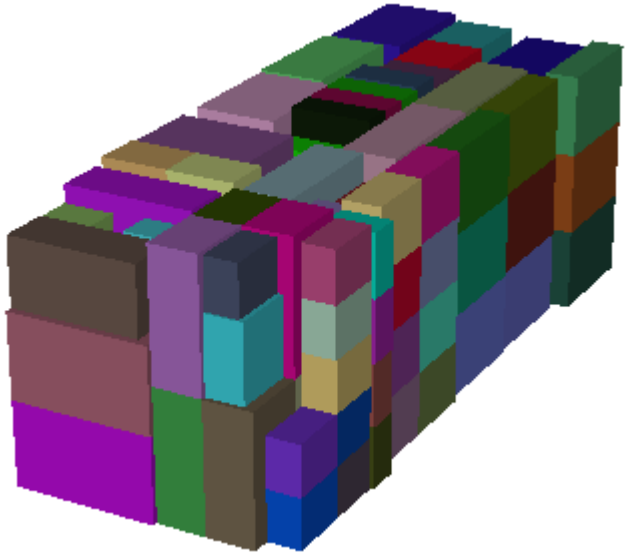
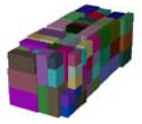
Container loading by GRASP



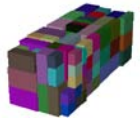
2nd ESICUP Meeting - Southampton, UK, April 14-16, 2005

Hugo Duque Caldeira - ISEP; INESC Porto
José Soeiro Ferreira - FEUP; INESC Porto

Container loading by GRASP

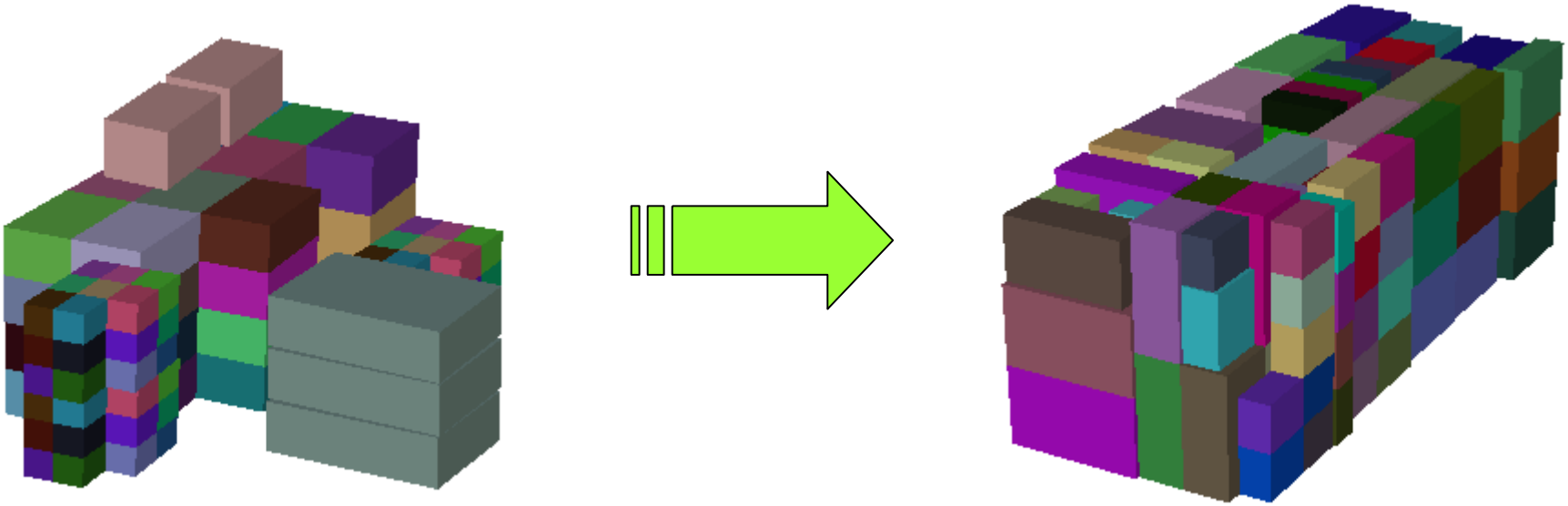


1. Introduction
2. Problem
3. Resolution methods
4. Application of GRASP
5. Results
6. Minimum number of containers
7. Conclusions

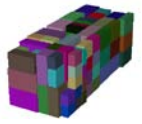


Problem

How to load several rectangular objects into one or more containers, optimising the space available?



1. Introduction



Relevance of the problem

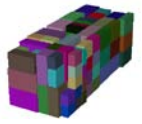
Applications

- Transporters
- Industries delivering goods
- Molds industries

Complexity

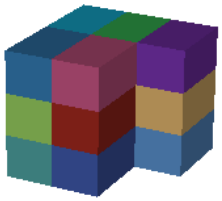
- Combinatorial Optimisation
- Geometrical restrictions

2. Problem



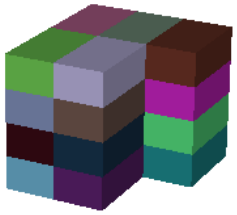
Boxes

type 1



width: 68 cm
length:109 cm
height: 59 cm
quantity: 15

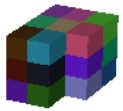
type 2



width: 71 cm
length: 120 cm
height: 47 cm
quantity: 21

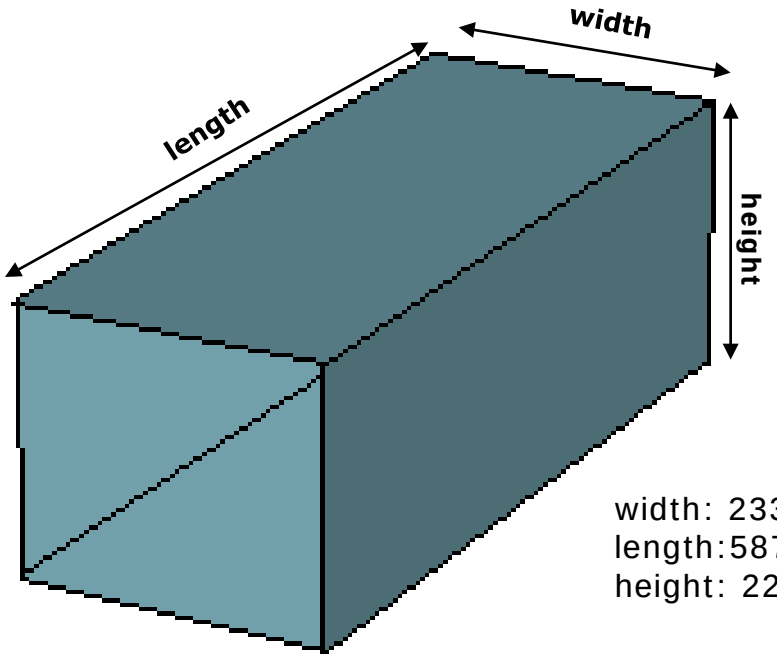
...

type 8



width: 32 cm
length: 49 cm
height: 30 cm
quantity: 24

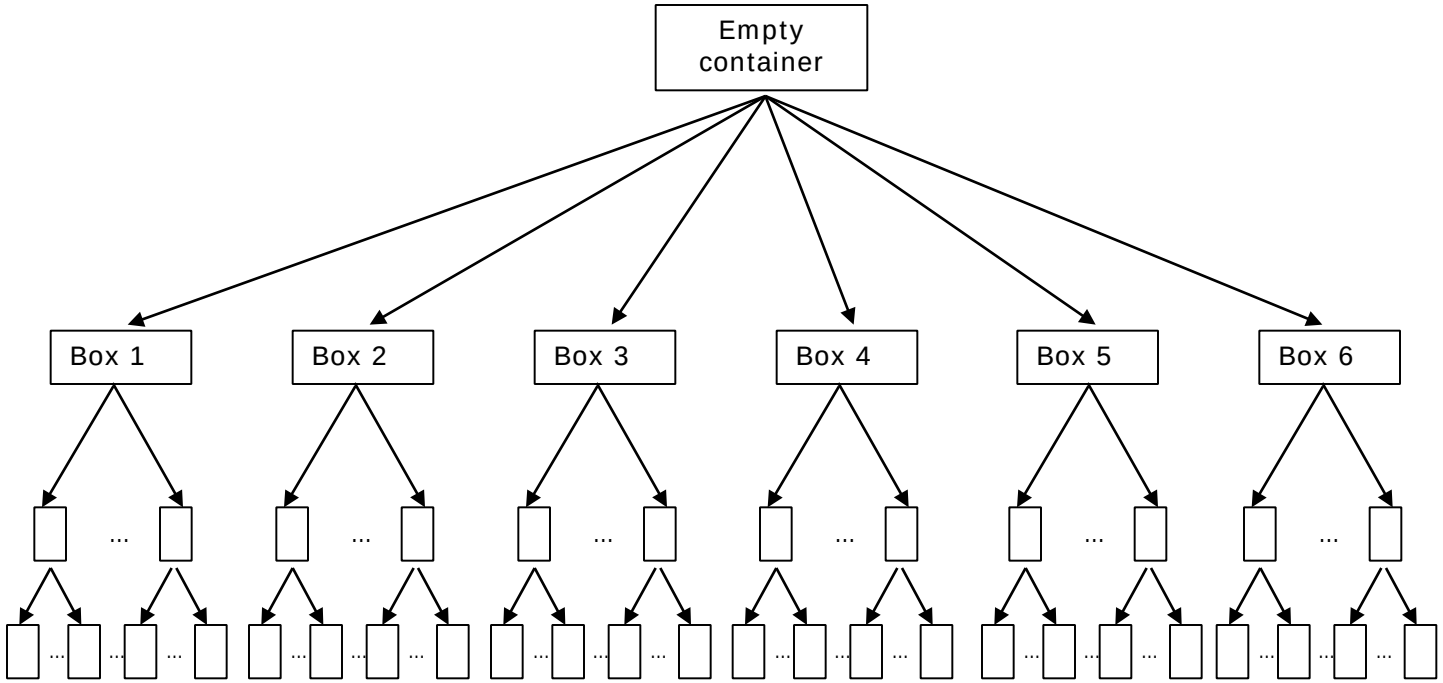
Container



width: 233 cm
length:587 cm
height: 220 cm

2. Problem

Complexity



1st box
6 possibilities

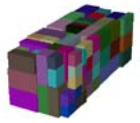
2nd box
30 possibilities

3rd box
120 possibilities

For 20 boxes there are 2432902008176640000 possibilities.

If one solution takes 10^{-5} seconds to be built, for 20 boxes it is necessary almost 7822 centuries to check all possibilities.

3. Resolution methods



George & Robinson (1980) – Dedicated Heuristic

Morabito & Arenales (1994) – AND/OR Graph

Chen, Lee & Shen (1995) –Analytical Model

Gehring & Bortfeldt (1997) – Genetic Algorithm

Michael Eley (2001) – Block Arrangement

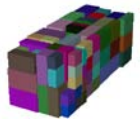
Hifi (2002, 2004) – Approximate and Exact Algorithms

Bortfeldt & Gehring (2002) - Parallel Genetic Algorithm

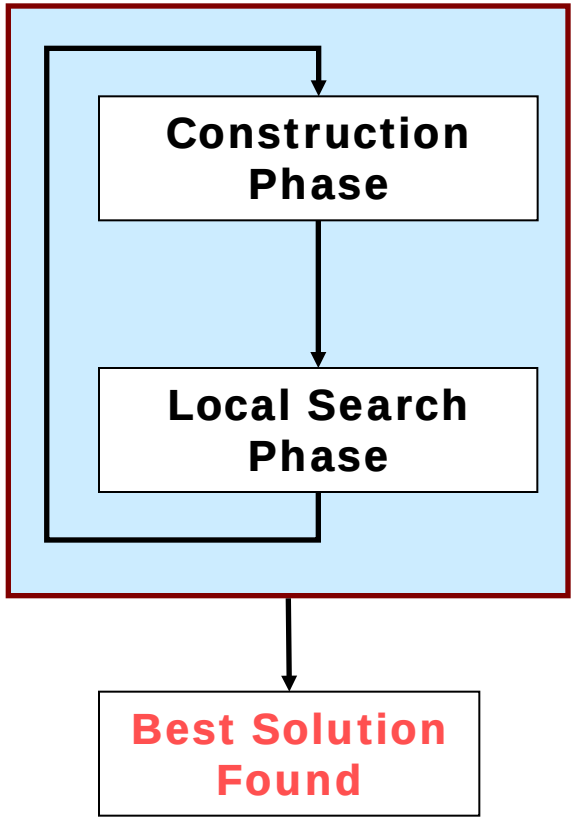
Bortfeldt, Gehring & Mack (2002) - Parallel Tabu Search

Mack, Bortfeldt & Gehring (2004) – Parallel Hybrid Local Search

4. Application of GRASP



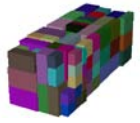
GRASP - Greedy Randomized Adaptive Search Procedures

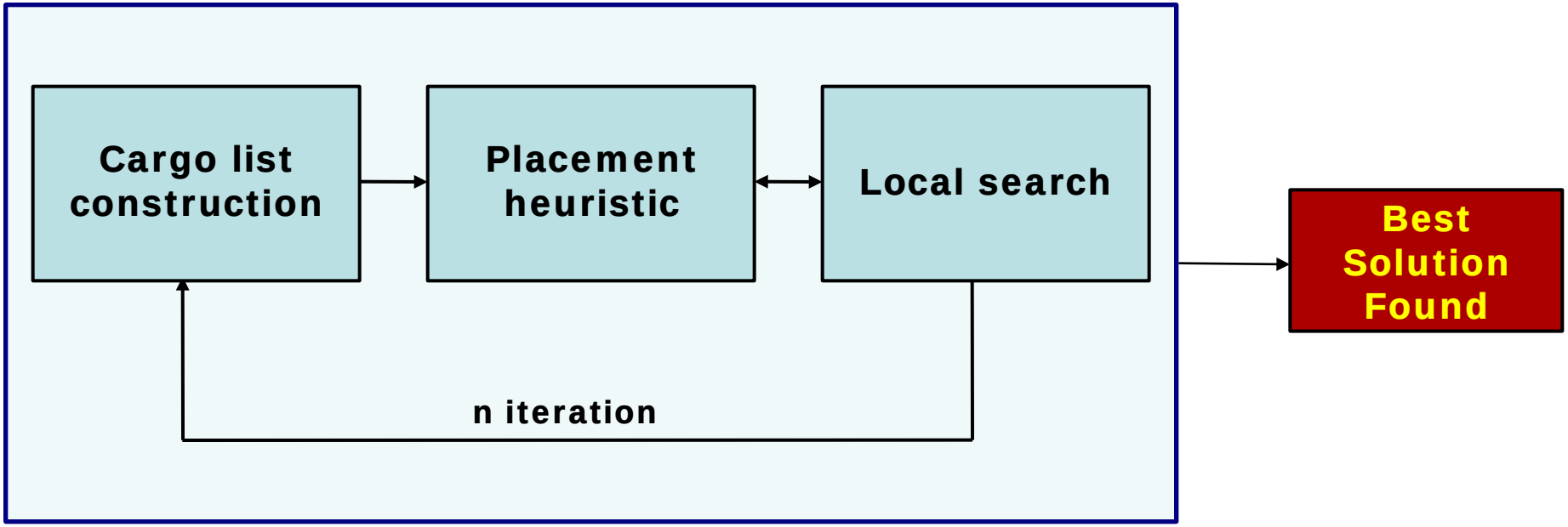



```
procedure construct(α,x)
   Initialize candidate list C;
   while C not empty do
      RCL=α best candidates from C;
      Select s, at random, from the RCL;
      Add s to solution x;
      Update candidate set C;
   end while;
   f=value of solution x
end construct;
```

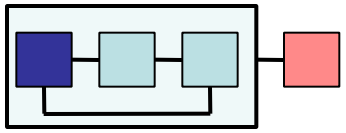
```
procedure localsearch(N(x),x)
   for all y ∈ N(x)
      g=value of solution y
      if g>f then
         solution x=solution y
         f=g
      end if
   end for
end localsearch;
```

4. Application of GRASP





4. Application of GRASP



Cargo list construction

Boxes	Volume (m3)	Qtt
Box4	0,44	2
Box6	0,40	3
Box7	0,25	1
Box3	0,20	5
Box1	0,11	8
...	...	...

→

Restrict candidate list
Box4
Box6
Box7

α
best candidates

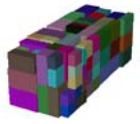
→

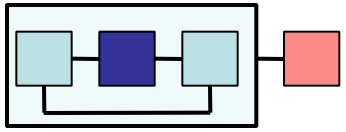
Selected
Box4

→

Cargo list
Box4
Box4

4. Application of GRASP



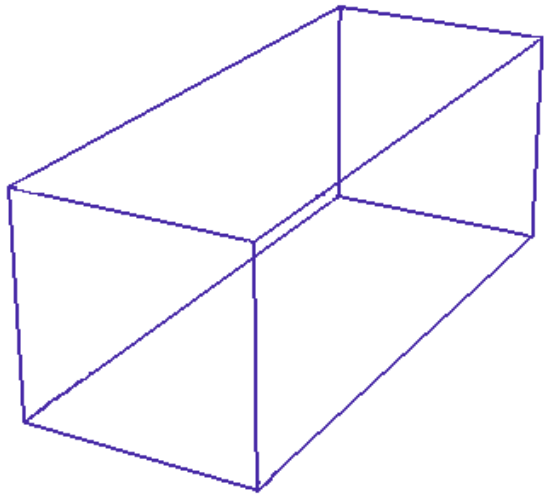


Placement heuristic

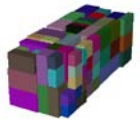
Cargo list
Box4
Box4
Box7
Box6
Box6
Box6
...

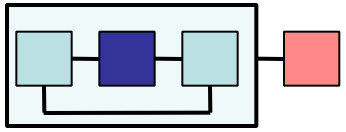
Space	x	y	z	width	length	height
S1	0	0	0	233	587	220

Placed box list
(empty)



4. Application of GRASP



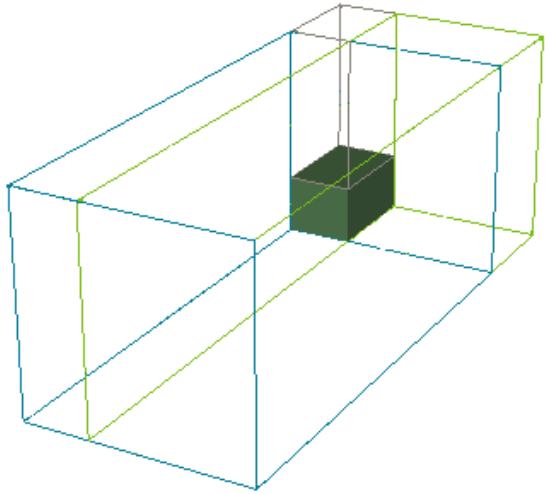


Placement heuristic

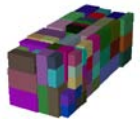
Cargo list
Box4
Box7
Box6
Box6
Box6
Box3
...

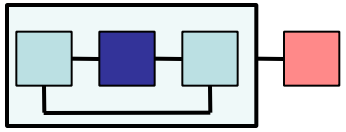
Space	x	y	z	width	length	height
S2	68	0	0	165	587	220
S3	0	0	59	68	109	161
S4	0	109	0	233	478	220

Placed box list
Box4



4. Application of GRASP



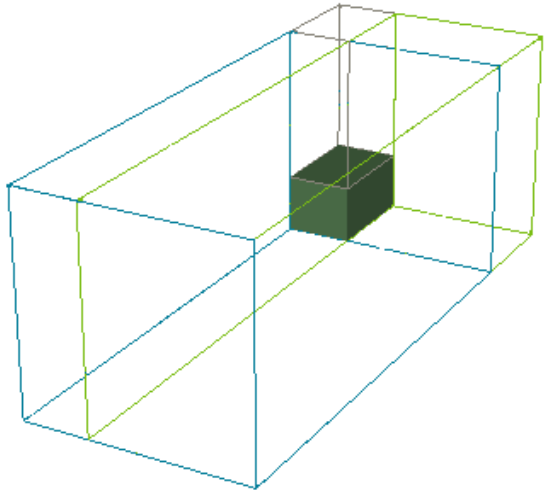


Placement heuristic

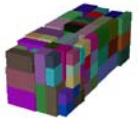
```
for all available spaces
   for all possible orientations
      evaluate box placement
   end-for
end-for
```

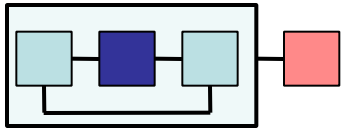
Points

- No remaining space: big M
- 1 point for each type of box that fits in the remaining spaces



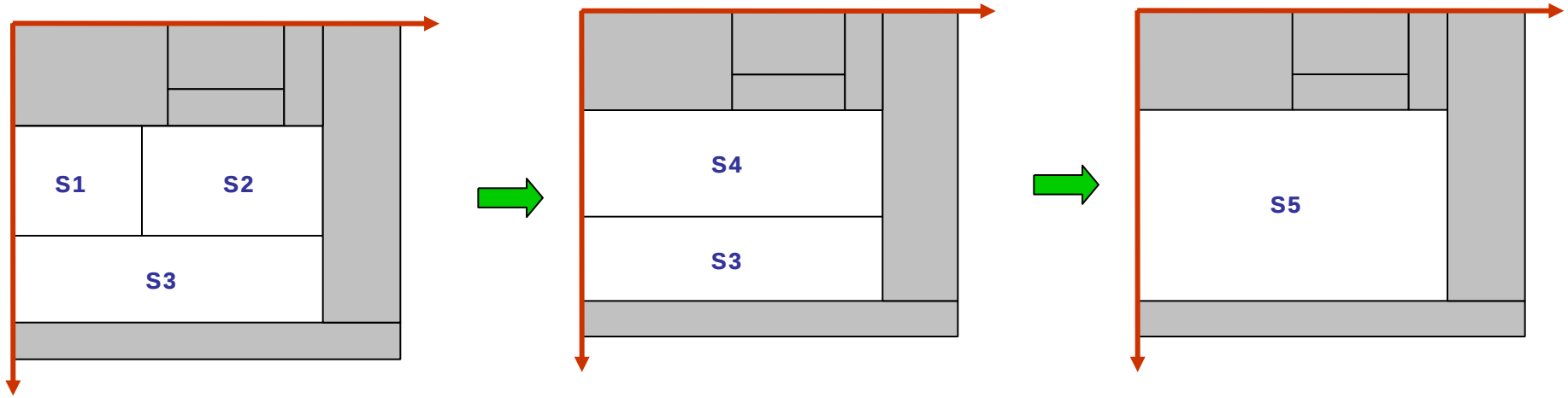
4. Application of GRASP



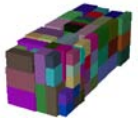


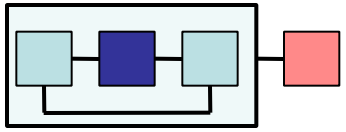
Placement heuristic

- Merging spaces -



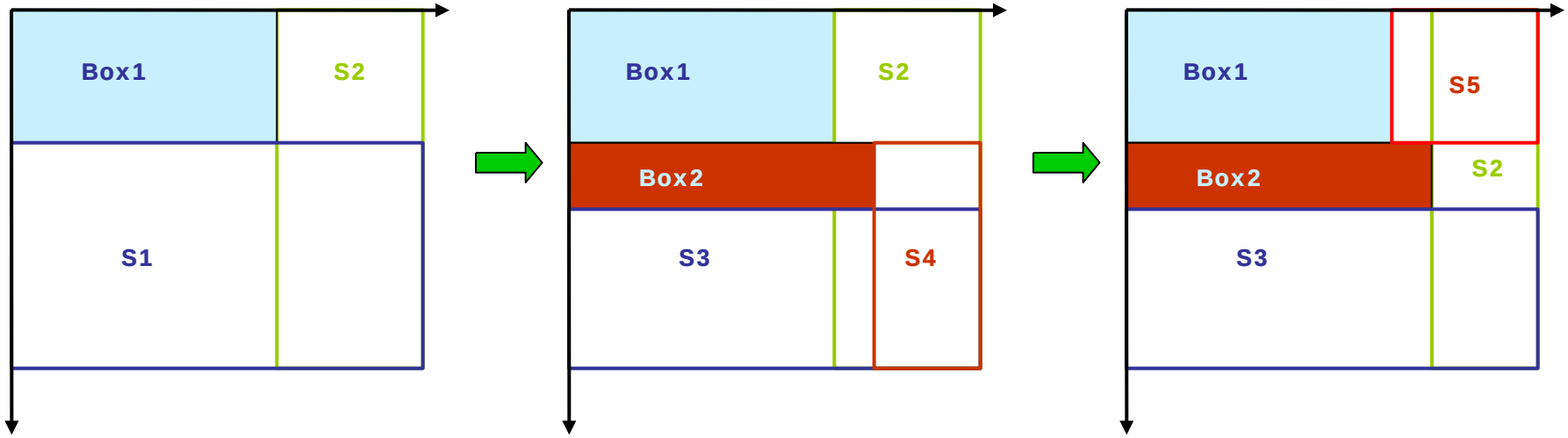
4. Application of GRASP



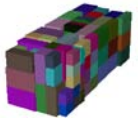


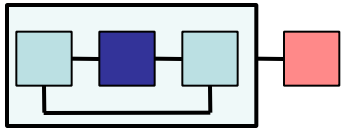
Placement heuristic

- When a box overlaps a space -



4. Application of GRASP



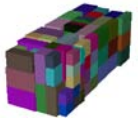


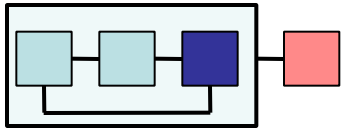
Placement heuristic

- Value of the solution -

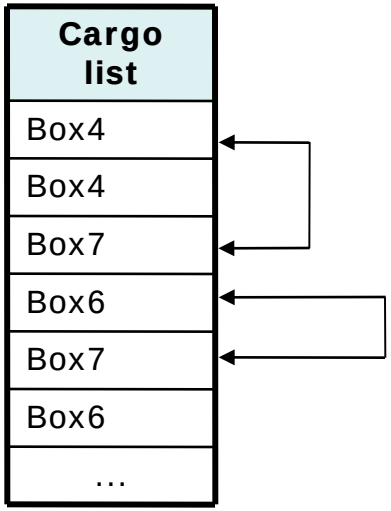
$$\text{Used space} = \frac{\text{Volume of the loaded boxes}}{\text{Volume of the container}}$$

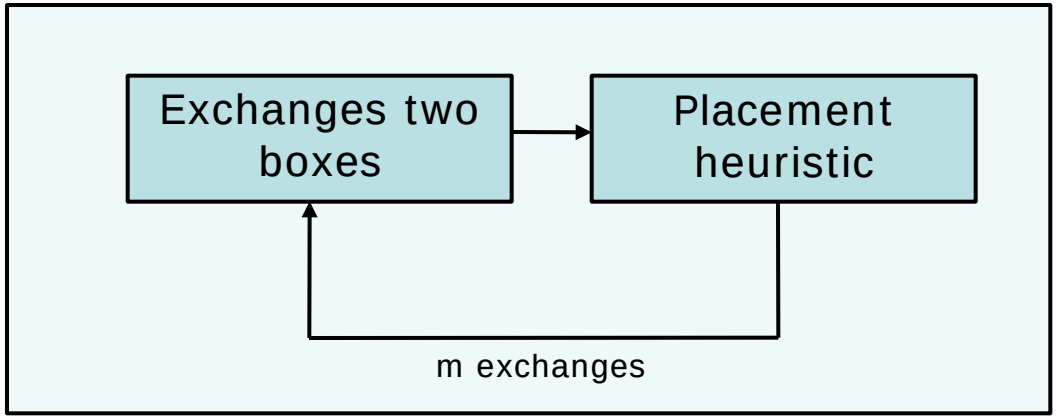
4. Application of GRASP



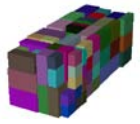


Local search



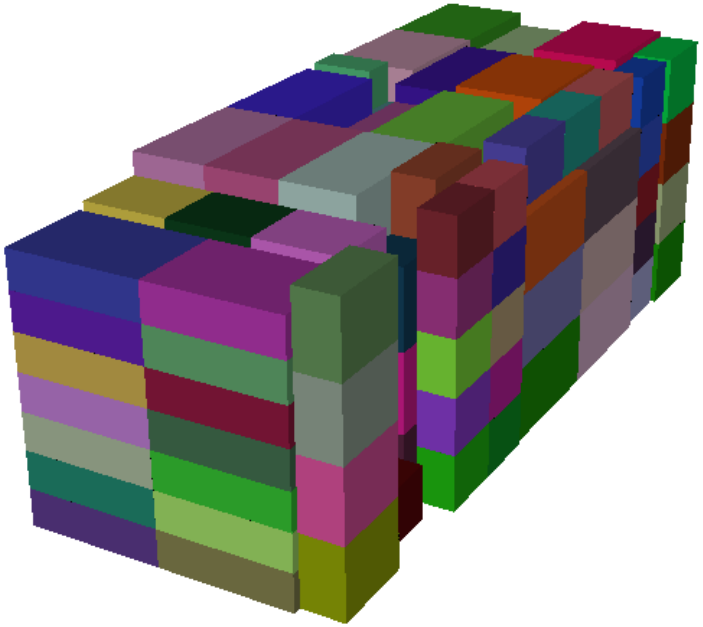


4. Application of GRASP



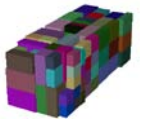
Example of a solution

box	width	vertical	length	vertical	height	qtt.	vol. (m3)
Box4	68	Yes	109	Yes	59	15	0,44
Box6	71	Yes	120	No	47	21	0,40
Box7	57	Yes	109	No	41	19	0,25
Box8	70	No	100	No	28	18	0,20
Box1	53	Yes	64	No	32	23	0,11
Box3	43	Yes	66	Yes	32	21	0,09
Box2	42	Yes	51	Yes	28	20	0,06
Box5	32	Yes	49	No	30	24	0,05



Number of iterations: **21**
Computing time: **38 seconds**
Best solution founded: **90, 82% used space**

5. Results



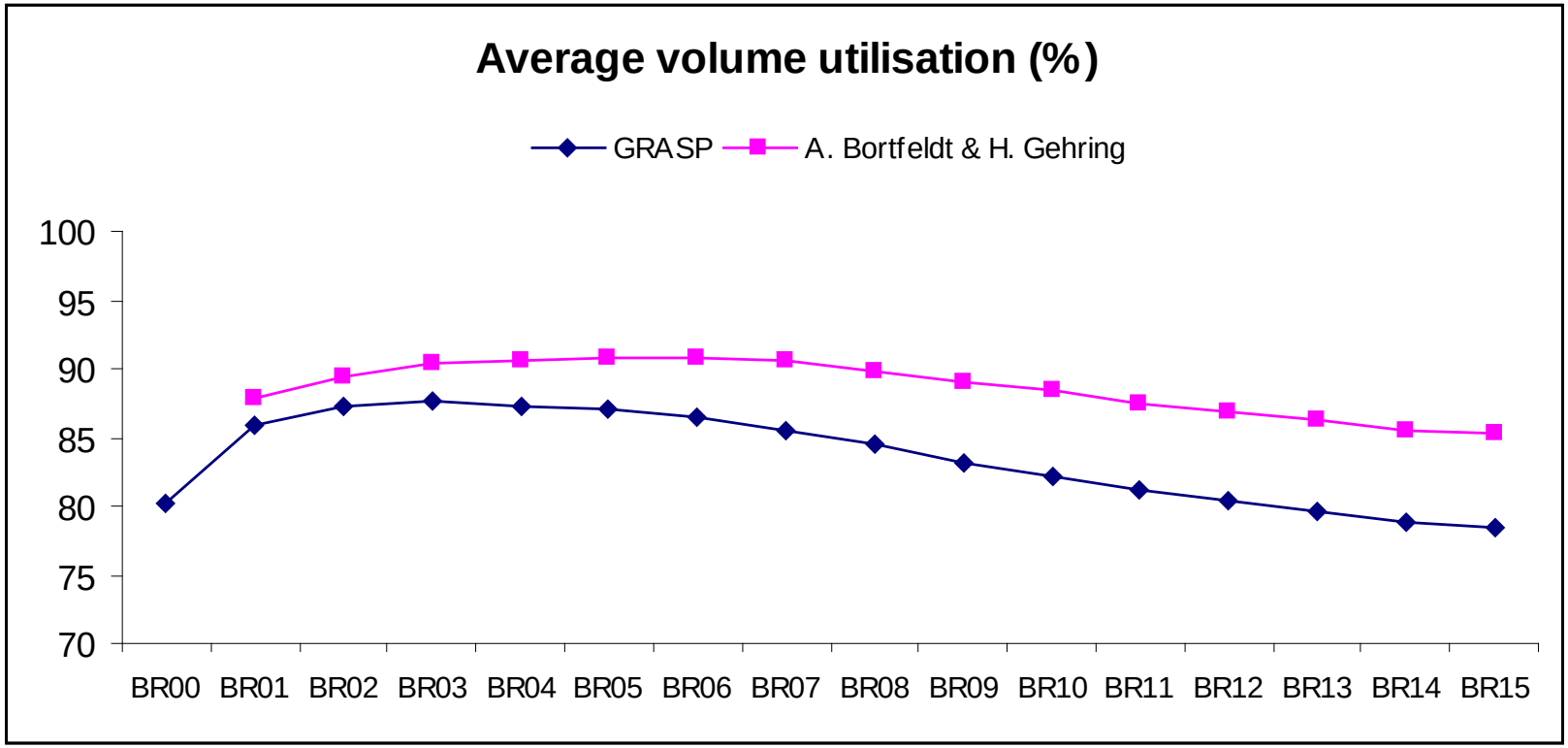
Test cases BR00 to BR15 from Bischoff and Ratcliff (1995)

Class	Types of boxes	Volume utilisation (%)		Time (seconds)	
		Average	SD	Average	SD
BR00	1	80,13	8,65	0,13	0,06
BR01	3	86,03	3,87	6,60	4,78
BR02	5	87,51	2,70	13,31	5,38
BR03	8	87,86	2,05	25,57	9,95
BR04	10	87,47	1,99	29,73	10,31
BR05	12	87,23	1,79	34,66	10,80
BR06	15	86,58	1,60	36,99	9,98
BR07	20	85,63	1,48	41,51	9,95
BR08	30	84,46	1,32	49,00	9,92
BR09	40	83,34	1,20	52,90	9,76
BR10	50	82,26	1,35	58,39	10,56
BR11	60	81,37	1,26	63,84	10,60
BR12	70	80,60	1,48	68,07	10,20
BR13	80	79,91	1,35	72,58	10,70
BR14	90	79,04	1,40	74,77	9,28
BR15	100	78,58	1,50	78,28	9,70
All problems		83,63	4,30	44,14	25,68

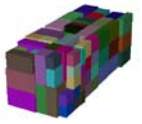
Pentium IV 2,4 GHZ; 768 MB of RAM - JAVA; MySQL; LINUX Kernel 2.6

5. Results

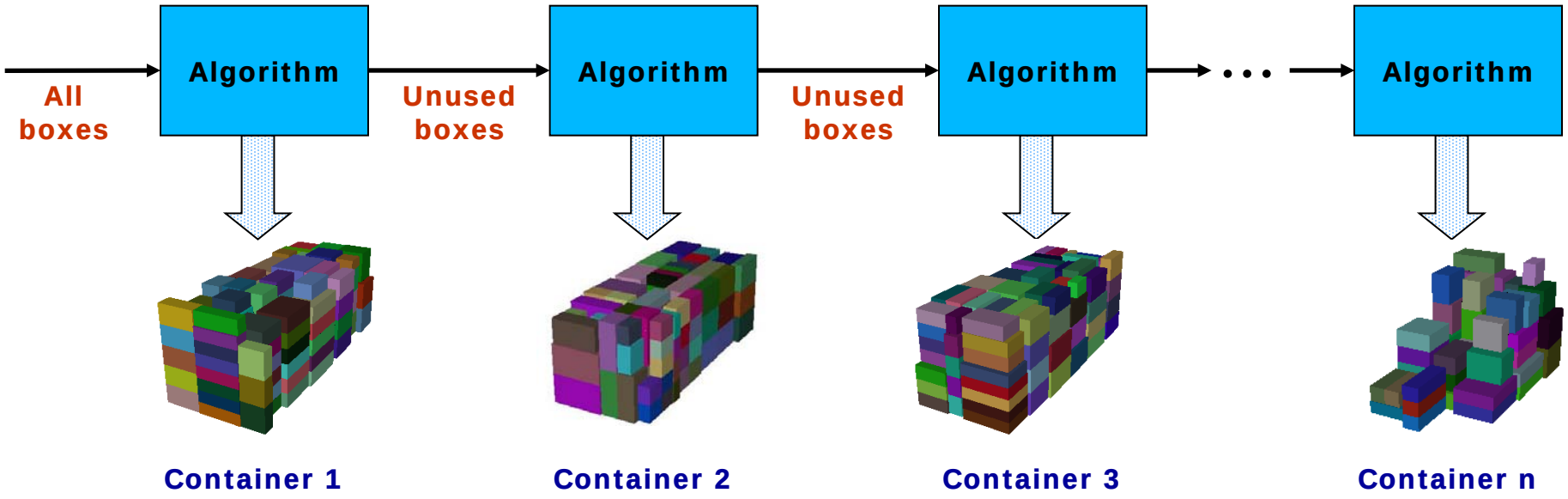
Comparative results



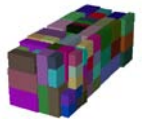
6. Minimum number of containers



Sequential approach



6. Minimum number of containers



Results

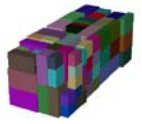
Test cases	Total boxes	Number of type of boxes	Minimum nr. of containers	Last container	Computing time (seconds)
BR00	2032	9	6	89,34%	25
BR01	1264	18	8	22,78%	791
BR02	1227	45	9	1,69%	1210
BR03	1174	72	9	50,9%	1532
BR04	1189	90	9	48,05%	1588
BR05	1145	108	9	59,59%	1674
BR06	1194	135	9	44,54%	1890
BR07	1157	180	9	62,95%	2439
BR08	1229	270	10	7,26%	2888
BR09	1161	360	10	8,15%	2670
BR10	1166	450	10	39,13%	3059
BR11	1149	540	10	25,93%	3137
BR12	1145	630	10	44,07%	3479
BR13	1178	720	11	9,2%	3798
BR14	1165	810	11	1,73%	3963
BR15	1153	900	11	3,34%	4067

Pentium IV 2,4 GHZ – 768 MB of RAM - JAVA; MySQL; LINUX Kernel 2.6

Container	Used vol.
1	95,59%
2	92,81%
3	93,06%
4	91,35%
5	93,33%
6	88,79%
7	89,02%
8	86,75%
9	50,90%

Test cases: the first nine problems from class BR00 to class BR15

7. Conclusions



New container loading heuristic

- **GRASP approach**
- **Method of box placement**
- **Space management**

Future work:

- Improvement of the loading heuristic
  - Evaluation function
  - Alternatives to local search
  - Implement other conditions
- Minimum number of containers