A GRASP Approach to the Container-Loading Problem

Ana Moura, *Escola Superior de Tecnologia e Gestão, Bragança, and Instituto de Engenharia de Sistemas e Computadores do Porto*

José Fernando Oliveira, *Faculdade de Engenharia da Universidade do Porto and Instituto de Engenharia de Sistemas e Computadores do Porto*

GRMODGRASP is a new algorithm for solving the container-loading problem. Based on a wall-building, constructive heuristic, it can achieve high levels of cargo stability without compromising the container's volume use.

The *container-loading problem* aims to determine the arrangement of items in a container. Researchers approach this 3D, NP-hard problem[1] using heuristic methods. Usually, the CLP aims to maximize loading efficiency—that is, the container space use. Here, the problem we address involves only one container with known dimensions, and the cargo varies from weakly to strongly heterogeneous, independent of the total number of boxes. We consider three requirements related to the load's physical arrangement and to the transportation requirements: box orientation (for example, "this side up"), cargo stability, and container volume. Although considering both volume use and cargo stability could lead to a bi-objective CLP, we tackle cargo stability as a constraint in the constructive phase of the algorithm, and we explicitly consider volume use as the only objective in the constructive and local-search phases.

In this article, we present GRMODGRASP, a new algorithm for the CLP based on the GRASP (greedy randomized adaptive search procedure) paradigm.[2] We evaluate GRMODGRASP's performance in terms of volume use and load stability and by comparing it with nine well-known algorithms. Our approach produces solutions that surpass other approaches' solutions in terms of volume use and cargo stability.

The Modified George and Robinson heuristic

We based GRMODGRASP on GRMOD, an improved version of the George and Robinson heuristic.[3] This wall-building heuristic packs boxes in a container, with an opening in the front, from the back to the front along its length.

One modification to the George and Robinson heuristic relates to the container length.[4] The original heuristic considers an infinite-length container (the 3D strip-packing problem), but it doesn't guarantee that the resulting packing will have a length equal to or less than the container's length; GRMOD deals with a finite-length container. With this modification, we can eliminate the George and Robinson algorithm's "unsuccessful packing" and "automatic repacking" procedures, which basically compare the final packing length with the container's length and reapply the heuristic with different parameters. Successively executing the George and Robinson heuristic might obtain a feasible solution if the cargo's total volume is equal to or less than the container's volume.

Another modification addresses packing the container's final layers.[4] The George and Robinson heuristic uses a minimal-length parameter that inhibits constructing new layers at the end of the packing process. This causes layers with low volume use. In GRMOD (see figure 1), the layer depth dimension depends on the unpacked boxes' volume. So, the container's final layers have a smaller depth but better volume use. GRMOD incorporates the two modifications just described plus two improvements that we introduced to improve cargo stability. The first deals with new-space generation, and the other relates to the flexible-width value.

Constructive heuristic

Like the George and Robinson heuristic, the GRMOD constructive heuristic builds on the concept of *empty space*—a parallelepipedic region without a box packed inside. GRMOD deals with empty spaces in two different ways. When the empty space's height and width equals the container's height and width,

```
FreeSpacesList = Container
repeat
    Pick a space from FreeSpacesList
    If the space is a container front space
        Start a new layer
        NewSpaces = Generate new spaces
        FreeSpacesList = FreeSpacesList ∪
            NewSpaces
        Amalgamate rejected spaces
    Else
        Fill free space
        NewSpaces = Generate new spaces
        FreeSpacesList = FreeSpacesList ∪
            NewSpaces
        Amalgamate rejected spaces
    If no box fits inside the space then mark it
        as rejected
Until there are no more boxes to pack or spaces to fill
```

Figure 1. The GRMod heuristic.

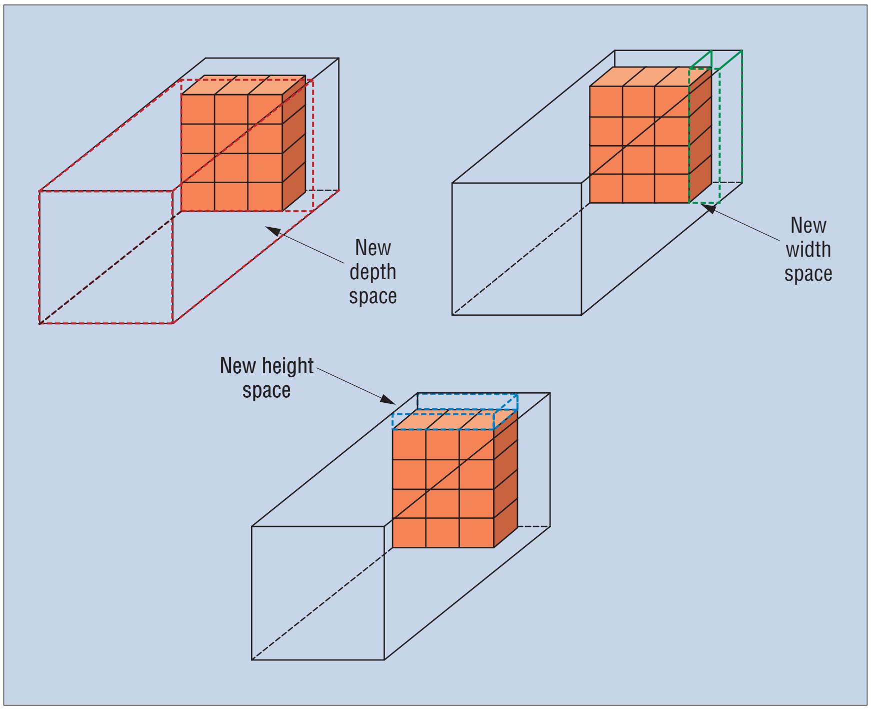

Figure 2. New-space generation.

the heuristic treats this space as a new layer. In this case, the layer's depth dimension is defined by the depth dimension of the type of box chosen to start the layer. Otherwise, GRMod considers the space a free space.

When the heuristic starts a new layer, it places boxes in vertical columns along the container's width. In other cases—that is, empty spaces where the height and width differ from the container's height and width—GRMod tries to pack the boxes in spaces not occupied by boxes in the current and previous layers.

When none of the unpacked boxes fit a given free space, that space is temporarily marked as "rejected." However, if a new adjacent layer is built, GRMod can amalgamate this space with another to make it useful.

Building a layer

Because GRMod is based on a wall-building procedure, it fills the container with transversal walls and the first box placed in a layer determines the container's depth. In the George and Robinson heuristic, a layer's depth depends on a K parameter, which limits the new layer's depth and ranks and chooses the box that starts the new layer. To eliminate this dependency on box-ranking schemes, GRMod generates and evaluates all box types and box-orientation combinations and selects the best one to open a new layer.

Starting a new layer. When starting a new layer, the first consideration is which box to use to open the layer. Once GRMod chooses a box type and orientation, it sets the layer depth equal to the box's dimension placed along that direction. The container's dimensions and that box type's availability limit the height and number of boxes placed along the width. The algorithm fills the height as much as possible with an integer number of boxes and then replicates these columns along the width. Incomplete columns are allowed.

New-space generation. New-space generation (see figure 2) follows a fixed order. The first space GRMod creates is the depth space that corresponds to the free space in front. This space is always created unless the algorithm has already reached the front. The next spaces generated are the width and then the height. If the arrangement of boxes perfectly fits the container along one of these dimensions, these new spaces will have null dimensions—that is, they won't exist. When the George and Robinson heuristic creates a width space, if one dimension is smaller than the minimum unpacked box dimension, it doesn't insert a new width space in the space list. In this case, the height space assumes the original space's width (see figure 3a). This usually results in no fully supported boxes—that is, box bases that aren't fully in contact with others boxes or the container's floor. To improve cargo stability, GRMod marks as

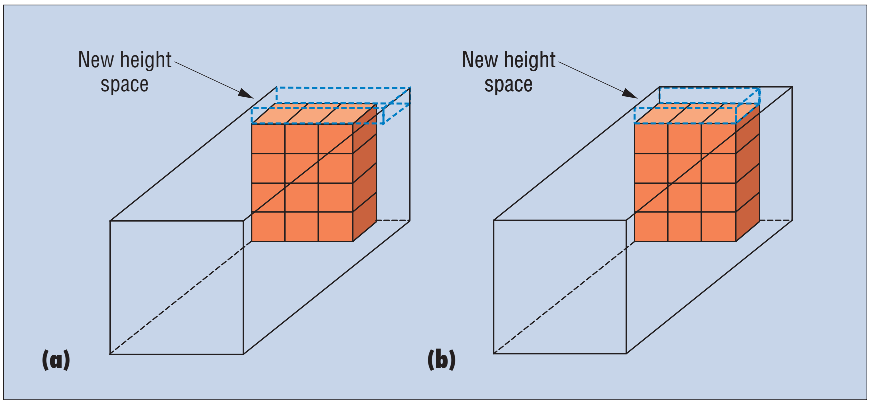

Figure 3. Differences in space creation: (a) the George and Robinson and (b) GRMod heuristics.

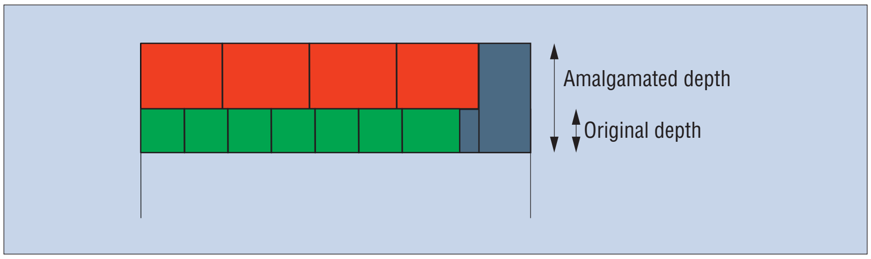

Figure 4. Amalgamation procedure.

rejected a new space in which one of the dimensions is smaller then the smallest dimension of the boxes not yet packed and doesn't increase the height space's size (see figure 3b). This guarantees full support for all boxes and increases cargo stability.

GRMod inserts all spaces in a list in the order in which they were generated. Later, during packing, the algorithm uses free spaces following a first-in, last-out strategy.

Amalgamation. When GRMod marks a new space as rejected, it tries to increase the space's size by amalgamating the space with contiguous spaces belonging to the previous layer that have also been marked rejected (in blue) to generate a new useful space (see figure 4). If the algorithm can't amalgamate any spaces of the previous layer, the rejected space stays on the list with the hopes that it will be amalgamated with a new rejected space in the next layer.

This process favors more efficient, dense packing. Consequently, the algorithm can pack boxes with a depth dimension larger than the current layer's depth, which generates interlocking walls. Interlocking is a common strategy for increasing cargo stability that human operators also use when loading containers. However, this process makes any kind of search process based on the interchange of layers much harder and leads to important changes among neighbor solutions. Despite these disadvantages, we decided to use space amalgamation in this implementation, but we based our search process on box interchanges instead of layer interchanges.

Flexible width. GRMod can reject very small spaces that haven't been amalgamated with contiguous spaces and that can't be used to pack any box. To avoid this space fragmentation, the original George and Robinson heuristic proposed the concept of flexible width. The heuristic uses this parameter to bind the number of columns that it can place along the width in a new layer. Its value propagates from the previous layer and equals the width of the boxes that started the previous layer. For instance, if layer n started with a column of boxes with a width of 30 cm and four columns were placed along the container's width, the flexible width for layer $n + 1$ would be 120 cm (see figure 5a).

The George and Robinson heuristic would place an additional column in the new layer. However, in GRMod, the largest integer smaller than the flexible width binds the number of columns in the new layer (see figure 5b). This new way of determining a new layer's number of columns leads to a larger empty space on the right side (see figure 4), increasing its later probability of use. Moreover, with this new strategy, the new layer's last column layer is always fully in contact with the previous layer, increasing cargo stability.

Filling a free space

A layer's construction ends by filling the free spaces generated in the first phase of this layer's construction. The algorithm first fills the height space, considering only boxes that have smaller dimensions than the space dimensions. For each box type, it computes all possible arrangements (number of columns for depth and width and number of boxes per column, considering all feasible box orientations) and selects the one that yields the best volume use. If more than one arrangement leads to the best volume use, it randomly chooses one. The algorithm fills the free space with the chosen box type and box orientation. If it can't find a feasible arrangement of boxes, it marks the space as rejected and tries to amalgamate this space with any other space previously marked as rejected.

After filling a space's new depth, the algorithm generates width and height spaces and inserts them in the space list. It applies this space-filling procedure recursively until no more free spaces, other than the container front space, are available. Then, it repeats the new layer procedure, applying it to the container front space.

A Grasp approach

GRModGrasp has two steps. The algorithm builds a solution, and then it improves the solution with a local-search algorithm. In the construction phase, it loads the container until it meets one of the following three conditions: the container has no more free space, there are no more boxes to be packed, or the dimensions of the remaining free spaces are smaller than the dimensions of the available boxes for packing.

Afterwards, GRModGrasp runs a local-search phase to improve this solution. GRModGrasp uses GRMod as the basis for the Grasp algorithm's constructive phase. However, following the Grasp strategy, this constructive heuristic is randomized. After choosing an empty space, the algorithm chooses the next type of box to pack from a candidate list that contains the several box types available, ordered by the volume use that that box type

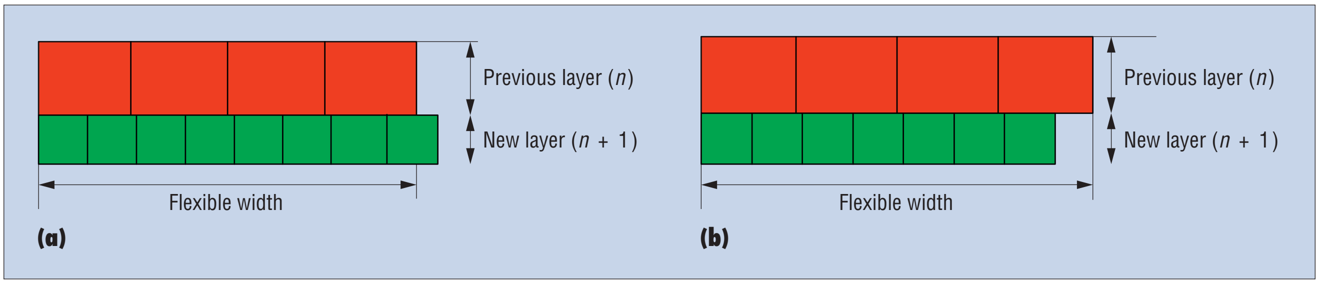

Figure 5. Flexible width for the new layer: (a) the George and Robinson and (b) GRMod heuristics.

Table 1. LN (Loh and Nee) test problem results.*

	H_B_al	H_BR	GA_GB	TS_BG	HGA_BG	H_E	PTS_B_al	GRModGrasp
Mean of volume use (%)	69.5	68.6	70.0	**70.9**	70.1	69.9	**70.9**	70.3
Best value	10	11	12	**15**	13	13	**15**	13
Global optima solution	10	11	12	13	13	13	13	13

*The best values appear in blue.

achieves in that concrete space. A totally greedy strategy would lead to the choice of the best box type (the first element of the candidate list), and a completely random strategy would draw from the entire list. The Grasp approach builds a *restricted candidate list*, containing the best candidates, and randomly chooses from this list. For the ranking criterion, we adopted volume use to measure the benefit of selecting each box type for a new layer or for filling free space.

To define which candidates will belong to the RCL, our new algorithm uses a parameter α, which will control the algorithm's greediness. This parameter can vary between 0 and 1. After computing the volume use for all candidates (box types), GRModGrasp fills the RCL according to the following threshold: $T = MVU + \alpha * (mVU - MVU)$, where T stands for the volume use threshold, MVU for the maximum volume use computed for all feasible box types, and mVU for the minimum volume use computed for all feasible box types. If the volume use for one box type is greater or equal to the T parameter, it's added to the RCL. It's easy to see that when $\alpha = 1$, T is the minimum and the basic heuristic is random; if $\alpha = 0$, T is the maximum and the basic heuristic is greedy.

In the Grasp local-search phase, the algorithm starts with the solution built in the construction phase. It defines, builds, and searches a neighborhood. If it finds a better solution, this new solution replaces the old and a new neighborhood is built around it. The algorithm uses a *first better* strategy when more than one better solution exists. The local-search procedure stops when it can't find a better solution in the neighborhood.

To build a neighborhood, we must modify the solutions. In this approach, sequences of boxes represent the solutions. These sequences are the order the boxes should be placed in the container. Then, GRModGrasp randomly selects a position in the sequence and removes all boxes from that position until the end of the sequence from the list of packed boxes and inserts them in the unpacked boxes list. The box type that corresponds to the random position becomes "forbidden" and is temporarily removed from both lists. Then, for all boxes belonging to the unpacked boxes list, the heuristic applies GRMod, but now without any randomness ($\alpha = 0$). After packing the first box type, GRMod reinserts the forbidden-box type in the unpacked-boxes list. This mechanism guarantees that the box type that previously occupied the modified position in the sequence will not retake that place. GRMod, in its greedy flavor, continues until it can't pack any more boxes and therefore obtains a new solution.

Computational experiments

We used test problems taken from the literature for benchmarking. Han Tong Loh and Andrew Nee generated 15 test problems, named LN problems.[5] Each test problem uses a different container's size. The container volume is large enough to pack all the items in 13 of the 15 test problems.

E.E. Bischoff and M.S.W. Ratcliff also presented 15 classes of test problems (called BR1 to BR15), each class with 100 problems.[6] With respect to the boxes' dimensions and assortment, the classes vary from weakly to strongly heterogeneous.

For each test problem, we ran 200 Grasp iterations (on a Pentium IV at 2.4 GHz with 480 Mbytes of RAM). We considered 10 different values of α, ranging from 0 to 1 with a step of 0.1, and for each α value, we ran the Grasp algorithm 20 times. The Grasp local search stops when it can't find a better neighbor.

For our benchmarking, we compared GRModGrasp's performance with nine approaches:

- H_B_al, a heuristic approach;[7]
- H_BR, a heuristic approach;[6]
- GA_GB, a genetic algorithm;[8]
- TS_BG, a tabu search approach;[9]
- HGA_BG, a hybrid genetic algorithm;[10]
- PGA_GB, a parallel genetic algorithm;[11]
- H_E, a heuristic approach;[12]
- PTS_B_al, a parallel tabu search algorithm;[1]
- H_B, a heuristic approach.[13]

Most of these approaches specifically deal with the CLP's loading arrangement's efficiency. Heuristic procedures such as H_BR, which build loading plans from a series of horizontal layers, are common. A different family of heuristics also builds packing arrangements iteratively but doesn't restrict configurations to walls or layers. Often a single box is placed in each iteration (as in H_E and H_B). More sophisticated approaches such as metaheuristics and tabu search and genetic algorithms attempt to solve the CLP, both in its weak and strongly heterogeneous versions. Parallel versions of these algorithms (such as PGA_GB and PTS_B_al) were developed to address the CLP.

Volume use is a container-loading algorithm's first performance evaluation criterion. Table 1 shows the results for the 15 LN test problems (an algorithm that achieves the best-known volume use for a problem is labeled a *best value* (in blue)). When an algorithm manages to pack all of a problem's boxes in the container, it's achieved a *global optima solution*. To the best of our knowledge, only seven of the nine algorithms compared here have published results for these problems (we didn't find any for PGA_GB or H_B). For the average value of volume use, TS_BG and PTS_B_al achieved the best results. Those approaches achieved 15 best values—two more than the other approaches. GRModGrasp outperforms the remaining five approaches by producing a higher mean value for volume use.

Table 2 presents the results for the 15 BR problem classes. Some approaches (for example, H_E, H_B, and PTS_B_al) report results for only weakly heterogeneous problems (from BR1 to BR7). The average volume use for all approaches decreases as cargo heterogeneity increases. By comparing the volume use for all BR problems, we can state that GRModGrasp offers a competitive approach.

To simplify the results comparison, we divided the 15 BR problem sets into two groups: the weakly heterogeneous (from BR1 to BR7) and the strongly heterogeneous (from BR8 to BR15):

- Within the weakly heterogeneous problem results, only two approaches always

Table 2. BR (Bischoff and Ratcliff) test problem results.*

Problem	Volume use									
	H_B_al	H_BR	GA_GB	TS_BG	HGA_BG	PGA_GB	PTS_B_al	H_B	H_E	GRMod Grasp
BR1	81.76	83.37	86.77	92.63	87.81	88.10	**93.52**	89.39	88.00	89.07
BR2	81.70	83.57	88.12	92.70	89.40	89.56	**93.77**	90.26	88.50	90.43
BR3	82.98	83.59	88.87	92.31	90.48	90.77	**93.58**	91.08	89.50	90.86
BR4	82.60	84.16	88.68	91.62	90.63	91.03	**93.05**	90.90	89.30	90.42
BR5	82.76	83.89	88.78	90.86	90.73	91.23	**92.34**	91.05	89.00	89.57
BR6	81.50	82.92	88.53	90.04	90.72	91.28	**91.72**	90.70	89.20	89.71
BR7	80.51	82.14	88.36	88.63	90.65	**91.04**	90.55	90.44	88.00	88.05
BR8	79.65	80.10	87.52	87.11	89.73	**90.26**	—	—	—	86.13
BR9	80.19	78.03	86.46	85.76	89.06	**89.50**	—	—	—	85.08
BR10	79.74	76.53	85.53	84.73	88.40	**88.73**	—	—	—	84.21
BR11	79.23	75.08	84.82	83.55	87.53	**87.87**	—	—	—	83.98
BR12	79.16	74.37	84.25	82.79	86.94	**87.18**	—	—	—	83.64
BR13	78.23	73.56	83.67	82.29	86.25	**86.70**	—	—	—	83.54
BR14	77.40	73.37	82.99	81.33	85.55	**85.81**	—	—	—	83.25
BR15	75.15	73.38	82.47	80.85	85.23	**85.48**	—	—	—	83.21
Mean	**80.17**	**79.20**	**86.39**	**87.15**	**88.61**	**88.97**	—	—	—	**86.74**

*The best values appear in blue.

achieved better results than GRModGrasp for all problem classes: PTS_B_al and TS_BG. However, for the strongly heterogeneous problems, GRModGrasp outperforms TS_BG for the last five BR problem classes, and there are no published results for PTS_B_al for strongly heterogeneous problems.

- For the first three weakly heterogeneous problem classes, H_B and GRModGrasp achieved very similar results. For other problem classes, H_B outperformed GRModGrasp.
- GRModGrasp outperforms H_E, H_B_al and H_BR for all problem classes.
- For the first three BR problem classes, PGA_GB and HGA_BG had worse volume use than GRModGrasp. With the increase of cargo heterogeneity, their results became better than GRModGrasp's results. In these kind of (strongly heterogeneous) problems, the genetic algorithms PGA_GB and HGA_BG always achieved better results.
- GRModGrasp outperforms GA_GB for the weakly heterogeneous problem classes and for the last two strongly heterogeneous problem classes.

So, when considering the results published in the literature, no algorithm outperforms GRModGrasp for all BR classes of problems, and for each problem class, there's always an algorithm that's better than GRModGrasp in terms of mean volume use.

Cargo stability analysis

Stability, which helps prevent cargo from being damaged during transportation, is one of the most important aspects to consider in the CLP. A cargo is considered stable if

- all boxes are fully supported,
- several other boxes support each box, or
- all boxes have at least three sides supported.

Bischoff and Ratcliff presented two measurements for evaluating cargo stability.[6] The first (Measurement1) gives the average number of boxes (higher is better) supporting each box that isn't on the container floor. The second (Measurement2) gives the average percentage of boxes (smaller is better) that aren't surrounded by other boxes on at least three sides.

In our new approach, we can guarantee the full support of the boxes because we especially designed GRMod to meet this objective.

No results concerning Measurement1 and Measurement2 have been published for BR8–BR15. Table 3 shows the BR1–BR7 classes' results compared with the GRModGrasp results. Because we took the H_E results[12] from a graphical representation, they might not be totally accurate. Eberhard Bischoff, H_B's author, directly provided its results. H_B_al holds the best results for Measurement1 in all problems. For Measurement2, the best results are spread out among all the compared approaches. GRModGrasp's low performance on Measurement2 stems directly from the wall-building nature of its basic constructive heuristic. Unfortunately, no results exist for GA_GB, TS_BG, and H_E for Measurement1, but GRModGrasp generally performs quite well when compared with the best published approaches.

Comparison with other metaheuristics

We tested other nonpopulational metaheuristics that use GRMod, such as simulated annealing (GRModSA), tabu search (GRModTS), and iterated local search (GRModILS). We used the same neighborhood structure as that used in the Grasp local-search phase. Tables 4 and 5 show the results achieved with these three approaches for LN and BR problem tests. The numbers in parentheses stand for the number of unpacked boxes (some approaches weren't able to pack all the cargo). When comparing these three approaches, GRModGrasp outperforms the other metaheuristics. For the BR test problems and for weakly heterogeneous problems, GRModGrasp's results are clearly better than

Table 3. BR problem stability results.*

	H_B_al		H_BR		GA_GB	TS_BG	H_E	H_B		GRModGrasp	
Problem	M1	M2	M1	M2	M2	M2	M2	M1	M2	M1	M2
BR1	2.02	8.50	1.13	10.36	11.00	13.00	9.80	1.17	12.37	1.07	11.53
BR2	2.22	11.21	1.10	14.60	16.00	19.00	13.50	1.14	15.30	1.10	12.67
BR3	2.20	15.93	1.08	19.67	18.50	24.50	18.00	1.09	17.05	1.09	17.75
BR4	2.10	17.51	1.07	23.53	21.50	29.90	20.50	1.07	18.65	1.10	20.03
BR5	2.09	21.60	1.06	26.03	22.50	34.00	21.50	1.06	20.79	1.10	22.75
BR6	2.04	22.13	1.06	31.04	25.00	33.50	22.90	1.05	23.31	1.10	26.50
BR7	1.92	27.07	1.04	35.99	28.50	46.10	26.00	1.03	24.25	1.11	28.86
BR8	—	—	—	—	—	—	—	—	—	1.12	32.77
BR9	—	—	—	—	—	—	—	—	—	1.10	37.49
BR10	—	—	—	—	—	—	—	—	—	1.10	39.21
BR11	—	—	—	—	—	—	—	—	—	1.14	40.63
BR12	—	—	—	—	—	—	—	—	—	1.15	41.44
BR13	—	—	—	—	—	—	—	—	—	1.16	41.67
BR14	—	—	—	—	—	—	—	—	—	1.16	43.14
BR15	—	—	—	—	—	—	—	—	—	1.17	44.12

*The best values appear in blue.

the other metaheuristics' results. For strongly heterogeneous problems, GRModSA almost matches GRModGrasp's results.

The running times for all approaches are almost insignificant for the weakly heterogeneous problems. When the number of different box types increases, GRModGrasp outperforms the other approaches. For the BR problems, the computing time is less than 200 seconds (on a Pentium IV at 2.4 GHz with 480 Mbytes of RAM), even for 100 different types of boxes.

We also studied the cargo stability of the packing solutions obtained with the four metaheuristics, for LN and BR problems (see tables 6 and 7). For LN problems, GRModSA achieved the best values (Measurement1 indicator). For BR problems, GRModGrasp and GRModILS achieved the best results for Measurement1 and Measurement2, respectively.

Table 4. LN test problem results for different metaheuristics.*

	GRMod		GRModGrasp		GRModSA		GRModTS		GRModILS	
Problem	Vol. use (%)	Computing time (sec.)	Vol. use (%)	Computing time (sec.)	Vol. use (%)	Computing time (sec.)	Vol. use (%)	Computing time (sec.)	Vol. use (%)	Computing time (sec.)
LN01	62.5	< 1	62.5	28	62.5	97	62.5	74	62.5	10
LN02	89.2 (34)	< 1	92.6 (19)	45	91.3 (25)	103	90.4 (27)	86	91.7 (24)	7
LN03	53.4	< 1	53.4	105	53.4	56	53.4	102	53.4	9
LN04	55.0	< 1	55.0	54	55.0	140	55.0	93	55.0	13
LN05	75.9 (3)	< 1	77.2	16	77.2	68	77.2	54	77.2	4
LN06	84.6 (49)	< 1	91.7 (28)	34	86.9 (34)	87	91.7 (31)	64	91.7 (31)	6
LN07	79.3 (22)	< 1	84.7	56	82.3 (4)	73	82.6 (3)	25	82.6 (4)	8
LN08	59.4	< 1	59.4	20	59.4	129	59.4	16	59.4	5
LN09	61.9	< 1	61.9	78	61.9	144	61.9	91	61.9	16
LN10	67.3	< 1	67.3	54	67.3	67	67.3	103	67.3	5
LN11	62.2	< 1	62.2	34	62.2	112	62.2	12	62.2	8
LN12	75.4 (6)	< 1	78.5	57	78.5	128	78.5	27	78.5	9
LN13	81.8 (7)	< 1	85.6	68	85.6	101	81.8 (7)	36	85.6	11
LN14	62.8	< 1	62.8	54	62.8	98	62.8	17	62.8	7
LN15	59.5	< 1	59.5	42	59.5	119	59.5	48	59.5	12
Mean	**68.7**	**< 1**	**70.3**	**49.7**	**69.7**	**101.5**	**69.7**	**56.5**	**70.1**	**8.7**

*The best values appear in blue.

Table 5. BR test problem results for metaheuristics.*

	GRModGRMod		GRModGrasp		GRModSA		GRModTS		GRModILS	
Problem	Vol. use (%)	Computing time (sec.)	Vol. use (%)	Computing time (sec.)	Vol. use (%)	Computing time (sec.)	Vol. use (%)	Computing time (sec.)	Vol. use (%)	Computing time (sec.)
BR1	86.67	< 1	**89.07**	8	88.14	33	88.21	7	88.05	**2**
BR2	87.77	< 1	**90.43**	12	89.19	50	89.50	11	89.05	**3**
BR3	87.19	< 1	**90.86**	25	89.47	89	89.52	15	89.22	**5**
BR4	86.21	< 1	**90.42**	28	89.14	106	89.06	18	88.89	**9**
BR5	85.54	< 1	**89.67**	40	88.92	127	88.88	19	88.62	**10**
BR6	84.20	< 1	**89.71**	59	88.55	212	88.47	85	88.12	**19**
BR7	82.37	< 1	**88.05**	64	87.57	274	87.63	102	86.88	**37**
BR8	79.24	< 1	**86.13**	**71**	85.97	304	85.60	327	85.49	78
BR9	76.81	< 1	**85.08**	**85**	85.08	329	84.37	498	84.10	120
BR10	73.76	< 1	**84.21**	**89**	84.20	371	83.82	589	83.37	219
BR11	71.11	< 1	83.98	**94**	**84.08**	405	83.14	601	82.50	416
BR12	67.59	< 1	**83.64**	**98**	83.54	418	82.94	793	82.49	684
BR13	65.32	< 1	**83.54**	**110**	83.40	522	82.65	935	82.13	1,011
BR14	60.62	< 1	**83.25**	**121**	83.17	511	82.21	1228	81.42	1,235
BR15	59.81	< 1	83.21	**128**	**83.41**	627	82.04	1864	81.08	1,601
Mean	**76.95**	**< 1**	**86.75**	**68.80**	**86.26**	**291.80**	**85.87**	**472.79**	**85.43**	**363.21**

*The best values appear in blue.

In terms of volume use, some approaches perform better than GRModGrasp for weakly heterogeneous problems, while others perform better than GRModGrasp for strongly heterogeneous problems. No approach outperforms our algorithm for all 15 problem classes. For cargo stability and volume use, we made similar conclusions. For cargo stability, some approaches perform better than GRModGrasp, while for volume use, other approaches perform better.

However, most important, approaches that outperform GRModGrasp in cargo stability don't outperform it for volume use, with the exception of H_B, which behaves much like GRModGrasp, without either approach clearly dominating.

Finally, all metaheuristics obtained solutions in a very small amount of time, even for problems with many different box types.

Table 6. Metaheuristics stability results for LN test problems.*

	GRMod		GRModGrasp		GRModSA		GRModTS		GRModILS	
Problem	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2
LN01	1.00	14.00	**1.07**	17.00	1.00	14.00	1.00	**3.30**	1.00	15.00
LN02	1.01	5.42	1.00	**2.89**	1.00	3.01	1.00	**2.89**	**1.07**	5.88
LN03	1.08	8.00	**1.03**	16.05	**1.03**	**6.50**	1.01	7.07	**1.03**	9.50
LN04	1.00	18.00	1.00	17.00	**1.01**	**3.00**	1.00	16.00	1.00	16.00
LN05	1.10	5.13	**1.36**	12.50	1.16	**3.42**	1.15	6.67	**1.36**	12.50
LN06	1.01	13.91	**1.08**	**4.73**	1.00	9.93	**1.08**	**4.73**	**1.08**	**4.73**
LN07	1.05	10.11	1.05	**2.03**	1.04	3.93	1.05	**2.03**	**1.08**	6.63
LN08	1.01	12.31	**1.04**	13.85	1.01	16.15	1.01	13.08	1.02	**6.92**
LN09	1.06	20.50	**1.07**	15.00	**1.07**	**9.00**	**1.07**	19.00	1.06	24.50
LN10	1.00	14.80	1.00	12.00	1.01	8.40	1.00	11.60	1.00	**6.80**
LN11	1.00	13.00	**1.00**	**12.00**	**1.00**	14.00	**1.00**	14.00	**1.00**	14.00
LN12	1.00	12.17	1.03	12.50	1.01	**3.51**	1.04	12.50	1.00	8.33
LN13	1.02	8.84	**1.02**	11.54	**1.02**	11.54	**1.02**	**8.94**	**1.02**	11.54
LN14	1.14	14.17	1.02	8.33	**1.10**	**2.50**	**1.10**	3.33	**1.10**	8.33
LN15	1.11	9.60	1.11	9.60	**1.12**	8.80	1.11	15.60	1.10	16.00
Mean	**1.04**	**12.00**	**1.06**	**11.13**	**1.04**	**7.85**	**1.04**	**9.38**	**1.06**	**11.11**

*The best values appear in blue.

Table 7. Metaheuristics stability results for BR test problems.*

	GRMod		GRModGrasp		GRModSA		GRModTS		GRModILS	
Problem	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2
BR1	1.09	12.00	**1.07**	11.53	1.07	8.22	1.06	7.10	1.06	**6.54**
BR2	1.14	13.59	**1.10**	**12.67**	1.04	17.41	1.04	13.80	1.05	15.58
BR3	1.09	17.82	**1.09**	17.75	**1.09**	12.50	1.08	**11.06**	1.05	14.80
BR4	1.11	20.60	1.10	20.03	1.07	21.69	1.09	20.43	**1.14**	18.46
BR5	1.08	24.74	1.10	22.75	**1.13**	23.70	1.07	**20.95**	1.09	21.53
BR6	1.12	27.86	1.10	26.50	**1.11**	27.56	**1.11**	23.67	**1.11**	**19.76**
BR7	1.11	32.00	**1.11**	**28.86**	1.10	31.08	**1.11**	30.53	1.10	33.31
BR8	1.16	38.86	1.12	**32.77**	1.11	33.67	**1.15**	35.25	**1.15**	37.34
BR9	1.19	41.87	1.10	37.49	1.11	**35.30**	**1.15**	38.24	1.12	39.86
BR10	1.25	48.69	1.10	39.21	**1.15**	**36.39**	**1.15**	41.29	**1.15**	48.42
BR11	1.25	53.22	1.14	**40.63**	1.22	43.29	1.18	46.63	**1.23**	48.84
BR12	1.32	54.96	1.15	**41.44**	1.23	50.21	1.21	49.98	**1.29**	49.22
BR13	1.34	57.62	1.16	**41.67**	1.22	51.79	1.23	53.21	**1.29**	54.51
BR14	1.33	58.75	1.16	**43.14**	1.25	56.51	1.25	53.98	**1.33**	51.40
BR15	1.34	60.27	1.17	**44.12**	1.27	58.51	1.27	56.33	**1.39**	55.50
Mean value	1.19	37.52	1.12	30.70	1.14	33.85	1.14	33.50	1.17	34.34

*The best values appear in blue.

Acknowledgments

This work was partially supported by FCT (Fundação de Ciência e Tecnologia), POSI (Programa Operacional Sociedade da Informação), e FEDER (Fundo Europeu de Desenvolvimento Regional), CPackMO: Cutting and Packing Problems with Multiple Objectives POSI/SRI/40908/2001.

The Authors

Ana Moura is an assistant professor in the Department of Electrotechnic Engineering of the Institute Polytechnic of Bragança (ESTiG - IPB) and a researcher in the Manufacturing Systems Engineering Unit at INESC Porto, Portugal. Her research interests include the application of decision and optimization methods to industrial and organizational problems, particularly in packing and vehicle-routing problems. She received her PhD in computer and electrical engineering from the Faculty of Engineering of the University of Porto. Contact her at ESTiG - IPB, Campus de Santa Apolónia, Apt. 134, 5301-857 Bragança, Portugal; amoura@ipb.pt.

José Fernando Oliveira is an associate professor in the Department of Electrical and Computer Engineering in the Faculty of Engineering of the University of Porto (FEUP) and a senior researcher in the Manufacturing Systems Engineering Unit at INESC Porto, Portugal. His research interests include the application of decision and optimization methods to industrial and organizational problems, particularly in cutting and packing problems and decision-support systems. He received his PhD in computer and electrical engineering from FEUP. He is the coordinator of the EURO Working Group ESICUP (Special Interest Group on Cutting and Packing). Contact him at FEUP, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal; jfo@fe.up.pt.

References

1. A. Bortfeldt, H. Gehring, and D. Mack, "A Parallel Tabu Search Algorithm for Solving the Container Loading Problem," *Parallel Computing*, vol. 29, no. 5, 2003, pp. 641–662.
2. M.G.C. Resende and L.S. Pitsoulis, "Greedy Randomized Adaptive Search Procedures," *Handbook of Applied Optimization*, Oxford Univ. Press, 2002.
3. A.J. George and D.F. Robinson, "A Heuristic for Packing Boxes into a Container," *Computers and Operations Research*, vol. 7, no. 3, 1980, pp. 147–156.
4. M.H. Correia et al., "Problemas de Empacotamento Tridimensional [Three-Dimensional Packing Problems]," *Investigação Operacional,* vol. 12, vol. 2, 1992, pp. 169–180 (in Portuguese).
5. H.T. Loh and A.Y.C. Nee, "A Packing Algorithm for Hexahedral Boxes," *Proc. Conf. Industrial Automation*, 1992.
6. E.E. Bischoff and M.S.W. Ratcliff, "Issues in the Development of Approaches to Container Loading," *Omega, Int'l J. Management Science*, vol. 23, no. 3, 1995, pp. 377–390.
7. E.E. Bischoff, F. Janetz, and M.S.W. Ratcliff, "Loading Pallets with Nonidentical Items," *European J. Operational Research*, vol. 84, 1995, pp. 681–692.
8. H. Gehring and A. Bortfeldt, "A Genetic Algorithm for Solving the Container Loading Problem," *Int'l Trans. Operational Research*, vol. 4, 1997, pp. 401–418.
9. A. Bortfeldt and H. Gehring, "A Tabu Search Algorithm for Weakly Heterogeneous Container Loading Problems," *OR Spektrum*, vol. 20, no. 4, 1998, pp. 237–250 (in German).
10. A. Bortfeldt and H. Gehring, "A Hybrid Genetic Algorithm for the Container Loading Problem," *European J. Operational Research*, vol. 131, no. 1, 2001, pp. 143–161.
11. H. Gehring and A. Bortfeldt, "A Parallel Genetic Algorithm for Solving the Container Loading Problem," *Int'l Trans. Operational Research*, vol. 9, no. 4, 2002, pp. 497–511.
12. M. Eley, "Solving Container Loading Problems by Block Arrangement," *European J. Operational Research*, vol. 141, no. 2, 2002, pp. 393–409.
13. E.E. Bischoff, *Dealing with Load Bearing Strength Considerations in Container Loading Problems*, tech. report, European Business Management School, Univ. of Wales, Swansea, 2003.