into equal small rectangles. *European Journal of Operational Research* 125, 239–248.

Barnes, F.W., 1979. Packing the maximum number of m x n tiles in a large p x q rectangle. *Discrete Mathematics* 26, 93–100.

Barnett, S., Kynch, G.J., 1967. Exact solution of a simple cutting problem. *Operations Research* 15, 1051–1056.

Beasley, J., 1985. An exact two-dimensional non guillotine cutting tree search procedure. *Operations Research* 33, 49–64.

Bhattacharya, S., Roy, R., 1998. An exact depth-first algorithm for the pallet loading problem. *European Journal of Operational Research* 110, 610–625.

Birgin, E.G., Lobato, R., Morabito, R., 2010. An effective recursive partitioning approach for the packing of identical rectangles in a rectangle. *Journal of the Operational Research Society* 61, 306–320.

Birgin, E.G., Lobato, R.D., Morabito, R., 2012. Generating unconstrained two-dimensional non-guillotine cutting patterns by a recursive partitioning algorithm. *Journal of the Operational Research Society* 63, 183–200.

Birgin, E.G., Morabito, R., Nishihara, F.H., 2005. A note on an L-approach for solving the manufacturer's pallet loading problem. *Journal of the Operational Research Society* 56, 1448–1451.

Bischoff, E., Dowsland, W.B., 1982. An application of the micro to product design. *Journal of the Operational Research Society* 33, 271–280.

Bischoff, E.E., Ratcliff, M.S.W., 1995. Loading multiple pallets. *Journal of the Operational Research Society*, 46, 1322–1336.

Christofides, N., Whitlock, C., 1977. An algorithm for two-dimensional cutting problems. *Operations Research* 25, 30–44.

Daniels, J.J, Ghandforoush, P., 1990. An improved algorithm for the non-guillotine-constrained cutting-stock problem. *Journal of the Operational Research Society* 41, 141–149.

Dowsland, K.A., 1984. The three-dimensional pallet chart: an analysis of the factors affecting the set of feasible layouts for a class of two-dimensional packing problems. *Journal of the Operational Research Society* 35, 895–905.

Dowsland, K.A., 1985a. Determining an upper bound for a class of rectangular packing problems. *Computers & Operations Research* 12, 201–205.

Dowsland, K.A., 1987a. A combined data-base and algorithmic approach to the pallet-loading problem. *Journal of the Operational Research Society* 38, 341–345.

Dowsland, K.A., 1987b. An exact algorithm for the pallet loading problem. *European Journal of Operational Research* 31, 78–84.

Dowsland, K.A., 1993. Some experiments with simulated annealing techniques for packing problems. *European Journal of Operational Research* 68, 389–399.

Dowsland, W.B., 1985b. Two and three dimensional packing problems and solution methods. *New Zealand Operational Research* 13, 1–18.

Exeler, H., 1991. Upper bounds for the homogeneous case of a two-dimensional packing problem. *ZOR—Methods and Models of Operations Research* 35, 45–58.

Fowler, R.J., Paterson, M.S., Tanimoto, S.L., 1981. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters* 12, 133–137.

George, J.A., Furness, C.G., Scott, T.J., 1992. A correction of the improved bounds for the non-guillotine constrained cutting-stock problem. *Journal of the Operational Research Society* 43, 1187–1189.

Herbert, E.A., Dowsland, K.A., 1996. A family of genetic algorithms for the pallet loading problem. *Annals of Operations Research* 63, 415–436.

Herz, J.C., 1972. Recursive computational procedure for two-dimensional stock cutting. *IBM Journal of Research and Development* 16, 462–469.

Hodgson, T.J., 1982. A combined approach to the pallet loading problem. *IIE Transactions* 14, 175–182.

Isermann, H., 1987. Ein Planungssystem zur Optimierung der Palettenbeladung mit kongruenten rechteckigen Versandgebinden. *Operations Research Spektrum* 9, 235–249.

Letchford, A.N., Amaral, A., 2001. Analysis of upper bounds for the pallet loading problem. *European Journal of Operational Research* 132, 582–593.

Lins, L., Lins, S., Morabito, R., 2003. An L-approach for packing (l, w)-rectangles into rectangular and L-shaped pieces. *Journal of the Operational Research Society* 54, 777–789.

Martins, G.H.A., Dell, R.F., 2007. The minimum size instance of a pallet loading problem equivalence class. *European Journal of Operational Research* 179, 17–26.

Martins, G.H.A., Dell, R.F., 2008. Solving the pallet loading problem. *European Journal of Operational Research* 184, 429–440.

Morabito, R., Arenales, M., 1994. An AND/OR-graph approach to the container loading problem. *International Transactions in Operational Research* 1, 59–73.

Morabito, R., Farago, R., 2002. A tight Lagrangean relaxation bound for the manufacturer's pallet loading problem. *Studia Informatica Universalis* 2, 57–76.

Morabito, R., Morales, S., 1998. A simple and effective recursive procedure for the manufacturer's pallet loading problem. *Journal of the Operational Research Society* 49, 819–828.

Morabito, R., Morales, S.R., Widmer, J.A., 2000. Loading optimization of palletized products on trucks. *Transportation Research Part E: Logistics and Transportation Review* 36, 285–296.

Nelissen, J., 1993. New approaches to the pallet loading problem, Working paper, Schriften zur Informatik und Angewandten Mathematik, *RWTH Aachen*.

Nelissen, J., 1995a. How to use structural constraints to compute an upper bound for the pallet loading problem. *European Journal of Operational Research* 84, 662–680.

Nelissen, J., 1995b. *Neue Ansätze zur Lösung des Palettenbeladungsproblems*. Aachen, Verlag Shaker.

Pureza, V., Morabito, R., 2003. Uma heurística de busca tabu simples para o problema de carregamento de paletes do produtor. *Pesquisa Operacional* 23, 359–378.

Pureza, V., Morabito, R., 2006. Some experiments with a simple tabu search algorithm for the manufacturer's pallet loading problem. *Computers & Operations Research* 33, 804–819.

Ribeiro, G.M., Lorena, L.A.N., 2007. Lagrangean relaxation with clusters and column generation for the manufacturer's pallet loading problem. *Computers & Operations Research* 34, 2695–2708.

Ribeiro, G.M., Lorena, L.A.N., 2008. Optimizing the woodpulp stowage using Lagrangean relaxation with clusters. *Journal of the Operational Research Society* 59, 600–606.

Scheithauer, G., Terno, J., 1996. The G4-heuristic for the pallet loading problem. *Journal of the Operational Research Society* 47, 511–522.

Smith, A., De Cani, P., 1980. An algorithm to optimize the layout of boxes in pallets. *Journal of the Operational Research Society* 31, 573–578.

Steudel, H.J., 1979. Generating pallet loading patterns: a special case of the two-dimensional cutting stock problem. *Management Science* 25, 997–1004.

Steudel, H.J., 1984. Generating pallet loading patterns with considerations of item stacking on end and side surfaces. *Journal of Manufacturing Systems* 3, 135–143.

Tarnowski, A.G., Terno, J., Scheithauer, G., 1994. A polynomial time algorithm for the guillotine pallet loading problem. *INFOR: Information Systems and Operational Research* 32, 275–287.

Tsai, R.D., Malstrom, E.M., Kuo, W., 1993. Three dimensional palletization of mixed box sizes. *IIE Transactions* 25, 64–75.

Wäscher, G., Haußner, H., Schumann, H., 2007. An improved typology of cutting and packing problems. *European Journal of Operational Research* 183, 1109–1130.

Wu, K.C., Ting, C.J., 2007. A two-phase algorithm for the manufacturer's pallet loading problem. In Helander, M., Xie, M., Jiao, Tan, C.T. (eds), *Proceedings of the 2007 IEEE International Conference on Industrial Engineering and Engineering Management*. Singapore, pp. 1574–1578.

Yi, J., Chen, X., Zhou, J., 2009. The pinwheel pattern and its application to the manufacturer's pallet-loading problem. *International Transactions in Operational Research* 16, 809–828.

Young-Gun, G., Maing-Kyu, K., 2001. A fast algorithm for two-dimensional pallet loading problems of large size. *European Journal of Operational Research* 134, 193–202.