EURO XXII PRAGUE

22nd European Conference on Operational Research, July 8-11, 2007

Neighbourhood structures for the container loading problem: a VNS implementation

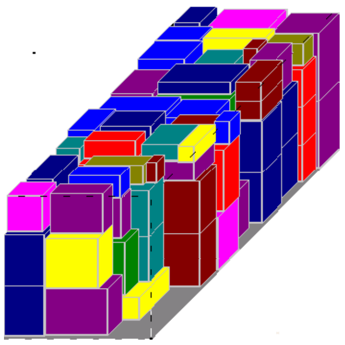
José Fernando Oliveira (1), Francisco Parreño (2), Ramon Alvarez-Valdes (3) and José Manuel Tamarit (3)

(1) University of Porto – Faculty of Engineering and INESC Porto
(2) University of Castilla-La Mancha – Department of Computer Science
(3) University of Valencia – Department of Statistics and Operations Research

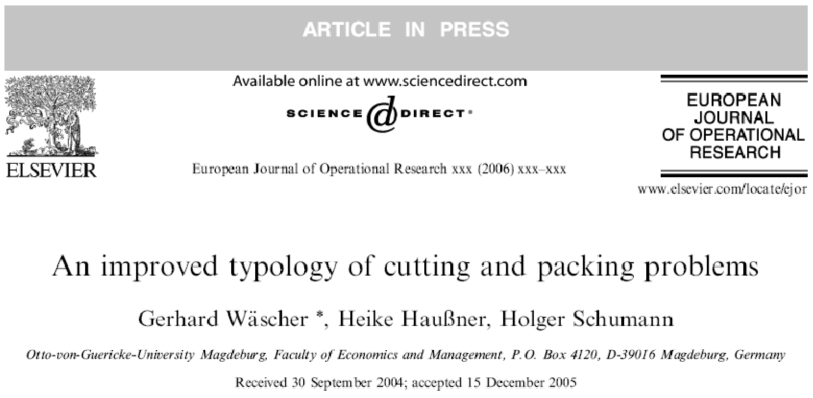
Partially supported by FCT, POSC and FEDER (POSC/EEA-SRI/57367/2004 (APIS)) and Ministerio de Educacion e Ciencia (DPI2005-04796 – OPEC)

Container Loading Problem

- A large 3D parallelepipedic container has to be filled with smaller parallelepipedic boxes, available in different sizes and quantities, so that the container's empty space is minimised, subject to geometric and loading constraints.



Container Loading Problem



ARTICLE IN PRESS

Available online at www.sciencedirect.com

SCIENCE DIRECT

ELSEVIER

European Journal of Operational Research xxx (2006) xxx–xxx

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

An improved typology of cutting and packing problems

Gerhard Wäscher *, Heike Haußner, Holger Schumann

Otto-von-Guericke-University Magdeburg, Faculty of Economics and Management, P.O. Box 4120, D-39016 Magdeburg, Germany

Received 30 September 2004; accepted 15 December 2005

SLOPP/SKP

Container Loading Problem

- An arrangement of boxes is geometrically feasible if the following conditions hold:
  - Each box is placed with its sides parallel to the walls of the container;
  - The container’s dimensions are not exceeded;
  - There are no “intersections” among boxes.
- Loading constraints:
  - Cargo stability
  - Boxes’ orientation (this side-up)
  - Multi-drop loads
  - Weight distribution

Previous work

- George and Robinson (1980)
  - Wall building heuristic
- Gehring and Bortfeld (1997, 1998, 2001, 2002)
  - GA, TS, Hybrid GA, Parallel GA and TS
- Mack, Bortfeld and Gehring (2004)
  - Parallel hybrid LS (SA + TS)
- Moura and Oliveira (2005)
  - GRASP
- Eley (2002)
  - Greedy heuristic + tree search
- Parreño, Alvarez-Valdes, Oliveira, Tamarit (2007)
  - GRASP + maximal-space
- Pisinger (2002)
  - Wall building heuristic

Bischoff and Ratcliff set of problems

The constructive algorithm

Constructive algorithm description

- Step 0: Initialization
- Step 1: Choosing the maximal-space
- Step 2: Choosing the boxes to pack and how to pack them
- Step 3: Updating the list of maximal-spaces

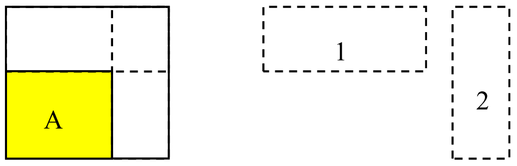
List of empty spaces *versus* list of boxes still to be packed

Step 0: Initialization

- $\mathcal{L} = \{C\}$ (set of empty maximal-spaces)
- $\mathcal{B} = \{b_1, b_2,\ldots,b_m\}$ (set of types of boxes still to be packed)
- $q_i = n_i$ (number of boxes of type *i* to be packed)
- $\mathcal{P} = \varnothing$ (set of boxes already packed)

Step 0 Initialization
Step 1 Choosing the maximal-space
Step 2 Choosing the boxes to pack
Step 3 Updating the list of maximal-spaces

Maximal-space concept (2D)



These spaces are called maximal because at each step they are the largest empty parallelepipedic that can be considered for filling with rectangular boxes.

- Additional complexity to space management procedures;
  - placing a box affects more than one space;
- But:
  - We do not have to decide which disjoint spaces to generate;
  - we do not need to combine disjoint spaces into new ones, to accommodate more boxes (they are maximal)
- Increased quality in container loading problem solutions

Step 1: Choosing the maximal-space

- The choice is based on a measure of the distance of the space to the container's corners:

  $d\,(\,a(x_1,y_1,z_1)\,,\,b(x_2,y_2,z_2)\,) =$ vector of components $|x_1-x_2|$, $|y_1-y_2|$ and $|z_1-z_2|$, ordered by non-decreasing order

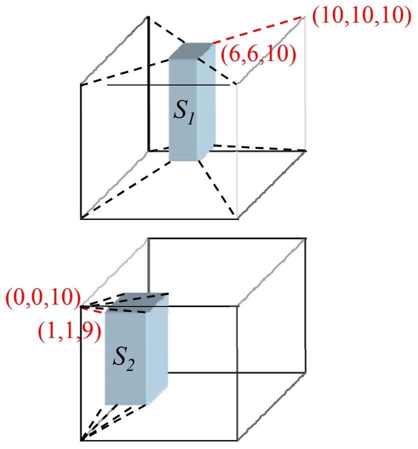
  $a = (3,3,2);\ b = (0,5,10) \Rightarrow d(a,b) = (2,3,8)$
- For each (new) maximal-space the distance from every corner of the space to the nearest corner of the container is computed and kept in lexicographic order:

  $d(S) = \min\{d(a,c),\ a \text{ vertex of } S,\ c \text{ vertex of container } C\}$

Step 0 Initialization
Step 1 Choosing the maximal-space
Step 2 Choosing the boxes to pack
Step 3 Updating the list of maximal-spaces

Step 1: Choosing the maximal-space

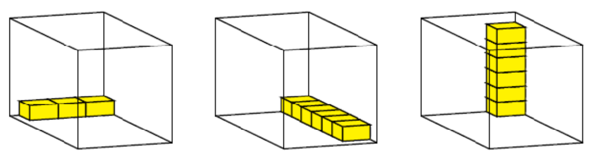
- (10,10,10) container
- $S_1 = \{(4,4,2), (6,6,10)\}$
  Corner of S_1 nearest to a corner of the container: (6,6,10)
  $d(S_1) = (0,4,4)$
- $S_2 = \{(1,1,2), (4,4,9)\}$
  Corner of S_2 nearest to a corner of the container: (1,1,9)
  $d(S_2) = (1,1,1)$
- $S^* = S_1$
  (tie-breaker: space's volume)
- Corners > Sides > Inner space

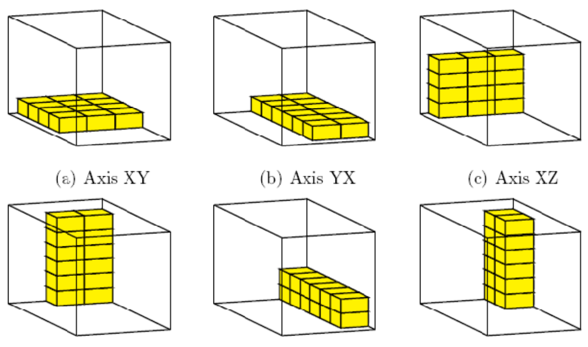


Step 0 Initialization
Step 1 Choosing the maximal-space
Step 2 Choosing the boxes to pack
Step 3 Updating the list of maximal-spaces

Step 2: Choosing the boxes to pack

- Columns


(a) Axis X (b) Axis Y (c) Axis Z


(a) Axis XY (b) Axis YX (c) Axis XZ
(d) Axis ZX (e) Axis YZ (f) Axis ZY

- Layers
  If only 12 boxes were available

Step 0 Initialization
Step 1 Choosing the maximal-space
Step 2 Choosing the boxes to pack
Step 3 Updating the list of maximal-spaces

Step 2: Choosing the boxes to pack

- Criterion 1

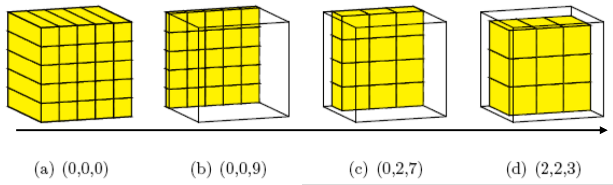
  The block of boxes producing the largest increase in the objective function (volume occupied by boxes).
- Criterion 2

  The block of boxes which fits best, in a lexicographical sense, into the maximal-space.

Update $\mathcal{P}$ with the type i

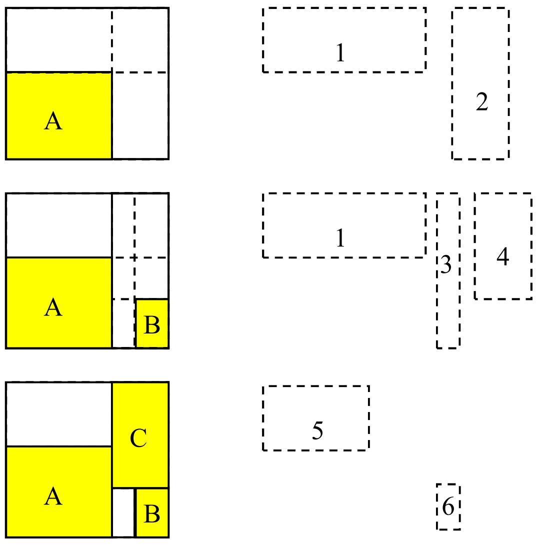
Set $q_i = q_i - r_i$

If $q_i = 0$, remove piece i from $\mathcal{B}$



Step 0 Initialization
Step 1 Choosing the maximal-space
Step 2 Choosing the boxes to pack
Step 3 Updating the list of maximal-spaces

Step 3: Updating the list of maximal-spaces



- Packing a block may intersect more than one space:
  - Old space is removed (2)
  - New spaces are added (3,4)
  - Existing spaces are reduced ($1 \rightarrow 5$)

Step 0 Initialization
Step 1 Choosing the maximal-space
Step 2 Choosing the boxes to pack
Step 3 Updating the list of maximal-spaces

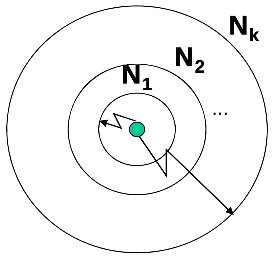
The VNS framework

The VNS framework

- **Variable Neighbourhood Search** (VNS) is a metaheuristic based on systematic changes of neighbourhoods.
- Usually, heuristic and meta-heuristics search over solution spaces considering one *neighbourhood structure*, i.e. one way of transforming one solution in another one (*movements*).
- VNS uses a ***series of neighbourhoods*** N_k, $k = 1, \ldots, kmax$.
- The basic idea of the VNS is to change the neighbourhood structure whenever local search gets trapped on a local minimum.

VND: Variable Neighbourhood Descent

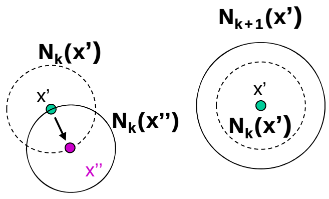
- The **VND** (***Variable Neighbourhood Descent***) method changes to another neighbourhood N_k each time a local optimum is reached.
- It ends when there is not improve with the all neighbourhoods
- The final solution provided by the algorithm should be a local optimum with respect to all k_{max} neighbourhoods.



VNS: Variable Neighbourhood Search

Initialization:
Select neighbourhoods *Mp*, for *p* = 1, ..., *pmax*, that will be used in the shaking phase
Select neighbourhoods *Nk*, for *k* = 1, ..., *kmax,*,that will be used in the local search.
Obtain initial solution *x* by applying the constructive algorithm.
Repeat the following sequence until the stopping condition is met:

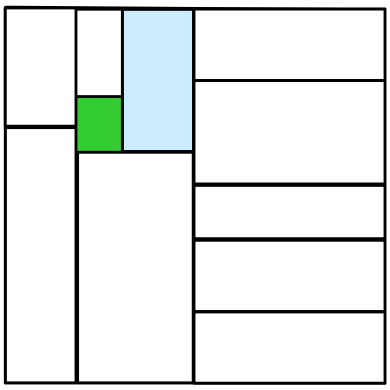
```
(1) Set $p \leftarrow 1$
(2) Repeat the following steps until $p = pmax$
        (a) Shaking. Generate at random a neighboor $x'$ of $x$,
            using the pth neighbourhood.
        (b) Local search by VND.
                (b1) Set $k \leftarrow 1$;
                (b2) Repeat the following steps until $k = kmax$
                        - Find the best neighboor $x''$ of $x'$ in $Nk(x)$;
                        - If the solution $x''$ is better than $x'$, set $x' \leftarrow x''$ and $k \leftarrow 1$.
                          Otherwise, set $k \leftarrow k + 1$;
        (c) If this local optimum $x'$ is better than the incumbent $x$, move
         ($x \leftarrow x'$) and continue the search with M1; otherwise, set $p \leftarrow p + 1$
```



Neighbourhood structures

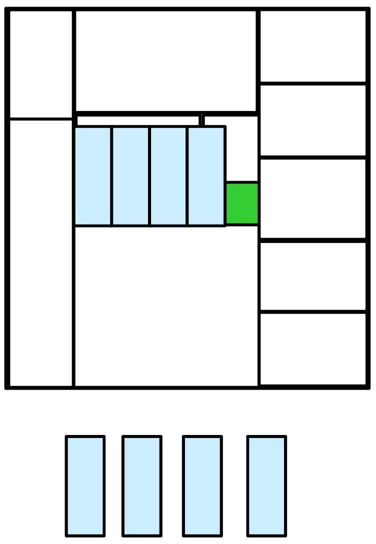
Definition of movements: N_1

- Layer reduction
  - Choosing the layer to reduce
  - Move the remaining layers to the container's nearest corner, measured by the lexicographic distance.
  - Update the list L of maximal spaces.
  - Fill the empty maximal spaces by applying the constructive algorithm with the Best-volume objective function.



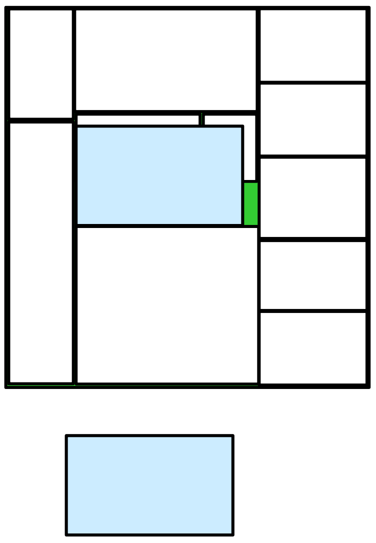
Definition of movements: N_2

- Column insertion
  - Choosing the space
  - Choosing the box to be inserted.
  - Put box B into the corner of S nearest to a corner of the container.
  - Choose a possible direction for building a column of boxes.
  - Remove the overlapping boxes of the container.
  - Update the list L of maximal spaces.
  - Fill the empty maximal spaces by applying the constructive algorithm with the Best-volume objective function.



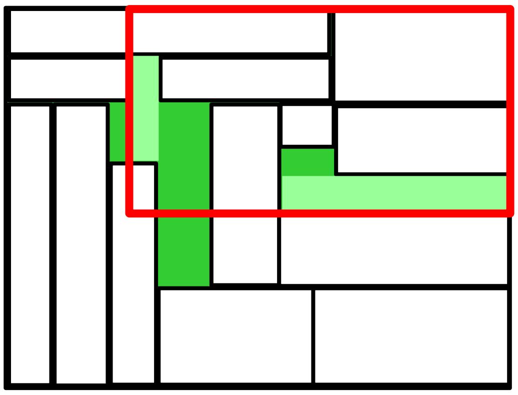
Definition of movements: N_3

- Box insertion
  - Choosing a box to insert.
  - Choosing the space to insert this piece.
  - Choosing the position of B in L
  - Remove the overlapping boxes of the container.
  - Update the list L of maximal spaces.
  - Fill the empty maximal spaces by applying the constructive algorithm with the Best-volume objective function.



Definition of movements: N_4-N_5

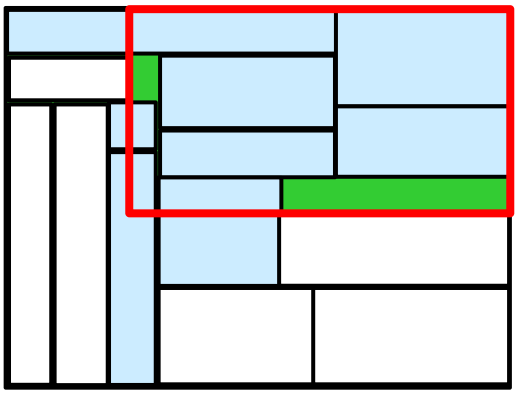
- Emptying a region
  - Take a first space S1
  - From among the spaces smallest than S1, take a second space
  - Create the smallest parallelepiped P containing S1 and S2.
    Remove all the boxes overlapping with P.
  - Update the list L of maximal spaces.
  - Fill the empty spaces by applying the constructive algorithm.



•According to the objective function used to fill the empty spaces, Best-Volume or Best-Fit, we have two different moves.

Definition of movements: N_4-N_5

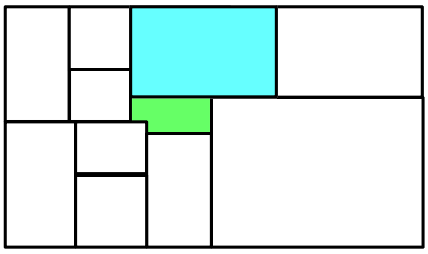
- Emptying a region
  - Take a first space S1
  - From among the spaces smallest than S1, take a second space
  - Create the smallest parallelepiped P containing S1 and S2.
    Remove all the boxes overlapping with P.
  - Update the list L of maximal spaces.
  - Fill the empty spaces by applying the constructive algorithm.



•According to the objective function used to fill the empty spaces, Best-Volume or Best-Fit, we have two different moves.

Definition of movements: N_6

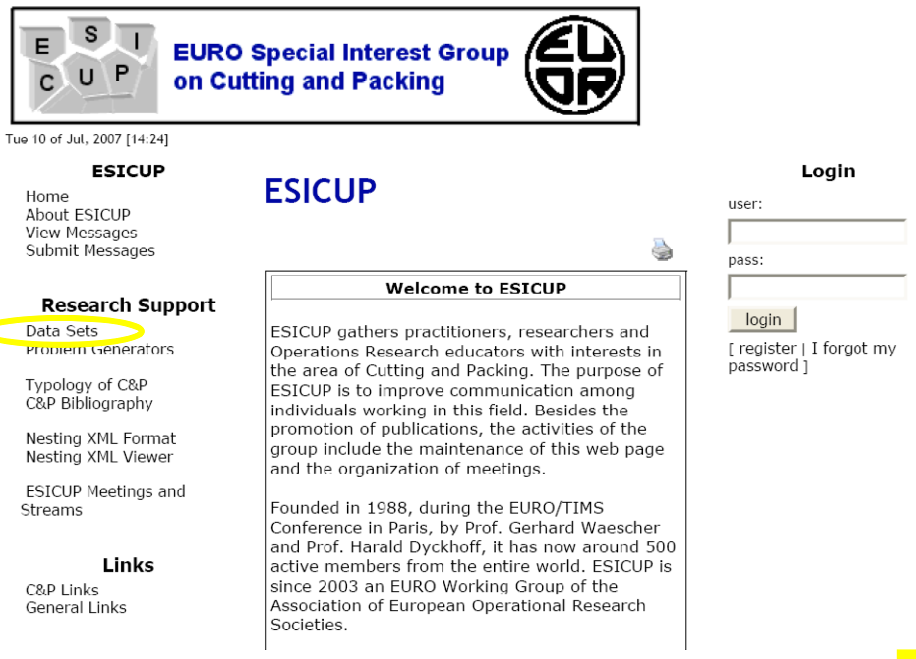
- Eliminating the final *K*% blocks of the solution (10% ≤ K ≤ 30%).
- Filling the empty spaces with the deterministic constructive algorithm.



Computational experiments

Computational experiments

- Algorithms coded in C++
- Pentium Mobile @ 1.5 GHz with 512 MB of RAM
- Test problems (Bischoff and Ratcliff / Davies and Bischoff)
  - 15 classes of 100 problems each
  - Number of box types ranges from 3 (BR1) to 100 (BR15)
    Average number of boxes per type ranges from 50.2 (BR1) to 1.3 (BR15)
    Problems from weakly heterogeneous to strongly heterogeneous
  - Total volume of boxes is on average 99.46% of the container's capacity (no guarantee that all boxes actually fit into the container)



EURO Special Interest Group on Cutting and Packing

Tue 10 of Jul, 2007 [14:24]

ESICUP

Home
About ESICUP
View Messages
Submit Messages

Research Support

Data Sets
Problem Generators

Typology of C&P
C&P Bibliography

Nesting XML Format
Nesting XML Viewer

ESICUP Meetings and Streams

Links

C&P Links
General Links

ESICUP

Welcome to ESICUP

ESICUP gathers practitioners, researchers and Operations Research educators with interests in the area of Cutting and Packing. The purpose of ESICUP is to improve communication among individuals working in this field. Besides the promotion of publications, the activities of the group include the maintenance of this web page and the organization of meetings.

Founded in 1988, during the EURO/TIMS Conference in Paris, by Prof. Gerhard Waescher and Prof. Harald Dyckhoff, it has now around 500 active members from the entire world. ESICUP is since 2003 an EURO Working Group of the Association of European Operational Research Societies.

Login

user:

pass:

login

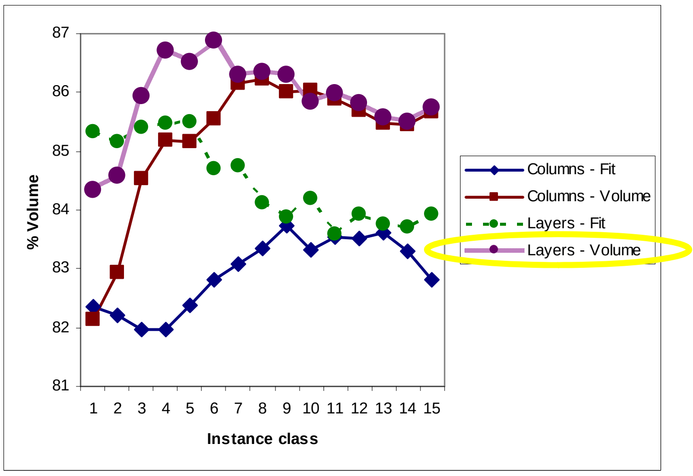
[register | I forgot my password]

http://www.fe.up.pt/~esicup

Preliminary experiments

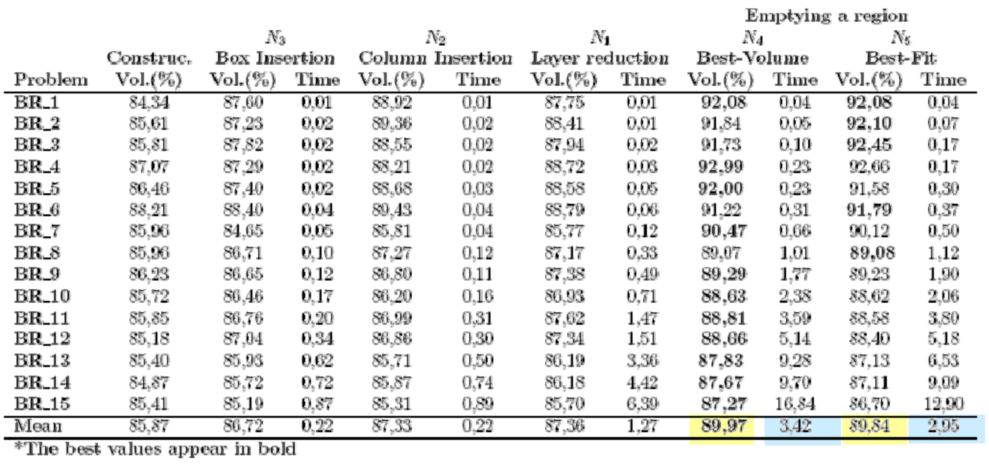
Initial solution – The constructive algorithm

- Comparing objectives and strategies



Comparing the neighbourhoods

15 × 10 problems

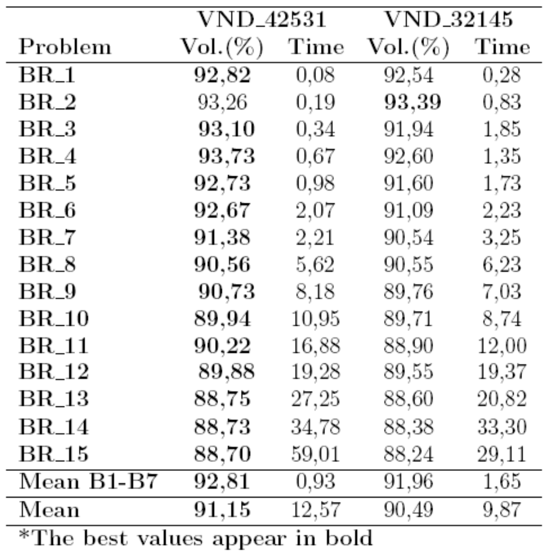


		N_3		N_2		N_1		Emptying a region N_4		N_5	
	Construc.	Box Insertion		Column Insertion		Layer reduction		Best-Volume		Best-Fit	
Problem	Vol.(%)	Vol.(%)	Time	Vol.(%)	Time	Vol.(%)	Time	Vol.(%)	Time	Vol.(%)	Time
BR_1	84,34	87,60	0,01	88,92	0,01	87,75	0,01	**92,08**	0,04	**92,08**	0,04
BR_2	85,61	87,23	0,02	89,36	0,02	88,41	0,01	91,84	0,05	**92,10**	0,07
BR_3	85,81	87,82	0,02	88,55	0,02	87,94	0,02	91,73	0,10	**92,45**	0,17
BR_4	87,07	87,29	0,02	88,21	0,02	88,72	0,03	**92,99**	0,23	92,66	0,17
BR_5	86,46	87,40	0,02	88,68	0,03	88,58	0,05	**92,00**	0,23	91,58	0,30
BR_6	88,21	88,40	0,04	89,43	0,04	88,79	0,06	91,22	0,31	**91,79**	0,37
BR_7	85,96	84,65	0,05	85,81	0,04	85,77	0,12	**90,47**	0,66	90,12	0,50
BR_8	85,96	86,71	0,10	87,27	0,12	87,17	0,33	89,07	1,01	**89,08**	1,12
BR_9	86,23	86,65	0,12	86,80	0,11	87,38	0,49	**89,29**	1,77	89,23	1,90
BR_10	85,72	86,46	0,17	86,20	0,16	86,93	0,71	**88,63**	2,38	88,62	2,06
BR_11	85,85	86,76	0,20	86,99	0,31	87,62	1,47	**88,81**	3,59	88,58	3,80
BR_12	85,18	87,04	0,34	86,86	0,30	87,34	1,51	**88,66**	5,14	88,40	5,18
BR_13	85,40	85,93	0,62	85,71	0,50	86,19	3,36	**87,83**	9,28	87,13	6,53
BR_14	84,87	85,72	0,72	85,87	0,74	86,18	4,42	**87,67**	9,70	87,11	9,09
BR_15	85,41	85,19	0,87	85,31	0,89	85,70	6,39	**87,27**	16,84	86,70	12,90
Mean	85,87	86,72	0,22	87,33	0,22	87,36	1,27	89,97	3,42	89,84	2,95

*The best values appear in bold

Comparing sequences of neighbourhoods

15 × 10 problems



	VND_42531		VND_32145	
Problem	Vol.(%)	Time	Vol.(%)	Time
BR_1	**92,82**	0,08	92,54	0,28
BR_2	93,26	0,19	**93,39**	0,83
BR_3	**93,10**	0,34	91,94	1,85
BR_4	**93,73**	0,67	92,60	1,35
BR_5	**92,73**	0,98	91,60	1,73
BR_6	**92,67**	2,07	91,09	2,23
BR_7	**91,38**	2,21	90,54	3,25
BR_8	**90,56**	5,62	90,55	6,23
BR_9	**90,73**	8,18	89,76	7,03
BR_10	**89,94**	10,95	89,71	8,74
BR_11	**90,22**	16,88	88,90	12,00
BR_12	**89,88**	19,28	89,55	19,37
BR_13	**88,75**	27,25	88,60	20,82
BR_14	**88,73**	34,78	88,38	33,30
BR_15	**88,70**	59,01	88,24	29,11
Mean B1-B7	**92,81**	0,93	91,96	1,65
Mean	**91,15**	12,57	90,49	9,87

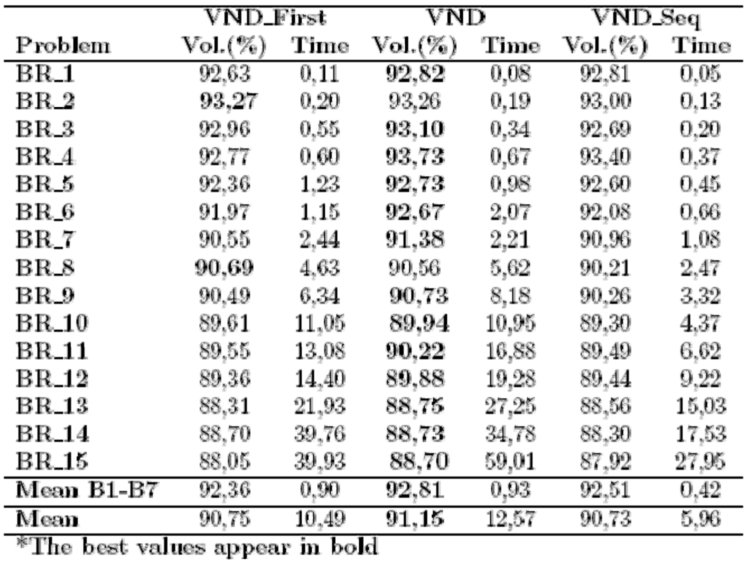
*The best values appear in bold

VND_42531 - strategy consisting on alternating neighbourhoods based on elimination and neighbourhoods based on insertion.

VND_32145 - alternative strategy of using first the simplest and fastest moves and only when they fail to produce improved solutions calling the more complex moves.

Comparing VND strategies

15 × 10 problems

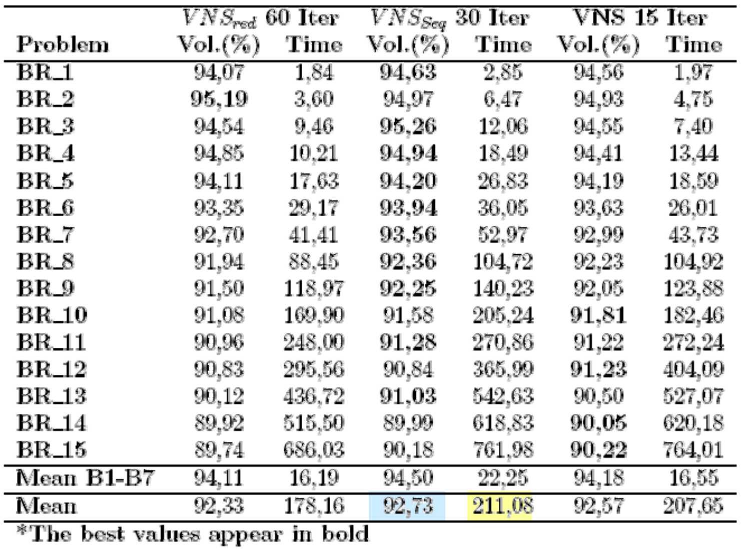


	VND_First		VND		VND_Seq	
Problem	Vol.(%)	Time	Vol.(%)	Time	Vol.(%)	Time
BR_1	92,63	0,11	**92,82**	0,08	92,81	0,05
BR_2	**93,27**	0,20	93,26	0,19	93,00	0,13
BR_3	92,96	0,55	**93,10**	0,34	92,69	0,20
BR_4	92,77	0,60	**93,73**	0,67	93,40	0,37
BR_5	92,36	1,23	**92,73**	0,98	92,60	0,45
BR_6	91,97	1,15	**92,67**	2,07	92,08	0,66
BR_7	90,55	2,44	**91,38**	2,21	90,96	1,08
BR_8	**90,69**	4,63	90,56	5,62	90,21	2,47
BR_9	90,49	6,34	**90,73**	8,18	90,26	3,32
BR_10	89,61	11,05	**89,94**	10,95	89,30	4,37
BR_11	89,55	13,08	**90,22**	16,88	89,49	6,62
BR_12	89,36	14,40	**89,88**	19,28	89,44	9,22
BR_13	88,31	21,93	**88,75**	27,25	88,56	15,03
BR_14	88,70	39,76	**88,73**	34,78	88,30	17,53
BR_15	88,05	39,93	**88,70**	59,01	87,92	27,95
Mean B1-B7	92,36	0,90	**92,81**	0,93	92,51	0,42
Mean	90,75	10,49	**91,15**	12,57	90,73	5,96

*The best values appear in bold

Comparing VNS alternatives

15 × 10 problems



	VNS_{red} 60 Iter		VNS_{Seq} 30 Iter		VNS 15 Iter	
Problem	Vol.(%)	Time	Vol.(%)	Time	Vol.(%)	Time
BR_1	94,07	1,84	**94,63**	2,85	94,56	1,97
BR_2	**95,19**	3,60	94,97	6,47	94,93	4,75
BR_3	94,54	9,46	**95,26**	12,06	94,55	7,40
BR_4	94,85	10,21	**94,94**	18,49	94,41	13,44
BR_5	94,11	17,63	**94,20**	26,83	94,19	18,59
BR_6	93,35	29,17	**93,94**	36,05	93,63	26,01
BR_7	92,70	41,41	**93,56**	52,97	92,99	43,73
BR_8	91,94	88,45	**92,36**	104,72	92,23	104,92
BR_9	91,50	118,97	**92,25**	140,23	92,05	123,88
BR_10	91,08	169,90	91,58	205,24	**91,81**	182,46
BR_11	90,96	248,00	**91,28**	270,86	91,22	272,24
BR_12	90,83	295,56	90,84	365,99	**91,23**	404,09
BR_13	90,12	436,72	**91,03**	542,63	90,50	527,07
BR_14	89,92	515,50	89,99	618,83	**90,05**	620,18
BR_15	89,74	686,03	90,18	761,98	**90,22**	764,01
Mean B1-B7	94,11	16,19	94,50	22,25	94,18	16,55
Mean	92,33	178,16	92,73	211,08	92,57	207,65

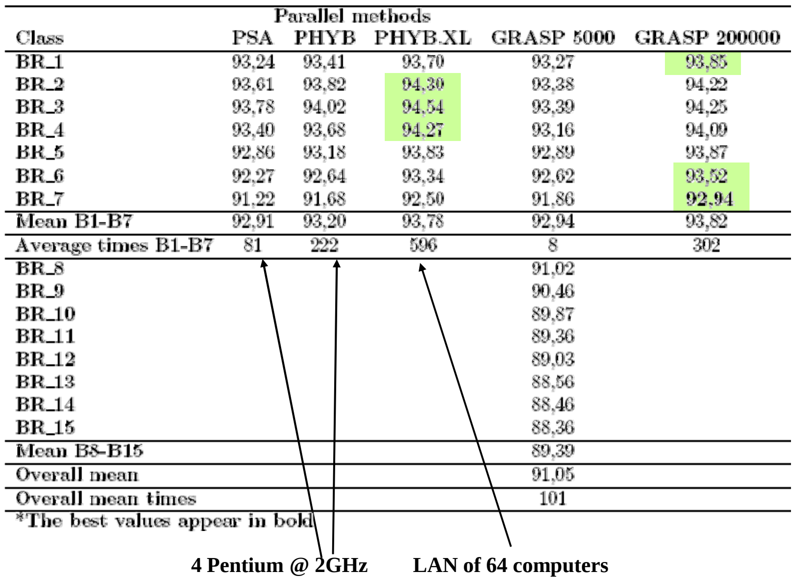
*The best values appear in bold

Full computational tests

VNS algorithm final parameterization

- Stopping criterion
  - we let it run for a minimum of 30 iterations and then go on until no improvement is found in the last 5 iterations or a maximum of 60 iterations is reached.
- Shaking
  - N1 to N5 + N6 (stronger shaking movement)
- VND
  - N1 to N5
  - alternating neighbourhoods based on elimination and neighbourhoods based on insertion
  - sequential VND algorithm
- The sets of possible neighbours are very large and therefore we do not fully explore them.
  - At each iteration we explore only 1000 moves for the three first neighbourhoods and 100 for the last ones.

Comparison of algorithms

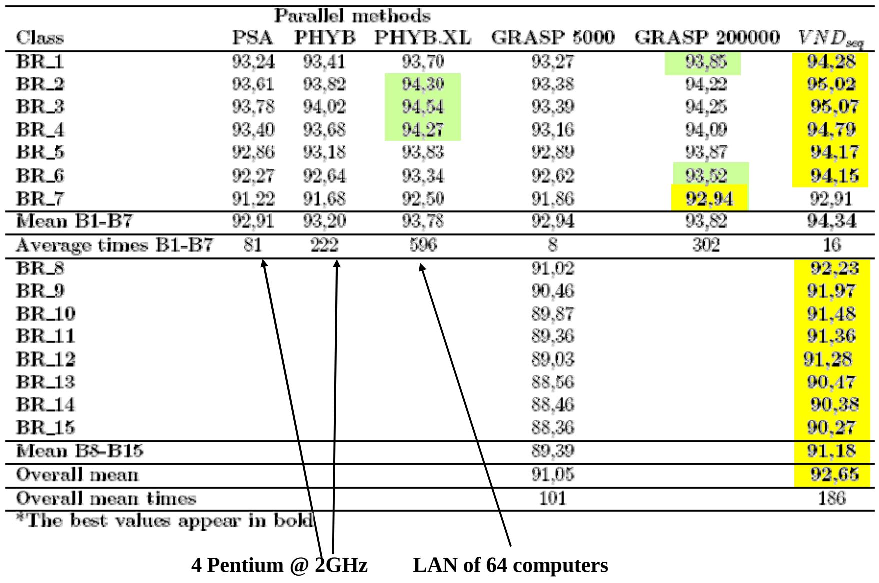


	Parallel methods				
Class	PSA	PHYB	PHYB.XL	GRASP 5000	GRASP 200000
BR_1	93,24	93,41	93,70	93,27	93,85
BR_2	93,61	93,82	94,30	93,38	94,22
BR_3	93,78	94,02	94,54	93,39	94,25
BR_4	93,40	93,68	94,27	93,16	94,09
BR_5	92,86	93,18	93,83	92,89	93,87
BR_6	92,27	92,64	93,34	92,62	93,52
BR_7	91,22	91,68	92,50	91,86	**92,94**
Mean B1-B7	92,91	93,20	93,78	92,94	93,82
Average times B1-B7	81	222	596	8	302
BR_8				91,02	
BR_9				90,46	
BR_10				89,87	
BR_11				89,36	
BR_12				89,03	
BR_13				88,56	
BR_14				88,46	
BR_15				88,36	
Mean B8-B15				89,39	
Overall mean				91,05	
Overall mean times				101	

*The best values appear in bold

4 Pentium @ 2GHz **LAN of 64 computers**

Comparison of algorithms



	Parallel methods					
Class	PSA	PHYB	PHYB.XL	GRASP 5000	GRASP 200000	VND_{seq}
BR_1	93,24	93,41	93,70	93,27	93,85	**94,28**
BR_2	93,61	93,82	94,30	93,38	94,22	**95,02**
BR_3	93,78	94,02	94,54	93,39	94,25	**95,07**
BR_4	93,40	93,68	94,27	93,16	94,09	**94,79**
BR_5	92,86	93,18	93,83	92,89	93,87	**94,17**
BR_6	92,27	92,64	93,34	92,62	93,52	**94,15**
BR_7	91,22	91,68	92,50	91,86	**92,94**	92,91
Mean B1-B7	92,91	93,20	93,78	92,94	93,82	**94,34**
Average times B1-B7	81	222	596	8	302	16
BR_8				91,02		**92,23**
BR_9				90,46		**91,97**
BR_10				89,87		**91,48**
BR_11				89,36		**91,36**
BR_12				89,03		**91,28**
BR_13				88,56		**90,47**
BR_14				88,46		**90,38**
BR_15				88,36		**90,27**
Mean B8-B15				89,39		**91,18**
Overall mean				91,05		**92,65**
Overall mean times				101		186

*The best values appear in bold

4 Pentium @ 2GHz **LAN of 64 computers**

Conclusions

- Very good results (space utilization) for all class of problems, ranging from weakly heterogeneous to strongly heterogeneous
  - Use of maximal-spaces
  - 8 corners strategy, keeping empty spaces together, in the middle of the container
- We have proposed several new neighbourhoods based on the elimination of layers, insertion of columns or boxes and a move based on emptying a region of the container
- Our experimentation shows that the VNS methodology competes favourably with the best algorithms for the container loading problem
- The algorithm has potential to accomodate cargo stability issues and multi-drop loads (4 corners and 2 corners strategies).

EURO XXII PRAGUE

22nd European Conference on Operational Research, July 8-11, 2007

Neighbourhood structures for the container loading problem: a VNS implementation

José Fernando Oliveira (1), Francisco Parreño (2), Ramon Alvarez-Valdes (3) and José Manuel Tamarit (3)

(1) University of Porto – Faculty of Engineering and INESC Porto
(2) University of Castilla-La Mancha – Department of Computer Science
(3) University of Valencia – Department of Statistics and Operations Research

Partially supported by FCT, POSC and FEDER (POSC/EEA-SRI/57367/2004 (APIS))
and Ministerio de Educacion e Ciencia (DPI2005-04796 – OPEC

Comparison with other algorithms

- Algorithms that pack the boxes so that they are completely supported by other boxes:
  - H_BR: a constructive algorithm by Bischoff and Ratcliff;
  - H_B_al: a constructive algorithm by Bischoff, Janetz and Ratcliff;
  - H_B: a heuristic approach by Bischoff;
  - GA_GB: a genetic algorithm by Gehring and Bortfeldt;
  - TS_BG: a tabu search approach by Bortfeldt and Gehring;
  - H_E: a greedy constructive algorithm with an improvement phase by Eley;
  - G_M: a GRASP approach by Moura and Oliveira.

Comparison with other algortihms

- Algorithms that do not consider that constraint:
  - HGA_BG: a hybrid genetic algorithm by Gehring and Bortfeldt;
  - PGA_GB: a parallel genetic algorithm by Gehring and Bortfeldt;
  - PTS_B_al: a parallel tabu search algorithm by Bortfeldt, Gehring and Mack;
  - TSA_MB: a tabu search algorithm by Mack, Bortfeldt and Gehring;
  - SA_MB: a simulated annealing algorithm by Mack, Bortfeldt and Gehring;
  - HYB_MB: a hybrid algorithm by Mack, Bortfeldt and Gehring;
  - PTSA_MB: a parallel tabu search algorithm by Mack, Bortfeldt and Gehring;
  - PSA_MB: a parallel simulated annealing algorithm byMack, Bortfeldt and Gehring;
  - PHYB_MB: a parallel hybrid algorithm byMack, Bortfeldt and Gehring;
  - PHYB_XL_MB: a massive parallel hybrid algorithm by Mack, Bortfeldt and Gehring;