Chapter 7
A Container Loading Problem MILP-Based Heuristics Solved by CPLEX: An Experimental Analysis

Stefano Gliozzi, Alessandro Castellazzo, and Giorgio Fasano

Abstract The issue of placing small boxes orthogonally, generally with the possibility of rotations, into a big box, maximizing the loaded volume, is usually referred to as the container loading problem. Despite its being notoriously of an NP-hard typology, a number of algorithms work out this problem very efficiently. The task becomes, nonetheless, even more challenging when additional conditions have to be taken account of. In such cases, a modeling-based approach is supposedly the most suitable and this definitely holds, in particular, when balancing requirements are posed. These, indeed, entail constraints of strong global impact that can hardly be coped with by sequential procedures, based on a step by step incremental loading of items.

MIP (Mixed Integer Programming) models relevant to the container loading problem or possible extensions of it are available in specialized literature. A dedicated MILP (Mixed Integer Linear Programming) formulation, supporting an overall heuristic approach, addressed to non-standard packing issues, is discussed in another chapter of this book. Hereinafter, some relevant computational aspects are looked into, restricting the consideration to the container loading problem, as per its classical statement. An ad hoc heuristics, derived from the above-mentioned overall approach, is outlined. The use of IBM ILOG CPLEX as an MILP optimizer is considered. Case studies concerning the solution of the MILP model tout court, when the instances involved are not of a large-scale nature, are reported first.

S. Gliozzi (✉)
IBM Italia S.p.A., Rome, Italy
e-mail: Stefano_gliozzi@it.ibm.com

A. Castellazzo
Altran Italia S.p.A. Consultant c/o Thales Alenia Space Italia S.p.A., Turin, Italy
e-mail: alessandro.castellazzo@external.thalesaleniaspace.com

G. Fasano
Thales Alenia Space Italia S.p.A., Turin, Italy
e-mail: giorgio.fasano@thalesaleniaspace.com

G. Fasano, J.D. Pintér (eds.), *Optimized Packings with Applications*, Springer Optimization and Its Applications 105, DOI 10.1007/978-3-319-18899-7_7

Outcomes relevant to the ad hoc heuristics are further shown through a number of difficult instances. Examples of container loading issues, involving also balancing conditions, are additionally provided.

Keywords Container loading • Orthogonal packing with rotations • Mixed integer linear programming • MILP model • Heuristics • CPLEX • Computational results

7.1 Introduction

Container loading is a typical packing problem, concerning the orthogonal placement of small boxes (i.e., rectangular parallelepipeds) into a big box, maximizing the loaded volume. A very large number of specialist works are devoted to this subject and the reader is referred to the available literature for a wide-ranging overview (e.g., [1]). Hereinafter, we shall recall the modeling-based methodology discussed in depth in a dedicated chapter of this book [2].

This approach has been conceived to solve complex non-standard packing problems, allowing for tetris-like items inside convex domains, with additional conditions, such as balancing. The container loading problem, as per its classical formulation, represents a specific case addressed by the general MILP (Mixed Integer Linear Programming) mathematical model discussed in Fasano [2], Section 2. This can be utilized, directly, when a limited number of items are involved. Otherwise, when large-scale instances have to be coped with, the above-mentioned MILP model represents the basic "engine" of the overall heuristic approach outlined in Fasano [2], Section 6. Specific versions of this model are, in such cases, adopted to support all the relevant phases of the whole heuristic process, i.e.: *Initialization*, *Packing*, *Item-exchange*, *Hole-filling*. The present chapter focuses on what we currently consider the most promising solution strategies relevant to the modeling-based approach in question. These act at two different levels.

Firstly, an ad hoc strategy, delineated in Sect. 7.2.1, has been looked into for the above-mentioned overall heuristic procedure. As pointed out in Fasano [2], Section 6, indeed, the way the various modules (i.e., *Initialization*, *Packing*, *Item-exchange*, *Hole-filling*) are activated/executed actually determines a specific heuristics.

Secondly, dedicated MILP strategies have been studied to work out the general MILP model, as utilized in its different versions, i.e. either when the container loading problem is tackled tout court, or the various phases of the heuristic process have to be performed. As is well known, when managing an MILP model, the solution search effectiveness is strongly affected by the general features of the optimizer adopted, but even more by the way it is "driven." For instance, different branch and bound (B&B) strategies may yield very different outcomes, both in terms of solution quality and computational effort. In our research, IBM ILOG CPLEX [3] has been selected as the MILP solver and appropriate *drivers* set up to solve the MILP model, in its various versions, efficiently. This is the subject of Sect. 7.2.2.

A dedicated experimental analysis, covering non-trivial instances, has been carried out. Although the strategies proposed here are suitable for a number of non-standard packing issues, not limited to the classical container loading problem, our attention has been concentrated on it. In this case, the procedure put forward in the present chapter (as well as the modeling-based approach in general), being aimed at non-standard applications, is typically outperformed by most of the off-the-shelf algorithms, specific for the classical container loading problem (e.g., [4]). The choice of focusing on this problem in particular, however, aims to offer a useful reference in a standard test framework. Section 7.3.1 reports computational results relevant to the direct solution of the MILP model, with small-scale instances. Section 7.3.2 shows outcomes regarding demanding test cases, solved by the heuristics of Sect. 7.2.1. Insights concerning the presence of balancing conditions are additionally provided in Sect. 7.3.3, to draw the reader's attention to an application quite frequent in practice.

7.2 Solution Search Strategies

7.2.1 Heuristic Approach

A specific procedure deriving from the overall heuristic approach proposed in Fasano [2] is outlined in this section (in a streamlined form). Major interrelated concepts are those of *relative position* and *abstract configuration*. A *relative position* between two items expresses that one, with respect to the other, is located in compliance with one of the following conditions: on the left, on the right, in front, behind, above or below. An *abstract configuration*, relative to N items, is a set of $\frac{N(N-1)}{2}$ *relative positions*, one for each pair of items, that are feasible (i.e., all of them can be respected) in any unbounded domain. With a given *abstract configuration* items may be rotated and translated, keeping their *relative positions* unaltered.

The underlying idea of the overall heuristic approach is to generate a sequence of *abstract configurations* that allow the feasible placement (i.e., with no overlapping) of an increasing number of items in the given domain (i.e., the container), maximizing the loaded volume.

In the specific heuristics proposed in this chapter, the whole process is split into two macro-phases, i.e. the *main* and the *incremental* one, respectively, see Fig. 7.1. Both of them activate the modules of the overall heuristic approach (i.e., *Initialization*, *Packing*, *Item-exchange*, *Hole-filling*) sequentially. The macro-phases are executed recursively, performing a number of cycles. Items are added, time after time, following an overall *greedy* approach. At each module execution, the selection of items as candidates for loading (in addition to those previously accepted) is made on a larger-first priority criterion. This way, the procedure attempts to load items with the largest volumes whilst the domain is still quite unexploited. On the contrary, the smaller ones are tentatively introduced to fill the empty spaces, when

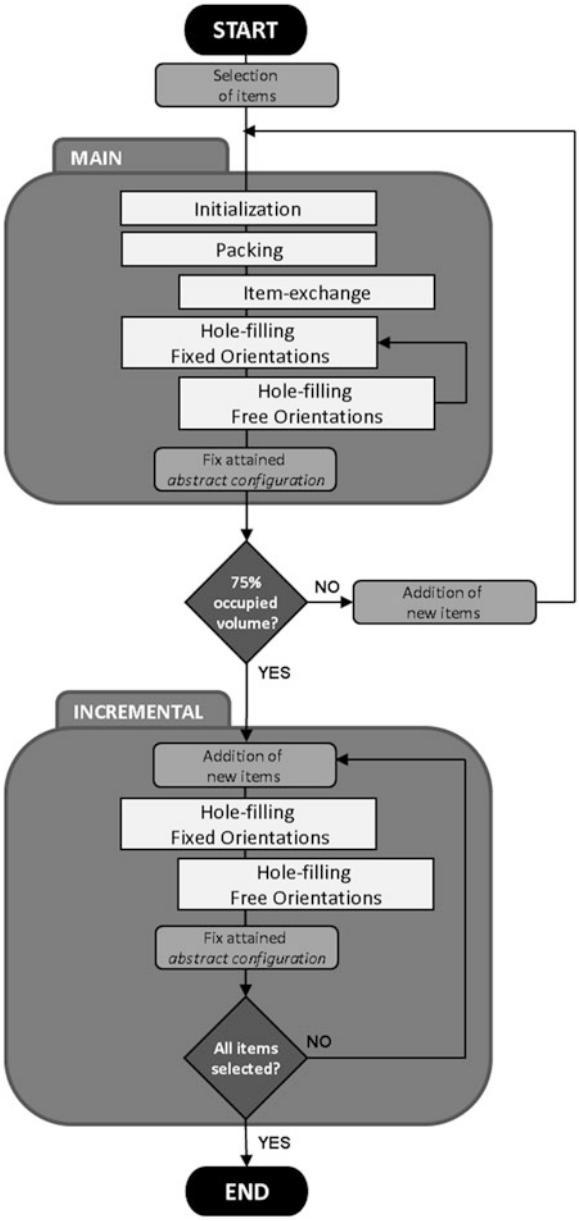

Fig. 7.1 Heuristics overall logic

high loading percentages have already been attained. At each module execution, the current *abstract configuration* is taken as input and an upgraded one is provided by it (if executed successfully).

7.2.1.1 *Main* Phase

A single cycle of the *main* phase consists of the prefixed sequence: *Initialization*, *Packing*, *Item-exchange*, and *Hole-filling*.

Their specific functionalities are summarized here below:

- the *Initialization* module generates a preliminary *abstract configuration*, for a subset of the items available, by means of a *relaxation* of the general MILP model (allowing item overlapping);
- the *Packing* module places items into the domain, in compliance with the current *abstract configuration* and maximizing the total volume loaded;
- the *Item-exchange* module attempts advantageous exchanges between (subsets of) non-loaded and loaded items (both are set free, with respect to the current *abstract configuration*, and a new one is correspondently generated, if the module execution has been successful);
- the *Hole-filling* module tries to add some of the unloaded items (that are set free, with respect to the current *abstract configuration*, and an upgraded one is correspondently generated, if the module execution has been successful). The *Hole-filling* module is first executed by fixing the orientation of all items involved. Afterwards, it is re-executed, if opportune, setting the item orientations free.

The *main* phase is carried on, by repeating single cycles, until either the loaded volume has attained 75 % of the domain's or a maximum time limit has been reached. The *abstract configuration* obtained at the end of this phase is handed over to the next.

7.2.1.2 *Incremental* Phase

A single cycle of the *incremental* phase consists of the prefixed sequence: *Hole-filling* and *Item-exchange* (at this level of volume exploitation, indeed, the first two modules are no longer effective, in particular *Initialization*, being based on a *relaxation* of the MILP model). Also in this case, the *Hole-filling* module is firstly executed by fixing the orientation of all items involved. Afterwards, if re-executed, these are set free. Here, the *Item-exchange* module has the role of performing backward iterations (to make up for possible previous inappropriate moves). The relevant functionalities of both modules employed are the same as described above. A number of single cycles are executed, until either all items have been processed or a maximum time limit has been reached.

7.2.2 Model Solving by CPLEX

The heuristics introduced in Sect. 7.2.1, and all of the tests reported in this chapter, were performed utilizing IBM ILOG CPLEX (see [3]) as the MILP optimizer. CPLEX carries out the optimization process by a branch & cut (B&C) algorithm, including several general purpose heuristics. It is also able to perform parallel optimization. Like most of the optimizers available to date, CPLEX has a default strategy for the MILP solution, which is flexible and adaptable to the model characteristics. Its level of sophistication is so advanced that a number of ad hoc optimizer parameters, able to outperform the default mode, can hardly be found. Moreover, the risk of "over engineering" the setting of the parameters, tuning them to a particular class of instances, rather than to the model intrinsic characteristics, cannot be neglected. Sometimes however, it can be useful to define a specific CPLEX optimization strategy. This holds, in particular, when the solution search is somehow time-boxed, and the proof of optimality is not necessary. This is the case of the two situations dealt with in Sect. 7.3, concerning either the solution of the MILP model directly or the execution of the heuristics of Sect. 7.2.1.

7.2.2.1 Direct Solution

When some difficult instances, albeit with a limited number of items, are tackled by solving the MILP model directly, i.e. in the first situation, the number of nodes generated by the B&C procedure really tends to "explode." In this circumstance, a specific strategy is needed, in order to reduce the node generation as much as possible and make the process spend more time at each B&B step, yielding (supposedly) better search choices. Another characteristic associated with the MILP model in question is that the LP-*relaxation* upper bound is usually coincident with the value of the optimal (integer) MILP solution. As a consequence, any strategy, aimed at generating cuts and improving the upper bound, results in being ineffective.

To cope with these difficult instances, an ad hoc approach was therefore devised. It is based on an intense employment of (CPLEX) heuristics, probing techniques (see [3]), a very limited use of cutting planes, and the "solution polishing" heuristics (see [3, 5]) that are activated when several solutions and at least 200 nodes have already been explored.

The priority order of the branching variables represents a further important feature of the approach studied. Since, from the model formulation and from the solution logic, some binary variables are supposedly able to induce a better separation in the search tree, they are provided with a higher priority in the process. Figure 7.2 illustrates the multiple CPLEX parameters (see [3]) that have been selected (the parameters not shown in the figure correspond to the default setting).

```
CPLEX Parameter File Version 12.6.0.1
#CPLEX Tuning for Direct Solution
CPX_PARAM_PRELINEAR 0
CPX_PARAM_MIPCBREDLP 0
CPX_PARAM_NODEFILEIND 3
CPX_PARAM_TILIM 3600
CPX_PARAM_POLISHAFTEREPAGAP 0.02
CPX_PARAM_POLISHAFTERNODE 200
CPX_PARAM_POLISHAFTERTIME 2400
CPX_PARAM_MIPEMPHASIS 1
CPX_PARAM_FLOWCOVERS 1
CPX_PARAM_MIRCUTS 1
CPX_PARAM_PRESLVND 3
CPX_PARAM_PROBE 3
CPX_PARAM_REPEATPRESOLVE 1
CPX_PARAM_RINSHEUR 5
CPX_PARAM_LBHEUR 1
CPX_PARAM_FRACCUTS -1
CPX_PARAM_LANDPCUTS 1
CPX_PARAM_SYMMETRY -1
```

Fig. 7.2 CPLEX parameter selection for the direct solution

```
CPLEX Parameter File Version 12.6.0.1
#CPLEX Tuning for Initialization
CPX_PARAM_TILIM          40
CPX_PARAM_PRELINEAR      0
CPX_PARAM_BRDIR          1
CPX_PARAM_POLISHAFTEREPGAP 0.1
```

```
CPLEX Parameter File Version 12.6.0.1
#CPLEX Tuning for
#Packing, Item-exchange and Hole-filling
CPX_PARAM_PRELINEAR 0
CPX_PARAM_MIPCBREDLP 0
CPX_PARAM_BRDIR 1
CPX_PARAM_OBJDIF 0.0001
```

Fig. 7.3 CPLEX parameter selection for the heuristic solution

7.2.2.2 Heuristic Solution

When the heuristics of Sect. 7.2.1 is utilized, i.e. in the second situation, it is of paramount importance to obtain quick, albeit sub-optimal, solutions for each module execution. Proof of optimality is not needed at all, although sometimes the heuristics performances can be biased by too many run interruptions (based on predefined-maximum-time limits). The priority order of the branching variable declaration is the same as in the direct solution situation. Figure 7.3 reports the CPLEX parameters adopted for *Initialization*, *Packing*, *Item-exchange*, and *Hole-filling*, respectively.

7.3 Experimental Analysis

The experimental analysis of this section is an extension of the previous, reported in Fasano [6]. The test campaign referred to hereinafter was performed using IBM CPLEX 12.6.0.1 (see [3]) as the optimizing engine, and IBM EasyModeler as the model generator. More precisely, the MILP solver available within CPLEX, statically linked to the C++ code generated by EasyModeler, was adopted, using the open source Coin-OR OSI 0.105.3 library as the interface between EasyModeler and the optimizer. The following computational supports were moreover utilized:

- platform: Lenovo Thinkpad W520 Laptop. with an Intel(R) Core (TM) i7-2620M at 2.7 GHz clock frequency (2 real core seen as 4 with Intel Hyperthreading) and 8 GB Ram available;
- operating system: Windows(R) 7 Professional OS.

All the tests were run using a parallel version of CPLEX. CPLEX 12.6 can execute the B&C in two different parallel flavors: Parallel Optimization on threads on the same (multi core) CPU, and Distributed Parallel with a messaging protocol among distinct CPUs. During the tests, the Parallel Optimization on Threads was employed; the number of Threads is defaulted to the number of cores seen by the OS, i.e. 4.

This section reports first a group of tests concerning the solution of the MILP model directly. Experimental results relevant to the use of the heuristics outlined in Sect. 7.2.1 are presented next. Additionally, instances with the balancing requirement are provided. All the case studies considered hereinafter involve box-shaped items and domains.

7.3.1 Direct Solution of Standard Instances

In order to test the MILP model for solving the container loading problem directly, we selected 5 fabricated instances (see [7]), whose optimal solutions were known a priori. Among them, 4 have a cube as a domain of 8, 9, 10, and 11 units, respectively. They are denoted in the following as: Cube-8, Cube-9, Cube-10, and Cube-11 tests. The domain of the further instance is a rectangular parallelepiped, obtained by merging two Cube-8 domains. The relevant test is referred to as: Double-cube-8. For all the tests considered in this section, no additional condition was posed. The instance data concerning the items available are reported, test by test, in the following Tables 7.1, 7.2, 7.3, 7.4, and 7.5.

A maximum time limit of 1 h was set for each test case. Two out of five (i.e., Cube-8 and Cube-9) were solved to optimality; in two cases (i.e., Cube-10 and Cube-11) only one item was rejected; in one (i.e., Double-cube-8) those not loaded were three. The relevant results are shown in Table 7.6 while Figs. 7.4 and 7.5 provide graphical views of the solutions obtained for Cube-8 and Double-cube-8.

Table 7.1 Cube-8 items

Item type	L1 side (units)	L2 side (units)	L3 side (units)	No. of items per type
A	4	4	4	1
B	2	3	5	6
C	1	3	6	6
D	1	2	6	6
E	1	3	3	6
F	1	2	2	6

Table 7.2 Cube-9 items

Item type	L1 side (units)	L2 side (units)	L3 side (units)	No. of items per type
A	5	5	5	1
B	2	4	6	6
C	1	3	7	6
D	1	2	7	6
E	1	3	4	6
F	1	2	2	6

Table 7.3 Cube-10 items

Item type	L1 side (units)	L2 side (units)	L3 side (units)	No. of items per type
A	6	6	6	1
B	2	5	7	6
C	1	3	8	6
D	1	2	8	6
E	1	3	5	6
F	1	2	2	6

Table 7.4 Cube-11 items

Item type	L1 side (units)	L2 side (units)	L3 side (units)	No. of items per type
A	7	7	7	1
B	2	6	8	6
C	1	3	9	6
D	1	2	9	6
E	1	3	6	6
F	1	2	2	6

Instances of the above tests, with all items pre-oriented (correspondently to the fabricated optimal solutions) were considered. Surprisingly enough, none of them was solved to optimality within 1 h. Further pre-oriented instances for Cube-8 and Cube-9 were hence taken into account, additionally. In such cases, the item pre-orientation was derived from the solutions reported in Table 7.6. The optimal solutions (or some equivalent) were re-obtained in almost half the time of the

Table 7.5 Double-cube-8 items

Item type	L1 side (units)	L2 side (units)	L3 side (units)	No. of items per type
A	4	4	4	2
B	2	3	5	12
C	1	3	6	12
D	1	2	6	12
E	1	3	3	12
F	1	2	2	12

Table 7.6 Direct solution tests

Test case	Max no. of items	No. of loaded items	Load factor (%)	Elapsed time (s)	Loaded items (%)	No. of nodes	Optimality proved
Cube-8	31	31	98.05	312	100.00	231	Yes
Cube-9	31	31	98.63	2,001	100.00	315	Yes
Cube-10	31	30	96.60	3,600	96.77	403	No
Cube-11	31	30	96.54	3,600	96.77	463	No
Double-cube-8	62	59	89.26	3,600	95.16	270	No

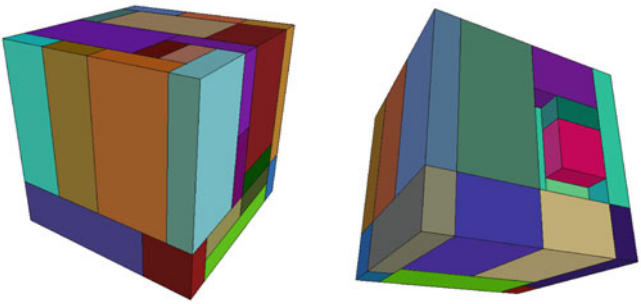

Fig. 7.4 Cube-8 graphical results

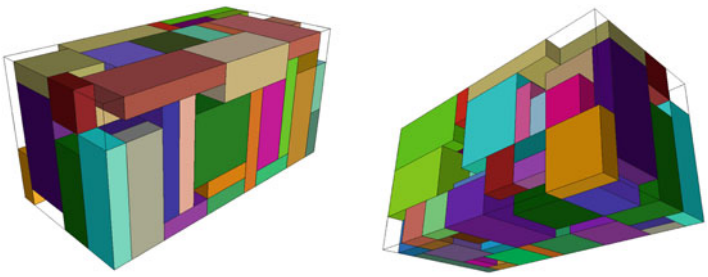

Fig. 7.5 Double-cube-8 graphical results

non-pre-oriented instances. Our interpretation in regard is that, since several *non-symmetric* solutions exist, the more "degrees of freedom" there are, the more effective the CPLEX heuristics (including the "solution polishing") results.

7.3.2 Heuristic Solution of Standard Instances

This section refers to 27 non-trivial test cases for the classical container loading problem, with no additional conditions. They are extracted from the reference: "Three Dimensional Cutting and Packing Data Sets - THPACK 1-7 BR" [8]: http://www.euro-online.org/web/ewg/25/esicup-euro-special-interest-group-on-cutting-and-packing. As is known, this test-bed consists of 7 sets of 100 test cases each. Among these, all those with an available number of items between 200 and 400 were selected. They are listed in Table 7.7 and, hereinafter, numbered sequentially, from 1 to 27.

The 27 selected test cases were solved using the heuristics of Sect. 7.2.1, with a maximum time limit of 1 h. The relevant results are summarized in Table 7.8 (the reported time elapses are often longer than 1 h, since the heuristics always finalized the last optimization steps, prior to performing the final housekeeping) (Fig. 7.6).

Load factors range from 57.45 to 87.07 %, with an average of 77.92 % and a standard deviation of 7.52 (graphical results relative to Test case 17 are illustrated in Fig. 7.7). It is interesting to note that the time spent in optimization is inversely correlated, as pointed out in Fig. 7.6. This appears as an indication that the heuristics' logic itself is more relevant than the optimizer speed. The instances are solved to a greater extent when the heuristics is able to generate easier sub-instances to solve. The heuristics' overall logic and its specific module features (including possible function extensions) are expected to represent the objective of further research.

Table 7.7 Selected test cases (THPACK 1-7 BR)

Set number	Test case
1	13,17,33,39,67,68,76,85,91,100
2	4,13,39,59,77,79,85,96
3	39,56,59,77
4	39,56,79
5	56
6	13

Table 7.8 Results for the selected test cases (THPACK 1-7 BR)

Test case	Max no. of items	No. of loaded items	Load factor (%)	Elapsed time	Loaded items (%)	Time spent in optimization	Time spent in optimization (%)	No. of submodels solved
1	284	186	84.60	01:01:28	65.5	00:19:05	31.0	86
2	213	155	84.96	01:01:59	72.8	00:35:31	57.3	71
3	282	159	76.57	01:01:08	56.4	00:38:08	62.4	52
4	243	163	85.57	01:00:49	67.1	00:27:32	45.3	78
5	221	140	72.44	01:04:49	63.3	00:47:48	73.7	50
6	238	119	57.45	01:00:20	50.0	00:52:24	86.9	37
7	269	149	59.30	01:02:21	55.4	00:49:28	79.3	42
8	319	165	67.39	01:01:25	51.7	00:46:29	75.7	40
9	238	149	86.82	00:54:08	62.6	00:28:41	53.0	93
10	214	151	79.18	00:55:35	70.6	00:35:49	64.4	64
11	201	143	87.07	00:53:25	71.1	00:31:57	59.8	83
12	228	174	84.87	01:01:29	76.3	00:27:27	44.6	84
13	266	168	78.34	01:01:13	63.2	00:36:58	60.4	64
14	201	129	69.62	00:52:51	64.2	00:43:21	82.0	39
15	202	144	77.08	00:58:37	71.3	00:43:38	74.4	52
16	206	145	79.46	01:00:37	70.4	00:40:17	66.5	74
17	209	163	85.32	01:01:20	78.0	00:36:30	59.5	74
18	202	139	82.01	01:03:20	68.8	00:31:56	50.4	72
19	232	200	77.11	01:01:35	86.2	00:31:07	50.5	69
20	212	157	82.89	01:00:58	74.1	00:33:58	55.7	84
21	216	145	74.05	01:00:38	67.1	00:45:52	75.6	50
22	201	135	76.88	01:01:22	67.2	00:44:10	72.0	57
23	225	138	77.60	01:01:02	61.3	00:40:57	67.1	66
24	233	151	80.47	01:01:46	64.8	00:39:40	64.2	56
25	217	138	80.42	01:01:32	63.6	00:39:41	64.5	72
26	218	131	78.11	01:00:37	60.1	00:42:30	70.1	63
27	203	119	78.15	01:00:40	58.6	00:44:54	74.0	64

7.3.3 Heuristic Solution of Test Cases with Balancing Conditions

An extension of the classical loading problem is briefly discussed here. The (quite frequent in practice) balancing requirement, for which the overall center of mass (of the loaded container) must stay inside a convex domain is considered (see [6]). Each item (supposed to be of homogeneous density) is therefore represented by its side lengths and mass. In the following sections (for the sake of simplicity) no mass is associated with the container itself and the overall center of mass domain is assumed to be a (rectangular) parallelepiped.

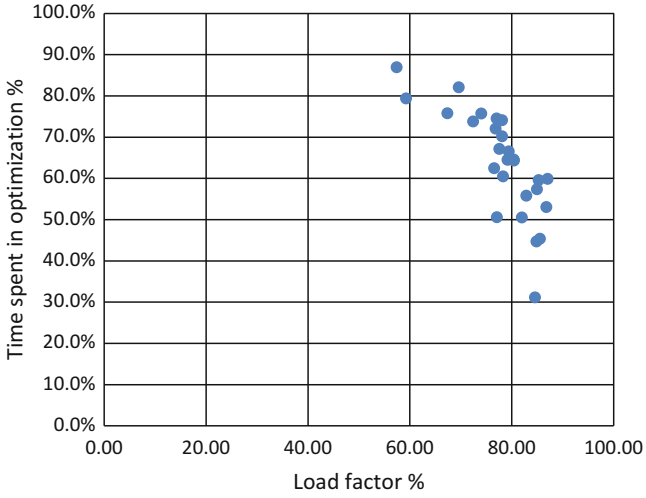

Fig. 7.6 Correlation between load factor and time spent in optimization

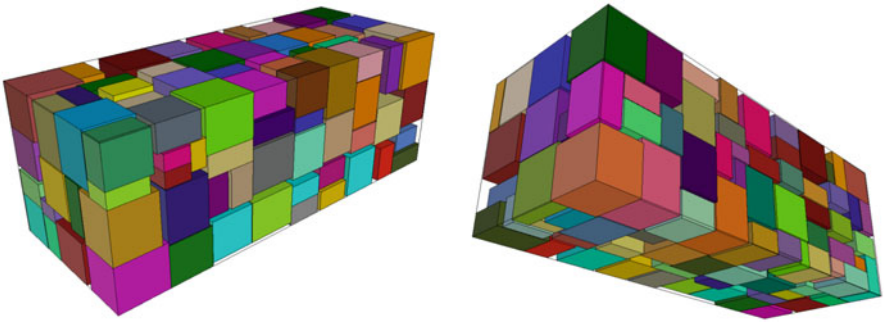

Fig. 7.7 Test case 17 graphical results

7.3.3.1 Items Having the Same Density

The 27 test cases of Sect. 7.3.2 were reconsidered assuming that all items had the same density. These test cases were run with a very tight restriction on the center of mass domain, which consisted of a cube of 2 units, centered with respect to the container. Since the container was a box of $587 \times 233 \times 220$ units, this meant a deviation from its center well below 0.5 % its side lengths. The relevant results are reported in Table 7.9.

Both from the load factor and computational performance viewpoints, the results are quite similar to the test cases reported in Sect. 7.3.2. A slight deviation from them can be noticed, consisting, essentially, of an average load factor decrement of a mere 1.14 %.

Table 7.9 Results for the selected test cases with items of constant density

Test case	Max no. of items	No. of loaded items	Load factor (%)	Elapsed time	Loaded items (%)
1	284	173	82.57	01:00:40	60.9
2	213	151	83.80	00:46:07	70.9
3	282	168	75.21	01:01:15	59.6
4	243	152	83.84	00:58:54	62.6
5	221	144	74.16	00:40:51	65.2
6	238	120	57.87	01:00:58	50.4
7	269	155	61.18	01:00:56	57.6
8	319	141	57.60	01:03:08	44.2
9	238	151	86.31	00:59:23	63.4
10	214	147	76.67	01:00:42	68.7
11	201	142	85.04	00:50:41	70.6
12	228	175	84.69	01:00:56	76.8
13	266	167	78.66	01:01:10	62.8
14	201	126	68.29	01:00:07	62.7
15	202	148	78.99	00:58:47	73.3
16	206	139	75.96	01:00:37	67.5
17	209	159	84.22	00:45:55	76.1
18	202	131	79.41	00:56:22	64.9
19	232	196	76.49	01:01:57	84.5
20	212	162	83.29	01:01:14	76.4
21	216	141	73.42	01:05:13	65.3
22	201	137	77.02	01:02:57	68.2
23	225	142	78.83	01:00:53	63.1
24	233	155	80.94	01:04:15	66.5
25	217	109	72.35	01:00:54	50.2
26	218	128	77.27	01:00:41	58.7
27	203	121	78.92	01:02:15	59.6

7.3.3.2 Items Having Different Densities

Three test cases, i.e. 4, 19 and 25, extracted from the set of 27 of Sect. 7.3.2 were considered, by providing the items with different densities (generated randomly). These are reported in Table 7.10 (referring to mass and volume units). For these three test cases the overall center of mass was requested to stay inside a slightly larger domain (roughly representing 5 % of tolerance over the length of each axis), centered with respect to the container. Table 7.11 shows the relevant results and Fig. 7.8 provides graphical views of the solution obtained for Test case 4.

Finally Test cases 4, 19 and 25 were considered, maintaining the same masses reported in Table 7.10, but with different conditions concerning the position of the center of mass domain (that continued to have the same dimension as before). This was placed in an off-centered position, inside the container. This request can occur

Table 7.10 Items with different densities

Test case	Average density	Standard deviation
4	1.5276	0.2373
19	1.5211	0.2269
25	1.5134	0.2317

Table 7.11 Results for Test cases 4, 19, and 25 with central balancing

Test case	Max no. of items	No. of loaded items	Load factor (%)	Elapsed time	Loaded items (%)
4	243	158	84.53	01:02:15	65.0
19	232	137	77.32	01:01:51	59.1
25	217	123	77.03	01:00:31	56.7

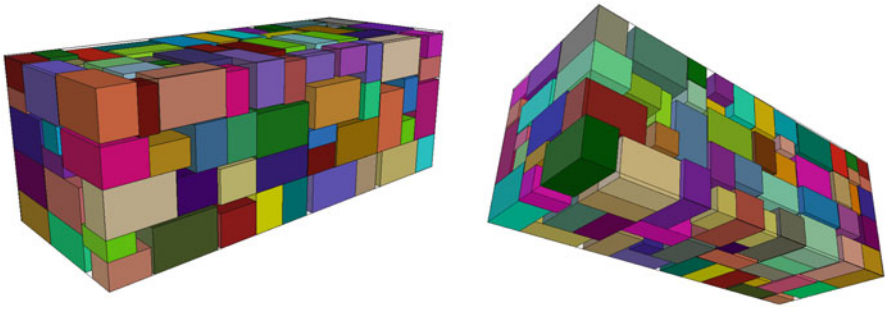

Fig. 7.8 Test case 4 (with balancing conditions) graphical results

Table 7.12 Center of mass domain off-centered locations

Domain dimensions			Center of mass coordinates		
x	y	z	x	y	z
587	233	220	293.5	116.5	55

Table 7.13 Results for Test cases 4, 19, and 25 with off-centered balancing

Test case	Max no. of items	No. of loaded items	Load factor (%)	Elapsed time	Loaded items (%)
4	243	87	51.31	01:06:35	34.4
19	232	73	47.36	01:03:10	31.5
25	217	46	36.77	01:01:54	21.2

in practice, for instance, for structural reasons (see [9]). Table 7.12 reports, for Test cases 4, 19, and 25 respectively, the positions of the relevant domain centers. The results obtained are shown in Table 7.13. Test case 4 solution is represented graphically in Fig. 7.9.

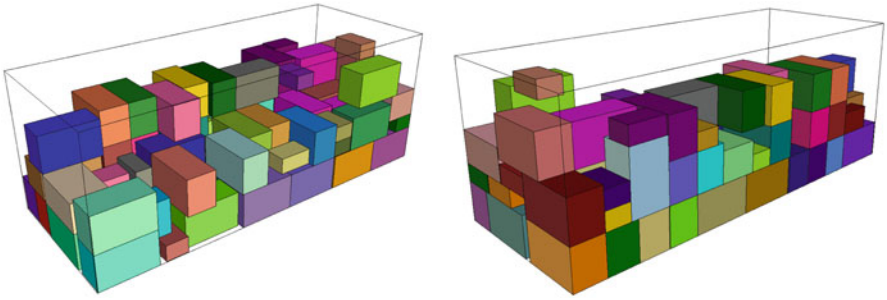

Fig. 7.9 Test case 4 (with off-centered balancing conditions) graphical results

7.4 Conclusive Remarks

This work focuses on experimental aspects relevant to the container loading problem, solved by a modeling-based heuristic approach. The relevant MILP model is discussed in depth in another chapter of this book and represents the reference framework for the underlying mathematical formulation. This approach is aimed at coping with complex non-standard packing problems, involving *tetris*-like items, non-box-shaped domains and additional conditions, such as balancing.

A standard context has, nonetheless, been targeted in this chapter, addressing the container loading problem, as per its classical statement. This concerns the placement of box-shaped items (with the possibility of rotation) into a box-shaped domain, with no additional conditions, maximizing the loaded volume.

The general MILP model has been tested to solve directly non-large-scale fabricated instances, whose optimal solutions were known a priori. Afterwards, a set of complex case studies have been studied and further examples involving the additional condition of balancing provided.

Although for the specific experimental context considered, the proposed approach is usually outperformed by most of the off-the-shelf container loading optimization methods, the authors deem that the results shown here provide a useful reference in a standard-based framework. Further research relevant both to the heuristics overall logics and its specific features, referred to a more general non-standard context, is in the pipeline for the near future.

Acknowledgements We wish to thank Janos D. Pintér for his accurate review of the manuscript. We are also grateful to Jane Evans for her support. Alessandro Castellazzo acknowledges the grant provided by Lagrange Project – Crt Foundation, Turin, Italy.

References

1. Bortfeldt, A., Wäscher, G.: Container Loading Problems – A State-of-the-Art Review. FEMM Working Papers 120007, Otto-von-Guericke University Magdeburg, Faculty of Economics and Management (2012)
2. Fasano, G.: A modeling-based approach for non-standard packing problems. In: Fasano, G., Pintér, J.D. (eds.) Optimized Packings and Their Applications. Springer Optimization and Its Applications. Springer Science + Business Media, New York (2015)
3. IBM: CPLEX 12.6.0 User Manual. http://www-01.ibm.com/support/knowledgecenter/SSSA5P_12.6.0/ilog.odms.studio.help/Optimization_Studio/topics/COS_home.html?lang=en (2013)
4. Pisinger, D.: Heuristics for the container loading problem. Eur. J. Oper. Res. **141**(2), 382–392 (2002)
5. Rothberg, E.: An evolutionary algorithm for polishing mixed integer programming solutions. INFORMS J. Comput. **19**(4), 534–541 (2007)
6. Fasano, G.: Solving Non-standard Packing Problems by Global Optimization and Heuristics. SpringerBriefs in Optimization, Springer Science + Business Media, New York (2014)
7. Rietz, J.: Special Difficulties in the Three-Dimensional Container Loading Problem. Universidade do Minho, Braga (2008)
8. Bischoff, E.E., Ratcliff, M.S.W.: Issues in the development of approaches to container loading. OMEGA **23**(4), 377–390 (1995)
9. Fasano, G., Lavopa, C., Negri, D., Vola, M.C.: CAST: a successful project in support of the international space station logistics. In: Fasano, G., Pintér, J.D. (eds.) Optimized Packings and Their Applications. Springer Optimization and Its Applications. Springer Science + Business Media, New York (2015)