Greedy methods for Bin Packing Benchmark problems

Gyorgy Dosa,
University of Pannonia (Veszprem, Hungary)

Joint work with
Gyula Abraham, Tibor Dulai, Agnes Werner-Stark, Zsolt Tuza

Greedy methods for Bin Packing Benchmark problems

We consider the Bin Packing Problem (BP) from the practical side.
No proof.
If someone would like to read proofs, let me advertise the next:

János Balogh, József Békési, György Dósa, Leah Epstein, Asaf Levin: A New and Improved Algorithm for Online Bin Packing. ESA 2018: 5:1-5:14 (the full version is on Archive, 46 pages long)
The currently best upper bound for online bin packing, 1.57829.
(The currently best LB is about 1.54278.)

György Dósa, Rongheng Li, Xin Han, Zsolt Tuza: Tight absolute bound for First Fit Decreasing bin-packing: FFD(I) ≤ 11/9 OPT(L) + 6/9. Theor. Comput. Sci. 510: 13-61 (2013) (49 pages long)
How big can be FFD(I), if OPT(I) is given, the tight result.

BP: solution techniques

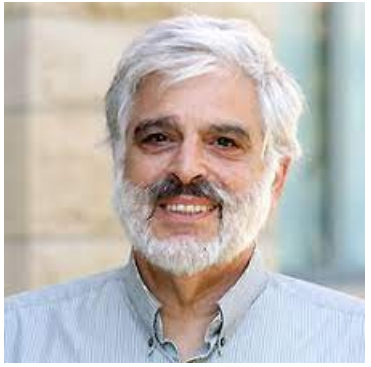
- Approximation algorithms
- Exact algorithms (and lower bounds)
- Meta-heuristic algorithms
- The metaheuristic Zoo
- We will discuss about a „new” paradigm: pre-processing. Solving (optimally) as many as possible, with a Greedy-type algorithm

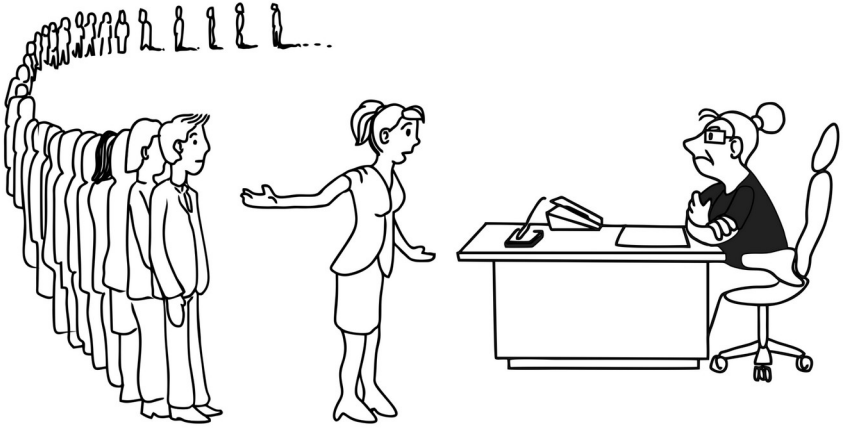
Approximation algorithms

Fit algorithms: First Fit, Best Fit, Worst Fit,…

- First Fit Decreasing,…
- Harmonic algorithm
- And others
- Johnson, Garey, Coffman, Ulmann,…

,…,





"I can't find an efficient algorithm, but neither can all these famous people."

$$FF(I) \leq 1.7 * OPT(I) + 0$$

Exact algorithms

Means that surely finds optimal solution (possibly needs much time)

Martello, S., and Toth, P., Bin-packing problem. In Knapsack Problems: Algorithms and Computer Implementations. Wiley. chapter 8, 221–245.

Martello, S., and Toth, P., 1990b. Lower bounds and reduction procedures for the bin packing problem. Discrete Applied Mathematics 28:59–70.

Delorme M., Iori, M., Martello, S., Bin packing and cutting stock problems: Mathematical models and exact algorithms, European Journal of Operational Research, 255 (1), 2016, 1-20.

Fekete, S., Schepers, J., New classes of fast lower bounds for bin packing problems. Math. Program. 91, 11–31 (2001).

(Meta)heuristic algorithms

Tabu search Lodi, A., Martello, S. & Vigo, D. TSpack: A Unified Tabu Search Code for Multi-Dimensional Bin Packing Problems. Ann Oper Res 131, 203–213 (2004)

Genetic Algorithm Bennell, J.A., Lee, L.S., Potts, C.N., A genetic algorithm for two-dimensional bin packing with due dates, International Journal of Production Economics, Volume 145, Issue 2, 2013, Pages 547-560

Simulated Annealing Kämpke, T. Simulated annealing: Use of a new tool in bin packing. Ann Oper Res 16, 327–332 (1988)

Ant Colony Levine, J., and Ducatelle, F., (2004) Ant colony optimization and local search for bin packing and cutting stock problems, Journal of the Operational Research Society, 55:7, 705-716

Hybrids Falkenauer, E. A hybrid grouping genetic algorithm for bin packing. J Heuristics 2, 5–30 (1996).

The Zoo

Cuckoo

Kangaroo

Whale

Bee

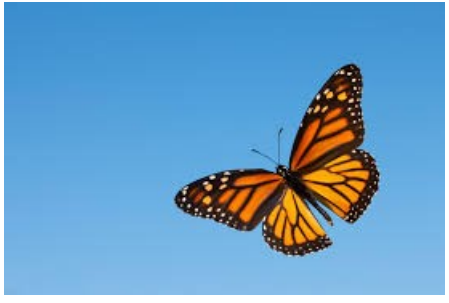
Ant colony

African buffalo

Bat(man)

Flying squirrel

Fly, butterfly



a „new” paradigm: pre-processing

We apply a GREEDY algorithm

Greedy means here:

- We do not want to solve ALL instances optimally
- But we want to solve AS MANY as possible, optimally
- Instead of packing „well” all bins at the same time, we pack well one bin, then a next bin, and so on. So we deal only with packing well one bin at any time.
- If a bin is packed, then we pack „well” the next bin. A previous bin (and its packing) is not changed later.

Benchmarks

- See: Unibo homepage:
- http://or.dei.unibo.it/library/bpplib
- We consider (for today) two classes:
  - Schwerin (100+100 instances)
  - Falkenauer U (20+20+20+20 instances)

Schwerin class:

C=1000 (the bin size)

Items are drawn from [150;200], by uniform distribution

The number of items is

- n=100 (Schwerin 1 subclass) here OPT=LB=18
- n=120 (Schwerin 2 subclass) here OPT=LB=21 or 22

What to do with Schwerin

(What shall we do with the Schwerin instance,

What shall we do with the Schwerin instance,

Early in the afternoon?)

Observation:

- no 7 items fit into a bin (sizes are between 150 and 200, C=1000)

- any 5 items fit into a bin

Conclusion: 8 bins will contain 5 items, and 10 bins will contain 6 items, we can put the 40 biggest items into 8 bins, and we need to think only about the remained 60 items (to pack them into 10 bins)!!!

Not 100 items, only 60 items to think about!!!

Already simplified!

So, remained 60 items, to carefully pack them, 6 into a bin.

A typical instance looks as follows (Schwerin1_BPP1, after packing the 40 biggest items: denoted by bold)

200 200 200 199 198 198 197 197 194 194 193 192 191 191 191

190 190 189 188 188 187 187 186 185 185 185 185 184 184 184

183 183 183 182 182 182 181 181 180 179 179 179 179 178 177

177 177 177 175 174 173 173 172 171 171 171 170 170 169 169

169 167 167 165 165 164 163 163 163 163 162 161 160 160 159

158 158 158 157 156 156 156 156 156 156 155 155 155 154 154

153 152 152 152 151 151 150 150 150 150

What shall we do:

Carefully fill one bin at a time. Two objectives:

- We pack several large items, as these can cause more trouble (if remain in the end)
- We also want to fill well the bin

So, we balance between these two objectives.

Let us perform as follows:

Definition of k

We look for k, such that:

The biggest k items, and the smallest 6-k items fit (into a bin).

(In the beginning, k is typically 2 , later it is growing).

So e.g., the 2 biggest items, and the 4 smallest items fit.

Then we pack: the 2 biggest items, and 4 further, carefully chosen items, to fill the bin as much as possible!

Generally: the k biggest items, and 6-k further, carefully chosen items, to fill the bin as much as possible!

We use some slave algorithms for the choice.

Packing the 9-th bin, k=3

200 200 200 199 198 198 197 197 194 194 193 192 191 191 191
190 190 189 188 188 187 187 186 185 185 185 185 184 184 184
183 183 183 182 182 182 181 181 180 179 179 179 179 178 177
177 177 177 175 174 173 173 172 171 171 171 170 170 169 169
169 167 167 165 165 164 163 163 163 163 162 161 160 160 159
158 158 158 157 156 156 156 156 156 156 155 155 155 154 154
153 152 152 152 151 151 150 150 150 150

Level of the bin: 1000

Packing the 10-th bin, k=3

200 200 200 199 198 198 197 197 194 194 193 192 191 191 191

190 190 189 188 188 187 187 186 185 185 185 185 184 184 184

183 183 183 182 182 182 181 181 180 179 179 179 179 178 177

177 177 177 175 174 173 173 172 171 171 171 170 170 169 169

169 167 167 165 165 164 163 163 163 163 162 161 160 160 159

158 158 158 157 156 156 156 156 156 156 **155** 155 155 **154 154**

153 152 152 152 151 151 150 150 150 150

Level of the bin: 1000

Packing the 11-th bin, k=3

200 200 200 199 198 198 197 197 194 194 193 192 191 191 191
190 190 189 188 188 187 187 186 185 185 185 185 184 184 184
183 183 183 182 182 182 181 181 180 179 179 179 179 178 177
177 177 177 175 174 173 173 172 171 171 171 170 170 169 169
169 167 167 165 165 164 163 163 163 163 162 161 160 160 159
158 158 158 157 **156 156 156** 156 156 156 **155** 155 155 **154 154**
153 152 152 152 151 151 150 150 150 150

Level of the bin: 1000

Packing the 12-th bin, k=3

200 200 200 199 198 198 197 197 194 194 193 192 191 191 191
190 190 189 188 188 187 187 186 185 185 185 185 184 184 184
183 183 183 182 182 182 181 181 180 179 179 179 179 178 177
177 177 177 175 174 173 173 172 171 171 171 170 170 169 169
169 167 167 165 165 164 163 163 163 163 162 161 160 160 159
158 158 158 **157 156 156 156 156** 156 156 **155** 155 155 **154 154**
153 152 152 152 151 151 150 150 150 150

Level of the bin: 1000

Packing the 13-th bin, k=3

200 200 200 199 198 198 197 197 194 194 193 192 191 191 191
190 190 189 188 188 187 187 186 185 185 185 185 184 184 184
183 183 183 182 182 182 181 181 180 179 179 179 179 178 177
177 177 177 175 174 173 173 172 171 171 171 170 170 169 169
169 167 167 165 165 164 163 163 163 163 162 **161 160** 160 **159**
158 158 158 **157 156 156 156 156** 156 156 **155** 155 155 **154 154**
153 152 152 152 151 151 150 150 150 150

Level of the bin: 1000

Packing the 14-th bin, k=3

200 200 200 199 198 198 197 197 194 194 193 192 191 191 191
190 190 189 188 188 187 187 186 185 185 185 185 184 184 184
183 183 183 182 182 182 181 181 180 179 179 179 179 178 177
177 177 177 175 174 173 173 172 171 171 171 170 170 169 169
169 167 167 165 165 164 **163 163** 163 163 162 **161 160 160 159**
158 158 158 **157 156 156 156 156** 156 156 **155** 155 155 **154 154**
153 152 152 152 151 151 150 150 150 150

Level of the bin: 1000

Packing the 15-th bin, k=3

200 200 200 199 198 198 197 197 194 194 193 192 191 191 191
190 190 189 188 188 187 187 186 185 185 185 185 184 184 184
183 183 183 182 182 182 181 181 180 179 179 179 179 178 177
177 177 177 175 174 173 173 172 171 171 171 170 170 169 169
169 167 167 165 165 **164 163 163 163** 163 **162 161 160 160 159**
158 158 158 **157 156 156 156 156** 156 156 **155** 155 155 **154 154**
153 152 152 152 151 151 150 150 150 150

Level of the bin: 1000

Packing the 16-th bin, k=6

200 200 200 199 198 198 197 197 194 194 193 192 191 191 191
190 190 189 188 188 187 187 186 185 185 185 185 184 184 184
183 183 183 182 182 182 181 181 180 179 179 179 179 178 177
177 177 177 175 174 173 173 172 171 171 171 170 170 169 169
169 167 167 **165 165 164 163 163 163 163 162 161 160 160 159**
158 158 158 **157 156 156 156 156** 156 156 **155** 155 155 **154 154**
153 152 152 152 151 151 150 150 150 150

Level of the bin: 962 (but we are already OK)

Packing the 17-th bin, k=6

200 200 200 199 198 198 197 197 194 194 193 192 191 191 191
190 190 189 188 188 187 187 186 185 185 185 185 184 184 184
183 183 183 182 182 182 181 181 180 179 179 179 179 178 177
177 177 177 175 174 173 173 172 171 171 171 170 170 169 169
169 167 167 165 165 164 163 163 163 163 162 161 160 160 159
158 158 158 157 156 156 156 156 156 156 155 155 155 **154 154**
153 152 152 152 151 151 150 150 150 150

Level of the bin: 919

Packing the 18-th bin, k=6

200 200 200 199 198 198 197 197 194 194 193 192 191 191 191
190 190 189 188 188 187 187 186 185 185 185 185 184 184 184
183 183 183 182 182 182 181 181 180 179 179 179 179 178 177
177 177 177 175 174 173 173 172 171 171 171 170 170 169 169
169 167 167 165 165 164 163 163 163 163 162 161 160 160 159
158 158 158 157 156 156 156 156 156 156 155 155 155 154 154
153 152 152 152 151 151 150 150 150 150

Level of the bin: 902

Summarizing the results (of Schwerin class)

Schwerin 1 subclass:

Optimal solution got in 100 cases out of 100 cases.

Schwerin 2 subclass:

- Optimal solution got in 98 cases out of the 100 cases.
- With a little modification (only $2 \leq k \leq 3$ is allowed) optimal solution in 100 cases out of the 100 cases.

Altogether: 100+100 instances, optimal solution got in all 200 instances.

If n grows?

Question: What can we say, if n (the number of items) grows?

(Short answer: I do not know.)

Possible answers:

A: the case is the same (this kind of algorithm works, no matter how big n is)

B: the case is similar, the algorithm works, but a little bit worse

C: this algorithm does not work, but some other (still very simple) algorithm works

D: disaster: no simple algorithm can work

If n grows?

Question: What can we say, if n (the number of items) grows?

A: the case is the same (this kind of algorithm works, no matter how big n is)

B: the case is similar, the algorithm works, but a little bit worse

C: this algorithm does not work, but some other (still very simple) algorithm works

D: disaster: no simple algorithm can work

Would you like to vote? (Send A, B, C, D or E)

Falkenauer U class

Here there are 20+20+20+20 instances.

C=150 (the bin size)

Items are drawn from [20,100]

The number of items is 120 or 250 or 500 or 1000.

We describe the algorithm for the U_120 subclass, for the other subclasses the case is similar.

Immediately we can see, that the case is more difficult.

Reason: Bigger diversity. The ratio of biggest and smallest item size is

100/20=5 now, and it was 200/150=1.33 in case of Schwerin type. This makes the problem harder.

Simplification is possible again!!!

Observation: If we have two items, the sum of their sizes is just C, we can pack them „calmly” into a dedicated bin for these two items, we do not lose from the optimality.

We will need two easy definitions:

Big item: bigger than C/2

Reserve: LB*C-total size (Usually OPT=LB= total size/C rounded up)

Observation: If we have much reserve, we do not need to be too restrictive, we can allow to lose some reserve during packing

(i.e., the level is e.g., C-1 or C-2, it is still fine)

We define three slave algorithms

We need three simple slave algorithms

The items are already sorted by decreasing size

The slave algorithms:

- -pack well pairs
- -pack well triplets
- -pack well quadruplets

Good pairs

The first slave algorithm (K: the targeted level)

Pair(K , r)

1. Let i be the biggest unpacked item.
2. If size(i) < K/2 or the current value of the reserve is res < r, then return.
3. If there exists a subsequent item j such that

   size(i)+size(j) = K

   pack i and j into a new bin, delete these items from the set of unpacked items, reduce res by (C - K), and go to Step 1.
4. Let i be the next unpacked item (i = i + 1), and go to Step 2.

Good triplets

The second slave algorithm

Triplet(K , r)

1. Let i be the biggest unpacked item.
2. If size(i) < K/3 or the current value of the reserve is res < r, then return.
3. If there exist two subsequent items j and k such that

   size(i)+size(j)+size(k) = K,

   pack i, j and k into a new bin, delete these items from the set of unpacked items, reduce res by (C - K), and go to Step 1.
4. Let i be the next unpacked item (i = i + 1), and go to Step 2.

Good quadruplets

Quadret (K , r) The third slave algorithm

1. Let i and j be the two biggest unpacked items.
2. If size(i)+size(j) < K/2 or the current value of the reserve is res < r, then return.
3. If there exist two subsequent items k and l such that

   size(i)+size(j)+ size(k)+size(l)= K,

   pack i, j, k and l into a new bin, delete these items from the set of unpacked items, reduce res by (C - K), and go to Step 1.
4. Let i = j, let j be the next unpacked item, and go to Step 2.

The master algorithm for the Falkenauer U_120 subclass

ALG FU

1. a, Call Pair(150, 0)
   b, Call Pair(149, 10)
   c, Call Pair(148, 15)
   d, Call Pair(147, 30)
   e, Call Pair(146, 30)
   f, Call Pair(145, 30)

2. While there exists unpacked big item
   a, Call Triplet(150, 0)
   b, Call Triplet(149, 5)
   c, Call Triplet(148, 10)
   d, Call Triplet(147, 15)
   e, Call Triplet(146, 20)
   f, Call Triplet(145, 25)

otherwise (there is no unpacked big item)
   a, Call Triplet(150, 0)
   b, Call Triplet(149, 5)
   c, Call Triplet(148, 15)
   d, Call Triplet(147, 30)
   e, Call Triplet(146, 30)
   f, Call Triplet(145, 30)

3. If there is not unpacked big item
   a, Call Quadret(150, 0)
   b, Call Quadret(149, 0)
   c, Call Quadret(148, 0)

4. Pack all unpacked items by algorithm FFD.

Falkenauer U_120_00, Bin 1

98 98 98 96 96 94 93 93 92 91 91 90 87 86 85

85 84 84 84 84 84 83 83 82 82 81 80 80 80 79

79 78 78 78 78 76 74 74 73 73 73 73 72 71 70

70 70 69 69 69 67 66 64 62 62 60 60 59 58 58

58 57 57 57 57 55 55 55 50 49 49 49 47 46 46

45 45 44 44 43 43 43 43 42 42 42 42 42 41 41

41 39 39 38 38 38 37 36 36 36 35 33 33 33 32

32 30 30 30 29 28 27 27 26 25 25 24 23 23 20

Step 1a, Load=150, res=122

Falkenauer U_120_00, Bin 15

98 98 98 96 96 94 **93 93 92 91** 91 **90** 87 **86** 85

85 **84** 84 84 84 84 **83** 83 82 82 **81 80 80 80 79**

79 **78** 78 78 78 76 74 74 73 73 73 73 **72 71 70**

70 70 69 69 69 **67 66 64** 62 62 **60** 60 **59 58** 58

58 **57 57** 57 57 55 55 55 50 49 49 49 47 46 46

45 45 44 44 43 43 43 43 42 42 42 42 42 41 41

41 39 39 38 38 38 37 36 36 36 35 33 33 33 32

32 30 30 30 29 28 27 27 26 25 25 24 23 23 20

Step 1a, Load=150, res=122

15 bins already filled completely with „good pairs”. Remained 90 items from the 120 items.

Falkenauer U_120_00, Bin 16

98 98 98 96 96 94 **93 93 92 91** 91 **90** 87 **86** 85

85 **84** 84 84 84 84 **83** 83 82 82 **81 80 80 80 79**

79 **78** 78 78 78 **76 74** 74 73 73 73 73 **72 71 70**

70 70 69 69 69 **67 66 64** 62 62 **60** 60 **59 58** 58

58 **57 57** 57 57 55 55 55 50 49 49 49 47 46 46

45 45 44 44 43 43 43 43 42 42 42 42 42 41 41

41 39 39 38 38 38 37 36 36 36 35 33 33 33 32

32 30 30 30 29 28 27 27 26 25 25 24 23 23 20

Step 1b, Load=149, res=121

Falkenauer U_120_00, Bin 18

98 98 98 96 96 **94 93 93 92 91 91 90** 87 **86** 85

85 **84** 84 84 84 84 **83** 83 82 82 **81 80 80 80 79**

79 **78** 78 78 78 **76 74** 74 73 73 73 73 **72 71 70**

70 70 69 69 69 **67 66 64** 62 62 **60** 60 **59 58 58**

58 **57 57** 57 57 **55** 55 55 50 49 49 49 47 46 46

45 45 44 44 43 43 43 43 42 42 42 42 42 41 41

41 39 39 38 38 38 37 36 36 36 35 33 33 33 32

32 30 30 30 29 28 27 27 26 25 25 24 23 23 20

Step 1b, Load=149, res=119

Falkenauer U_120_00, Bin 19

98 98 98 96 96 **94 93 93 92 91 91 90 87 86** 85

85 **84** 84 84 84 84 **83** 83 82 82 **81 80 80 80 79**

79 **78** 78 78 78 **76 74** 74 73 73 73 73 **72 71 70**

70 70 69 69 69 **67 66 64 62** 62 **60** 60 **59 58 58**

58 **57 57** 57 57 **55** 55 55 50 49 49 49 47 46 46

45 45 44 44 43 43 43 43 42 42 42 42 42 41 41

41 39 39 38 38 38 37 36 36 36 35 33 33 33 32

32 30 30 30 29 28 27 27 26 25 25 24 23 23 20

Step 1c, Load=148, res=117

Falkenauer U_120_00, Bin 20

98 98 98 96 96 **94** **93** **93** **92** **91** **91** **90** **87** **86** 85

85 **84** 84 84 84 84 **83** 83 82 82 **81** **80** **80** **80** **79**

79 **78** 78 78 78 **76** **74** 74 73 73 73 73 **72** **71** **70**

70 **70** **69** 69 69 **67** **66** **64** **62** 62 **60** 60 **59** **58** **58**

58 **57** **57** 57 57 **55** 55 55 **50** 49 49 49 47 46 46

45 45 44 44 43 43 43 43 42 42 42 42 42 41 41

41 39 39 38 38 38 37 36 36 36 35 33 33 33 32

32 30 30 30 29 28 27 27 26 25 25 24 23 23 20

Step 1c, Load=148, res=115

Falkenauer U_120_00, Bin 21

98 98 98 96 96 **94** **93** **93** **92** **91** **91** **90** **87** **86** 85

85 **84** 84 84 84 84 **83** 83 82 82 **81** **80** **80** **80** **79**

79 **78** 78 78 78 **76** **74** 74 73 73 73 73 **72** **71** **70**

70 **70** **69** **69** 69 **67** **66** **64** **62** 62 **60** 60 **59** **58** **58**

58 **57** **57** 57 57 **55** 55 55 **50** 49 49 49 47 46 46

45 45 44 44 43 43 43 43 42 42 42 42 42 41 41

41 39 39 38 38 38 37 36 36 36 35 33 33 33 32

32 30 30 30 29 28 27 27 26 25 25 24 23 23 20

Step 1d, Load=147, res=112

Falkenauer U_120_00, Bin 25

98 98 98 96 96 **94 93 93 92 91 91 90 87 86 85**

85 **84** 84 84 84 84 **83** 83 82 82 **81 80 80 80 79**

79 78 78 78 78 **76 74** 74 73 73 73 73 **72 71 70**

70 70 69 69 69 67 66 64 62 62 60 60 **59 58 58**

58 **57 57** 57 57 **55** 55 55 **50 49 49** 49 47 46 46

45 45 44 44 43 43 43 43 42 42 42 42 42 41 41

41 39 39 38 38 38 37 36 36 36 35 33 33 33 32

32 30 30 30 29 28 27 27 26 25 25 24 23 23 20

Step 1d, Load=147, res=100

Falkenauer U_120_00, Bin 26

98 98 98 96 96 **94 93 93 92 91 91 90 87 86 85**

85 **84** 84 84 84 84 **83** 83 82 82 **81 80 80 80 79**

79 78 78 78 78 **76 74 74 73** 73 73 73 **72 71 70**

70 70 69 69 69 67 66 64 62 62 60 60 **59 58 58**

58 **57 57** 57 57 **55** 55 55 **50 49 49** 49 47 46 46

45 45 44 44 43 43 43 43 42 42 42 42 42 41 41

41 39 39 38 38 38 37 36 36 36 35 33 33 33 32

32 30 30 30 29 28 27 27 26 25 25 24 23 23 20

Step 1e, Load=146, res=96

Falkenauer U_120_00, Bin 27

98 98 98 96 96 **94 93 93 92 91 91 90 87 86 85**

85 **84** 84 84 84 84 **83** 83 82 82 **81 80 80 80 79**

79 78 78 78 78 **76 74 74 73 73 73** 73 **72 71 70**

70 70 69 69 69 67 66 64 62 62 60 60 **59 58 58**

58 **57 57** 57 57 **55** 55 55 **50 49 49** 49 47 46 46

45 45 44 44 43 43 43 43 42 42 42 42 42 41 41

41 39 39 38 38 38 37 36 36 36 35 33 33 33 32

32 30 30 30 29 28 27 27 26 25 25 24 23 23 20

Step 1f, Load=145, res=91

Falkenauer U_120_00, Bin 28

98 98 98 96 96 **94 93 93 92 91 91 90 87 86 85**

85 **84** 84 84 84 84 **83** 83 82 82 **81 80 80 80 79**

79 78 78 78 78 **76 74 74 73 73 73** 73 **72 71 70**

70 70 69 69 69 67 66 64 62 62 60 60 **59 58 58**

58 **57 57** 57 57 **55** 55 55 **50 49 49 49** 47 46 46

45 45 44 44 43 43 43 43 42 42 42 42 42 41 41

41 39 39 38 38 38 37 36 36 36 35 33 33 33 32

32 30 30 30 29 28 27 27 26 25 25 24 23 23 20

Step 1f, Load=145, res=86

The last occasion in Step1 (packing 2 items)

Falkenauer U_120_00, Bin 29

98 98 98 96 96 **94 93 93 92 91 91 90 87 86 85**

85 84 84 84 84 84 **83** 83 82 82 **81 80 80 80 79**

79 78 78 78 78 **76 74 74 73 73 73** 73 **72 71 70**

70 70 69 69 69 67 66 64 62 62 60 60 59 58 58

58 **57 57** 57 57 **55** 55 55 **50 49 49 49** 47 46 46

45 45 44 44 43 43 43 43 42 42 42 42 42 41 41

41 39 39 38 38 38 37 36 36 36 35 33 33 33 32

32 30 30 30 29 28 27 27 26 25 25 24 23 23 20

Step 2a, Load=150, res=86

The first step in Step2 (packing 3 items)

Falkenauer U_120_00, Bin 41

98 98 98 96 96 94 93 93 92 91 91 90 87 86 85

85 84 84 84 84 84 83 83 82 82 81 80 80 80 79

79 78 78 78 78 76 74 74 73 73 73 73 72 71 70

70 70 69 69 69 67 66 64 62 62 60 60 59 58 58

58 57 57 57 57 **55** 55 55 **50 49 49 49 47 46** 46

45 45 **44** 44 **43 43** 43 43 **42 42** 42 42 42 **41 41**

41 39 39 38 38 38 37 36 36 36 **35** 33 33 33 32

32 **30 30** 30 29 **28 27 27 26 25 25 24 23 23 20**

Step 2a, Load=150, res=86

The last bin where Step 2a is performed: 3 items are packed with no losing reserve

Falkenauer U_120_00, Bin 42

98 98 98 96 96 94 93 93 92 91 91 90 87 86 85

85 84 84 84 84 84 83 83 82 82 81 80 80 80 79

79 78 78 78 78 76 74 74 73 73 73 73 72 71 70

70 70 69 69 69 67 66 64 62 62 60 60 59 58 58

58 57 57 57 57 55 55 55 **50 49 49 49 47 46** 46

45 45 **44** 44 **43 43** 43 43 **42 42** 42 42 42 **41 41**

41 39 39 **38** 38 38 37 36 36 36 **35** 33 33 33 32

32 **30 30** 30 29 **28 27 27 26 25 25 24 23 23 20**

Step 2e, Load=146, res=82

The last bin where 3 items are packed (Step 2e)

Falkenauer U_120_00, Bin 43

98 98 98 96 96 94 93 93 92 91 91 90 87 86 85

85 84 84 84 84 84 83 83 82 82 81 80 80 80 79

79 78 78 78 78 76 74 74 73 73 73 73 72 71 70

70 70 69 69 69 67 66 64 62 62 60 60 59 58 58

58 57 57 57 57 55 55 55 50 49 49 49 47 46 46

45 45 44 44 **43 43** 43 43 **42 42** 42 42 42 **41 41**

41 39 39 **38** 38 38 37 36 36 36 **35** 33 33 33 32

32 **30 30** 30 29 **28 27 27 26 25 25 24 23 23 20**

Step 3a, Load=150, res=82

The first occasion when 4 items are packed (Step 3)

Falkenauer U_120_00, Bin 44

98 98 98 96 96 94 93 93 92 91 91 90 87 86 85

85 84 84 84 84 84 83 83 82 82 81 80 80 80 79

79 78 78 78 78 76 74 74 73 73 73 73 72 71 70

70 70 69 69 69 67 66 64 62 62 60 60 59 58 58

58 57 57 57 57 55 55 55 50 49 49 49 47 46 46

45 45 44 44 43 43 43 43 **42 42** 42 42 42 **41 41**

41 39 39 **38** 38 38 37 36 36 36 **35 33** 33 33 32

32 **30 30 30** 29 **28 27 27 26 25 25 24 23 23 20**

Step 3a, Load=150, res=82

Still Step 3

Falkenauer U_120_00, Bin 45

98 98 98 96 96 94 93 93 92 91 91 90 87 86 85

85 84 84 84 84 84 83 83 82 82 81 80 80 80 79

79 78 78 78 78 76 74 74 73 73 73 73 72 71 70

70 70 69 69 69 67 66 64 62 62 60 60 59 58 58

58 57 57 57 57 55 55 55 50 49 49 49 47 46 46

45 45 44 44 43 43 43 43 42 42 42 42 42 **41 41**

41 39 39 **38** 38 38 37 36 36 36 **35 33 33** 33 **32**

32 **30 30 30** 29 **28 27 27 26 25 25 24 23 23 20**

Step 3a, Load=150, res=82

Still Step 3

Falkenauer U_120_00, Bin 46

98 98 98 96 96 94 93 93 92 91 91 90 87 86 85

85 84 84 84 84 84 83 83 82 82 81 80 80 80 79

79 78 78 78 78 76 74 74 73 73 73 73 72 71 70

70 70 69 69 69 67 66 64 62 62 60 60 59 58 58

58 57 57 57 57 55 55 55 50 49 49 49 47 46 46

45 45 44 44 43 43 43 43 42 42 42 42 42 41 41

41 39 39 **38** 38 38 **37** 36 36 36 **35 33 33** 33 **32**

32 **30 30 30 29 28 27 27 26 25 25 24 23 23 20**

Step 3a, Load=150, res=82

Still Step 3

Falkenauer U_120_00, Bin 47

98 98 98 96 96 94 93 93 92 91 91 90 87 86 85

85 84 84 84 84 84 83 83 82 82 81 80 80 80 79

79 78 78 78 78 76 74 74 73 73 73 73 72 71 70

70 70 69 69 69 67 66 64 62 62 60 60 59 58 58

58 57 57 57 57 55 55 55 50 49 49 49 47 46 46

45 45 44 44 43 43 43 43 42 42 42 42 42 41 41

41 39 39 **38 38** 38 **37** 36 36 36 **35 33 33** 33 **32**

32 30 30 30 29 28 27 27 26 25 25 24 23 23 20

Step 3b, Load=149, res=81

Still Step 3

There remained only 2 small items, 48 bins are used, and OPT=48

Its efficiency

Solved all instances optimally from the 20 instances.

Other subclasses (n=250, n=500, n=1000):

- We need to find carefully the parameters.
- Not so good as for the first subclass where n=120
- Still we can solve about 92% of the instances (among the 80 instances) optimally.

What can we say generally?

- The BP is strongly NP-hard.
- But sometimes the case is easy.
- Like:

All items have the same size

All items are 1 or 2

All items are 1 or k (or, e.g., 1, 2, 6, 12, 60, 180)

Bin size is C=3: FFD is optimal

Bin size is C=4: FFD is optimal

Bin size is C=5: FFD is optimal

Bin size is C=6: FFD is optimal

- Bin size is C=7: The first case when FFD is not surely optimal (if each bin: 3,2,2)

Write up an ILP?

We generate all „patterns”.

Any pattern (configuration vector) is a column.

Any row stands for the number of items of certain size.

Rhs: the number of items of the different sizes.

Variables: the number of bins with certain patterns.

Fixed C, fixed [a,b]: finite (but huge) number of patterns.

(Gomory-Gilmore, Cutting Stock, 1961. For 40 different item sizes:
100 million patterns, column generation and rounding.)

What happens if n is bigger

What happens if n grows big?

Theorem of Lenstra (83) provides the following:

Let C (bin size) be fixed.

Also, let the items come from a fixed [a;b] interval.

Then there exists an algorithm, polynomial in n (the number of items) that solves optimally the problem.

But: „poly" here does not mean small step number, the algorithm cannot be used in practice. So the case „must be" much easier, if we also take into account that the items sizes are chosen at random by uniform distribution!!!

What happens if n is bigger

What happens if n grows big?

Let C (bin size) be fixed.

Also, let the items come from a fixed [a;b] interval (then the number of different sizes is d=b-a+1).

Theorem (from talk of Klaus Jansen): There exists an algorithm, low order polynomial in n (the number of items) that solves the problem optimally (with possibly a huge coefficient hidden in O(.)).

What happens if n is bigger

What can we gain from getting the items chosen at random (by uniform distribution) from [a,b]?

It seems the problem becomes simpler.

Conjecture (or rather a „feeling”): There exists an algorithm ALG low order polynomial in n (the number of items) and having „small” multiplier in O(), that solves the problem optimally with high probability.

(Say, the step number is about $20*n^4$, and the number of optimal solutions is above 90%.)

Some other options if n grows

- Suppose there are 10,000 items. We partition them into 10 groups and pack the groups well. (This can work if the items are chosen randomly.)
- Also if the items come online (but follow a uniform distribution)! We wait for the first 1000 items, pack them, again wait, and so on. (In fact, this is semi online with a large buffer.)
- We apply the greedy algorithm as „first phase”.
- In many cases in fact, n does not grow too high.

Finally:

Thank you for your kind attention!

Many open questions remain, like

- investigating the other benchmark classes
- Celebrating „50 years of Bin Packing” ???