Journal of Heuristics, (Submitted, under review), 1–6 (1997)

Heuristic Solution of Open Bin Packing Problems

IAN P. GENT ipg@cs.strath.ac.uk
Department of Computer Science, University of Strathclyde, Glasgow G1 1XH, United Kingdom

Abstract. Benchmark problems should be hard. I report on the solution of the five open benchmark problems introduced by Falkenauer in this journal for testing bin packing problems. Since the solutions were found either by hand or by using very simple heuristic methods, these problems would appear to be easy. In four cases I give improved packings to refute conjectures that previously reported packings were optimal, and I give a proof that the fifth conjecture was correct. In some cases this led to implemented heuristic methods which produced better solutions than those reported by Falkenauer and up to 10,000 times faster. Future experimenters should be careful to perform tests on problems that can reasonably be regarded as hard.

Keywords: bin packing, heuristics, empirical study of algorithms, benchmark problems

1. Introduction

A one-dimensional bin packing problem consists of a set of objects S and a set of bins B. Each object $s \in S$ has a weight w_s, and each bin $b \in B$ has the same capacity c. The aim is to partition the objects between the bins so that the sum of weights in each bin is less than or equal to the capacity. All weights and capacities are integers. This is the NP-complete problem SR1 from (Garey and Johnson 1979). An extensive survey of approximation algorithms for the problem has been given recently (Coffman Jr. *et al.* 1997).

Falkenauer (1996) tested a novel hybrid grouping genetic algorithm (HGGA) for solving bin packing problems and also a Martello & Toth's (1990) procedure (MTP), a complete search method. Experimental testing of these techniques was on two problem classes. One of these classes (the 'triplets' class) consisted of problems with predetermined packings, but the other (the 'uniform' class) consisted of purely randomly generated numbers.[1] In the uniform class, the optimal solution is not known in advance, although a lower bound on the optimum can be computed easily. Falkenauer reported that neither HGGA nor MTP (with a backtracking limit imposed) was able to attain the lower bound in five problems, and conjectured that these problems had no solution meeting the lower bound. I have shown that four of these lower bounds *can* be reached.

In the rest of the paper I first describe the methods used to solve the open benchmark problems. But this paper is not, in Hooker's (1995) critical phrase, a 'track meet' intended to show the superiority of my methods. Instead, I argue that these problems appear to be easy, questioning the validity of any conclusions drawn from their perceived hardness. I then describe a methodology used in many domains, including bin packing, to avoid benchmark sets being found to be easy. In an Appendix I present solutions to these problems and a proof that one of the problems is indeed insoluble at the lower bound.

2. Finding Solutions

The bin packing problems from Falkenauer's uniform class involve integer weights generated randomly and uniformly in the range $[20, 100]$ and a fixed capacity 150. Since the average weight is 60, many bins will contain exactly two objects with most of the rest containing three objects.

A simple preprocessing step can be applied. Suppose that the objects s and t satisfy $w_s + w_t = c$, and that in some solution s is put in a bin with a set of objects U and t with a set of objects V. The sum of weights of U cannot be greater than w_t as can be seen from the inequality $w_s + \sum_{u \in U} w_u \leq c = w_s + w_t$. Similarly the sum of weights in V cannot be greater than w_s. So we have that $\sum_{u \in U} w_u + \sum_{v \in V} w_v \leq w_s + w_t = c$. Thus we can transform our supposed solution into one which pairs s and t in one bin and the sets of objects U and V in the other. As this uses the same number of bins as the original packing, applying this step iteratively shows that if any solution exists using a certain number of bins, there is also a solution in which we pair objects whose weights sum to exactly c. As a preprocessing step to reduce the size of the given problems, we can find all pairs of weights adding up to exactly c and put the corresponding objects together in bins. I applied this preprocessing step before using other approaches to solve the problems.

With minimal computer assistance for bookkeeping, I studied the 120 object problem `u120_19` by hand, attempting to find a packing into 49 bins. Since the sum of the 120 weights is greater than the capacity of 48 bins, such a packing would be optimal. Having removed pairs of weights adding up to exactly the capacity, 82 numbers remained to pack into 30 bins. No bin can contain a single object as even the heaviest object would leave too much space in the bin. So a simple counting argument showed that at least eight more bins would need to contain exactly two objects: allocating 82 numbers to 30 bins must leave at least eight with two objects and at most 22 with three. To minimise the loss of spare capacity, I selected eight pairs of weights which added to 149, picking pairs with as high a larger number as possible. This left 66 objects to pack into 22 bins. I attempted to solve this by repeatedly choosing triplets of weights adding to exactly 150, or failing that which came as near as possible. Remarkably, this was successful and produced a solution without any backtracking. The heuristics used were highly informal: it seemed sensible to start with the largest remaining numbers, as there are typically fewest possibilities for these numbers; and where given a choice I tended to try to pick a triplet that left many possibilities for the future open, for example not taking the last object with a particular weight.

The informal process just described was easy to implement. I wrote a program that first found as many pairs of objects as necessary to leave exactly three times as many objects as bins, picking pairs summing to the capacity c or as close to it as possible. Then the remaining numbers were considered in decreasing order: for each number in turn all pairs of other numbers are found which make the triplet sum to exactly c, or as close to it as possible. The only significant deviation from the manual method is that where a choice of triplets is available, the choice is made at random: as well as being easier to implement this allows for a repeated attempts to be made with different random seeds.

The procedure just outlined was extremely successful on Falkenauer's 120 and 250 object problems when implemented in Common Lisp on a DEC Alpha with 233 MHz processor. On 120 object problems it solved to optimality 19 out of 20 problems first time, and the remaining problem on a second attempt. In no case did an individual run take more than 0.29 seconds CPU time, compared to a maximum of more than 3000 seconds for both HGGA and MTP reported in Table 1 of (Falkenauer 1996) on the two problems which were not solved by either. Since both these problems were solved first time, a speedup of 10,000 times was achieved while finding better solutions. The solutions reported in the Appendix were found by this procedure. Even considering only problems solved to optimality by HGGA and MTP, the worst case was still more than 50 times worse for both those procedures than that using my very simple heuristic procedure.

On 250 object problems the simple heuristic program was slightly less successful. No solution was found better than those found by HGGA and in one case, problem `u250_15`, no optimal solution was found in 30 attempts, each using about 1 second of CPU time, whereas HGGA did find an optimal solution. However, in every other case using at most two attempts, a solution was produced as good as those found by HGGA or MTP. The maximum CPU time of any attempt was 6.2 seconds, 1000 times faster than the worst case of HGGA and MTP.

Of the three problems whose optimal packing this left open, one was quickly proved to be insoluble in less than the number of bins reported in (Falkenauer 1996), and this proof is given in the Appendix. I found, however, that with slight manual intervention my program was able to solve one of the remaining two problems, `u250_07`. The intervention was to pick out four quartets of numbers each of which summed to 150 before giving the remaining problem to the program: it was then able to pack the remaining 234 objects, as given in the Appendix.

In the remaining problem, the objects' total weight is only 11 units less than the total capacity of 105 bins. Nevertheless, I found a solution to this problem purely manually, following only the automatic elimination of pairs of objects of total weight 150. After earlier, failed, attempts to find a solution or prove insolubility in 105 bins, I found a solution in about 3 hours work with pencil and paper. The solution is given in the Appendix. This solution was found partly by use of 'repair' methods. For example, when no good triplet can be found from remaining objects, one can swap two of the remaining objects for two objects in an existing triplet provided the two pairs of objects' total weight is the same. This step opens new possibilities for the set of remaining numbers. This repair mechanism can of course be implemented: doing so and establishing if simple heuristics can lead to automated solutions for bin packing problems of this nature remains interesting future work.

3. Discussion

The open problems solved here were not originally an end in themselves, but instead provided as benchmarks in order to compare two general purpose methods. But the fact that these problems are amenable to solution by very simple methods must bring into doubt any conclusions based on the perceived hardness of the problems. The intrinsic search space of even the smallest problem solved here with 120 numbers to put into 49 bins is $49^{120}/49! > 10^{140}$ and the problems have very little underlying structure to help search. That nevertheless the problems were solvable manually suggests that they are not intrinsically hard: there is only a place for *easy* benchmark problems when they are intentionally set as such. Any conclusion that either of the methods tested by Falkenauer (1996) is superior based on experiments on these problems must therefore be doubted. Indeed, it could even be argued that both methods were particularly well suited to this kind of problem, as both rely on a dominance criterion which is expected to become ineffective as the number of objects in each bin increases (Falkenauer 1996): the test problems were such that the number of objects in each bin was mostly two or three.

The spare capacities left by optimal packings for the open problems are noted in the appendix. Gent and Walsh (1997) noted considerable variation in problem difficulty correlated with the amount of spare capacity, and overcame this by choosing a random problem generator which left a fixed amount of spare capacity.

How can one conduct experiments in such a way that results are unlikely to be criticised as being on easy problems? One method that has become favoured in the Artificial Intelligence community follows seminal work by Cheeseman, Kanefsky and Taylor (1991) on the existence of phase transition phenomena in randomly generated NP-complete problems. A point of critical constrainedness has been discovered in many random problem classes. At this point about half the problems have solutions while half are unsolvable, and problems seem to be intrinsically hard to solve. The notion of constrainedness can be formalised and this has led to unity in the study of phase transitions in different NP-complete problem classes (Gent *et al.* 1996). To date no method has been found which makes all problems at a phase transition easy, despite extensive empirical work in many domains: as described for example in a special volume of *Artificial Intelligence* (Hogg *et al.* 1996). This methodology can be applied to bin packing problems and indeed seems to lead to hard problems at a phase transition (Gent and Walsh 1997).

4. Conclusions

Five benchmark problems left open by CPU-intensive methods using a genetic algorithm and an exhaustive search method have been solved by very simple heuristic methods using at most seconds of CPU time and hours of manual work. Rather than make extravagant but unjustified claims for the methods used to solve these problems, I draw a more general conclusion from my work. Where simple special purpose methods are successful on benchmark problems, one must question the underlying hardness of test data sets. Experimenters should be aware of all techiques which exist to avoid their test data later being shown to be easy. One example is the exploitation of phase transition phenomena in combinatorial problems. Failure to learn this lesson may lead future workers into making unreasonable claims based on their experiments.

Notes

1. These data sets are available from the OR Library, a collection of test data sets for Operational Research problems, `http://mscmga.ms.ic.ac.uk/info.html`. In the uniform class, twenty problems are provided with each of 120, 250, 500, and 1000 objects. Note that the numbering of the data sets starts from 0, so is one less than the line numbers of tables reported by Falkenauer (1996). For example results on problem `u250_15` are reported as Run 16 in Table 2.

References

P. Cheeseman, B. Kanefsky and W. Taylor. 1991. Where the really hard problems are. In *Proceedings of the 12th IJCAI*, 331–337.

E. Coffman Jr., M. Garey and D. Johnson. 1997. Approximation algorithms for bin packing: a survey. In *Approximation Algorithms for NP-Hard Problems*, 46–93. PWS Publishing Company.

E. Falkenauer. 1996. A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics* 2:5–30.

M. R. Garey and D. S. Johnson. 1979. *Computers and intractability : a guide to the theory of NP-completeness*. W H Freeman.

I. Gent and T. Walsh. 1997. From approximate to optimal solutions: Constructing pruning and propagation rules. In *Proceedings of IJCAI 97*. In press.

I. Gent, E. MacIntyre, P. Prosser and T. Walsh. 1996. The constrainedness of search. In *Proceedings of AAAI-96*, 246–252.

T. Hogg, B. Huberman and C. Williams., eds. 1996. *Artificial Intelligence*, volume 81. Elsevier. Special Volume on Frontiers in Problem Solving: Phase Transitions and Complexity.

J. N. Hooker. 1995. Testing heuristics: We have it all wrong. *Journal of Heuristics* 1:33–42.

S. Martello and P. Toth. 1990. Lower bounds and reduction procedures for the bin packing problem. *Discrete Applied Mathematics* 22:59–70. Cited by (Falkenauer 1996).

Appendix: Detailed solutions

Problem `u250_13` is insoluble in 102 bins so Falkenauer's reported solution in 103 bins is optimal. If we are putting 250 numbers in 102 bins we must have at least 2 numbers in every bin, as there are only 6 units of capacity spare and the largest number is 100. So there must be at least 56 bins with only two numbers in each. And so at least 50 of those must add up to exactly 150, or the allowable spare capacity will be exceeded. But there are only 47 pairs of numbers adding up to 150, so the problem is insoluble.

The following are packings of `u120_08`, `u120_19`, `u250_07`, and `u250_12` into fewer than the number reported by Falkenauer and into the easily provable optimum number.

```
u120_08 into 50 bins: spare capacity 22 units
(51 51 47) (57 29 57) (57 43 48) (67 37 44) (67 37 45) (67 38 45)
(69 35 46) (70 23 57) (72 39 39) (79 29 42) (82 23 45) (82 34 34)
(84 26 40) (88 22 40) (91 26 33) (94 27 29) (94 20 36) (95 25 30)
(97 21 32) (97 20 33) (85 64) (89 60) (89 60) (91 58) (91 58) (95 54)
(100 49) (100 49) (100 49) (75 75) (76 74) (77 73) (79 71) (80 70)
(81 69) (81 69) (85 65) (85 65) (86 64) (87 63) (88 62) (89 61)
(91 59) (92 58) (92 58) (96 54) (97 53) (98 52) (98 52) (98 52)
```

```
u120_19 into 49 bins: spare capacity 28 units
(48 48 47) (50 49 50) (58 23 58) (62 43 44) (62 39 49) (63 43 44)
(66 38 46) (68 38 44) (74 37 39) (75 33 42) (77 32 41) (79 35 36)
(89 21 40) (90 26 34) (94 23 33) (94 23 33) (95 24 31) (95 27 28)
(97 24 29) (97 22 31) (97 21 32) (97 25 28) (77 72) (80 69) (86 63)
(90 59) (95 54) (97 52) (99 50) (99 50) (76 74) (77 73) (78 72)
(83 67) (84 66) (85 65) (85 65) (86 64) (88 62) (90 60) (90 60)
(92 58) (93 57) (95 55) (96 54) (96 54) (99 51) (100 50) (100 50)

u250_07 into 103 bins: spare capacity 32 units
(38 38 38 35) (54 44 46) (55 46 47) (58 45 47) (58 44 48) (61 44 45)
(63 41 46) (63 43 44) (64 39 47) (64 40 46) (69 38 43) (73 36 41) (83 64)
(83 25 41) (83 31 36) (84 32 34) (84 30 36) (84 24 42) (84 24 42) (90 27 33)
(90 21 39) (90 23 37) (93 23 34) (93 23 34) (93 22 35) (93 27 30) (97 25 28)
(98 24 28) (98 22 30) (99 20 31) (99 21 30) (99 24 27) (99 20 31) (99 23 28)
(99 23 28) (99 24 27) (93 55) (100 48) (100 48) (76 73) (78 71) (78 71)
(78 71) (78 71) (80 69) (85 64) (88 61) (88 61) (91 58) (100 49) (100 49)
(100 49) (75 75) (75 75) (76 74) (76 74) (76 74) (77 73) (77 73) (78 72)
(80 70) (80 70) (81 69) (81 69) (82 68) (82 68) (82 68) (83 67) (85 65)
(85 65) (86 64) (86 64) (86 64) (87 63) (87 63) (88 62) (89 61) (89 61)
(89 61) (89 61) (89 61) (90 60) (90 60) (91 59) (91 59) (92 58) (93 57)
(94 56) (94 56) (94 56) (95 55) (96 54) (96 54) (96 54) (97 53) (97 53)
(98 52) (99 51) (100 50)
(36 37 38 39) (40 41 34 35) (42 43 32 33) (44 45 30 31)

u250_12 into 105 bins: spare capacity 11 units
(67 46 37) (43 41 33 33) (71 49 30) (58 47 45) (64 43 43) (70 42 38)
(58 47 45) (57 57 36) (81 40 29) (94 33 23) (98 49) (82 34 34) (67 43 40)
(95 33 22) (98 49) (75 74) (95 54) (57 47 46) (57 47 46) (81 67) (100 49)
(57 47 46) (58 48 44) (58 48 44) (62 44 44) (64 43 43) (67 42 41) (67 45 38)
(70 42 38) (71 42 37) (81 36 33) (89 32 29) (90 31 29) (94 29 27) (95 28 27)
(95 29 26) (97 27 26) (97 27 26) (97 27 26) (97 31 22) (97 31 22) (98 28 24)
(98 30 22) (98 32 20) (98 32 20) (75 75) (76 74) (76 74) (77 73) (78 72)
(78 72) (79 71) (79 71) (80 70) (80 70) (81 69) (82 68) (82 68) (82 68)
(82 68) (84 66) (84 66) (84 66) (84 66) (85 65) (85 65) (85 65) (86 64)
(87 63) (87 63) (87 63) (87 63) (88 62) (88 62) (88 62) (89 61) (89 61)
(89 61) (90 60) (90 60) (91 59) (91 59) (91 59) (92 58) (93 57) (93 57)
(94 56) (94 56) (95 55) (95 55) (95 55) (95 55) (96 54) (96 54) (96 54)
(97 53) (98 52) (99 51) (99 51) (99 51) (99 51) (100 50) (100 50) (100 50)
(100 50)
```

Acknowledgments

Supported in part by a Strathclyde University starter grant. I am a member of the APES research group and I thank the other members for many stimulating discussions. I especially thank David Clark and Patrick Prosser for independently checking my solutions, and Toby Walsh for helpful comments on a draft.