A Genetic Algorithm with Exon Shuffling Crossover for Hard Bin Packing Problems

Philipp Rohlfshagen
School of Computer Science
University of Birmingham
Birmingham B15 2TT, United Kingdom
P.Rohlfshagen@cs.bham.ac.uk

John A. Bullinaria
School of Computer Science
University of Birmingham
Birmingham B15 2TT, United Kingdom
J.A.Bullinaria@cs.bham.ac.uk

ABSTRACT

A novel evolutionary approach for the bin packing problem (BPP) is presented. A simple steady-state genetic algorithm is developed that produces results comparable to other approaches in the literature, without the need for any additional heuristics. The algorithm's design makes maximum use of the principle of natural selection to evolve valid solutions without the explicit need to verify constraint violations. Our algorithm is based upon a biologically inspired group encoding which allows for a modularisation of the search space in which individual sub-solutions may be assigned independent cost values. These values are subsequently utilised in a crossover event modelled on the theory of exon shuffling to produce a single offspring that inherits the most promising segments from its parents. The algorithm is tested on a set of hard benchmark problems and the results indicate that the method has a very high degree of accuracy and reliability compared to other approaches in the literature.

Categories and Subject Descriptors

G.3 [**Mathematics of Computing**]: Probability and Statistics – Probabilistic Algorithms

General Terms

Algorithms, Experimentation, Performance

Keywords

Genetic algorithms, bin packing problem, exon shuffling

1. INTRODUCTION

Genetic algorithms (GAs; see [7], [11]) have become a popular choice of algorithm for numerous difficult optimisation problems where conventional methods tend to fail. GAs maintain a population of potential solutions to the problem of interest and loosely follow the principles of Darwinian evolution to evolve solutions of increasing quality. Individuals from the population are selected according to their fitness to produce offspring by means of crossover and mutation. The offspring are subsequently evaluated and, if successful, placed back into the population. GAs have traditionally been modelled on the field of population genetics (see [3]): Individuals in the algorithm's population represent "chromosomes with many loci and few alleles per locus" (see [7], p 71). While GAs were never meant to be accurate models of biological systems, we believe it may be beneficial to consider possible inconsistencies or shortcomings from a biological point of view, in order to yield improvements over the algorithm's traditional design. It may be helpful, for example, to view individuals in the population as (eukaryotic) genes rather than (prokaryotic) genomes. Often these distinctions are irrelevant from a computational point of view, but they may encourage exploration of additional biological processes in an overall more plausible framework. The design of the algorithm presented here is an example of such undertaking. It is inspired by processes of molecular genetics that have been suitably abstracted to fit our problem of interest, the bin packing problem (BPP). The results indicate how a nature inspired approach may produce a comparable performance to other approaches in the literature, without the need for additional problem specific heuristics.

The objective of the BPP is to fit a fixed number of n items, each of weight w_i, into the fewest possible number of bins, each of which has capacity c. In other words, the number of bins has to be minimised while none of the bins' capacities may be exceeded. This BPP has many industrial applications and is found frequently in widely occurring real world scenarios such as vehicle loading. Since this problem is NP-hard [2], and well studied in the literature, it is an excellent choice of problem on which to explore novel computational ideas. Furthermore, the structure of the problem is highly suitable for abstractions from molecular genetics, as will be discussed in section 4.

The remainder of this paper is structured as follows: A brief summary of previous work on the BPP is given is section 2. This is followed by a discussion of molecular processes that have inspired the design of the new algorithm introduced in this paper. Section 4 outlines how these biological processes have been suitably abstracted to be usefully employed in an algorithmic framework. The experimental setup is described in section 5, followed by an analysis and discussion of the results in section 6. Some conclusions are

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

presented in section 7, along with a brief discussion of possible future work in this area.

2. PREVIOUS WORK

There are numerous evolutionary approaches for the BPP in the literature. Interestingly, the majority of these approaches are hybrid, combining GAs with problem specific heuristics. It is now well established that evolutionary approaches work very well in combination with problem specific heuristics and the review of relevant literature confirms this. This section will therefore briefly overview a range of relevant heuristics, before outlining the use of GAs for the BPP. It should be noted that due to the vast number of publications addressing the BPP, this review is necessarily restricted to the key relevant ideas.

The most classic heuristics are the first fit decreasing (FFD) and the best fit decreasing (BFD) algorithms. Both of these approaches consider all items in decreasing order of their weights, and then place them systematically into the bins. The worst case result for FFD and BFD is $(11/9)opt + 4$, where *opt* represent the optimal number of bins [2]. As pointed out by Gupta et al. [6], the performance of both heuristics deteriorates when the optimal solution requires the majority of bins to be filled to (near) the maximum degree possible. Gupta et al. therefore suggested a new heuristic, called minimum bin slack (MBS), to overcome this deficiency. MBS is naturally bin-focused: Any given bin is filled to the maximum degree before the next bin is considered. Backtracking may be used to escape from local optima. Fleszar et al. have suggested several variations of MBS [5]: One of these heuristics is MBS', which is identical to MBS except that it uses an initialisation procedure that speeds up the algorithm. In their study, the authors found that amongst several suggested approaches, a combination of heuristics gave the overall best performance when tested on a large set of benchmark problems: The application of MBS' followed by a variable neighbourhood search (VNS) proved an effective combination. Alvim et al. [1] also suggested the use of a hybrid heuristic that incorporates many different techniques. Their approach is probably the most complex one, but also produces the best results of all reviewed approaches, including our work (see table 2). However, their algorithm is rather difficult to implement and relies crucially upon numerous parameters that need to be decided upon prior to execution. This brief review represents only a fraction of different approaches studied, but, interestingly, hybrid approaches seem the most promising: The results of one heuristic are used as starting point for another. The same holds true for evolutionary algorithms where the use of heuristics in addition to the GA seems commonplace. These approaches are reviewed next.

GAs maintain a population of potential solutions upon which selection, crossover and mutation may act. In order to represent solutions to the problem, an appropriate encoding needs to be defined. The most intuitive encoding for the BPP is a simple permutation of integers. The solution may subsequently be constructed from the encoding by scanning the permutation from one end to another, fitting items into each bin while possible before starting a new bin. This ensures that all the solutions are legal, and also allows the use of well established crossover and mutation operators that work for any permutation based encoding (such as the travelling salesman problem). However, as was pointed out by Falkenauer [4], this approach has a very high level of redundancy. In fact, it is clear that any permutation of the set of items in one bin will encode an identical solution. For example, if a combination of 5 items fit into one bin, there are 5!=120 different ways to encode the same solution, and multiplying such numbers across all the bins quickly leads to enormous redundancies. Falkenauer consequently suggested the idea of group encoding, whereby only the group membership of an item matters and not its order within the group [4]. This will overcome the aforementioned redundancy. However, a complicated crossover operation needs to be used to ensure that the offspring are complete and do not contain any duplicate items: Duplicates need to be deleted and missing items have to be re-inserted using a problem specific heuristic. The algorithm suggested by Falkenauer further implemented a concept based on the dominance criterion due to Silvano and Paolo [14].

The group based encoding approach has also been used by Lima and Yakawa [10], who suggested a different approach to crossover that makes use of MBS' and a first-fit heuristic. The crossover operator transmits some of the parental segments in their entirety to the offspring with the remaining items being added to the offspring using the aforementioned heuristics. Explicit care is taken to ensure that no grouping exceeds the bin's capacity. The authors compare their GA to two non-evolutionary methods and conclude that their GA has superior performance in terms of the solution's accuracy. Another hybrid GA, due to Kao and Lin [8], uses a simulated annealing procedure to enhance the performance of the underlying GA. The design of this algorithm is driven by the stochastic nature of both GAs and simulated annealing which is thought to overcome the shortcomings of deterministic approaches such as FFD or BFD. There are numerous further evolutionary approaches that have not been considered here as they are either tailored to specific instances of the BPP, or designed for variants of the one-dimensonal BPP.

3. BIOLOGICAL BACKGROUND

Various approaches to the BPP, such as the group encoding due to Falkenauer, have demonstrated how the BPP may be solved in a modular fashion. A bin focused view of the problem allows solutions to be composed of individual groups, each of which may be considered an independent unit. The following review of some selected aspects of molecular genetics will show how such modularity is apparent in natural systems, and how such systems may provide fruitful inspiration for the design of evolutionary algorithms for the BPP.

The information processing architecture of cells proceeds by transcribing regions of double-stranded DNA, known as genes, to single-stranded RNA. The RNA is then translated into a chain of amino acids using a mapping known as the genetic code. However, unlike prokaryotic genes, most eukaryotic genes are composed of numerous segments: Some of these segments (exons) contribute towards a gene's protein product, while other segments (introns) do not. The intermediate step of splicing introns is known as RNA processing. The expression of exons and introns is regulated by several factors (e.g., cell type) and post-transcriptional processes such as alternative splicing may alter the pathway of expression. In some cases, non-coding sections may also contain regulatory sequences that directly affect sur-

rounding segments or serve as buffers for crossover to occur. There are many different processes that make use of the modular structure of genes to produce a number of proteins far exceeding the number of genes in higher organisms (e.g., alternative splicing and RNA editing). Recent advances in molecular genetics have made it clear that the modular organisation of genes is highly important for the evolution of complexity. In fact, most of a gene's exons are associated with independent protein domains allowing natural systems to generate complexity in a piecewise fashion. One of the processes allowing the evolution of genes of increasing complexity is exon shuffling (see [9]), which will be described next.

Exon shuffling refers to the mechanism of exons being recombined by means of crossover to yield novel protein products, possibly of advanced functionality. If individual exons are independent units, then there is an increased likelihood that a combination of such units will yield a functional or near functional protein product. Introns may serve as buffers between exons: Introns occupy far larger regions of a gene, thus minimising the probability of crossover dividing an exon. Processes such as alternative splicing may be utilised to overcome temporary negative selection pressure [12]: Exons that are being integrated into an already functional gene may be expressed alternatively. The gene's original functionality is preserved while a small number of transcripts express the newly acquired exon. Non-functional transcripts are eventually discarded by processes such as non-mediated decay. This relief of negative selection pressure allows the accumulation of neutral mutations and the increased rate of change may eventually yield a fully functional gene. It follows that not genes but their constituents, exons and introns, are the "building blocks" of higher organisms.

4. AN EXON SHUFFLING APPROACH

The success of the algorithm presented in this paper is due mainly to its crossover operator, the algorithm's computationally most costly operation. This crossover makes use of the group encoding whereby individuals are composed of numerous segments. Each segment corresponds to a single bin and may contain any number of items without restriction. This modularisation of the search space allows the assignment of a cost value to each segment by calculating the associated remaining space. If the combined weight of all the items within a bin exceeds that bin's capacity (i.e. there is a constraint violation), the excess is multiplied by an appropriate penalty value and added to the cost. We found empirically that a penalty value of 10 was suitable for the test cases we considered, but other values may work better for different sets of benchmark problems.

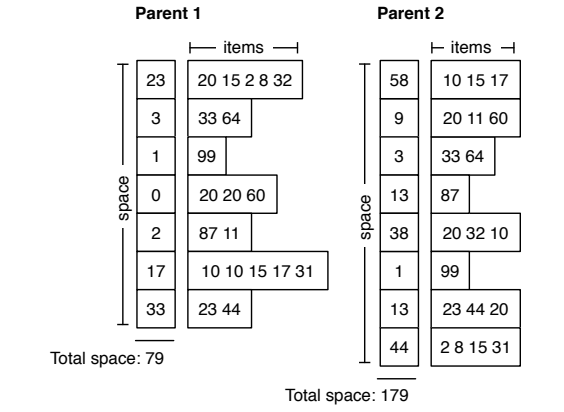
During crossover, all segments from both participating parents are evaluated and sorted by their cost (waste of space): Bins with the least cost (including bins violating the capacity constraint) are listed first. A single offspring is then created from a two phase crossover event that considers all segments in order of their increasing cost. Segments of equal cost are chosen at random. The first phase of crossover adds all segments to the offspring that are mutually exclusive from one another. In other words, only segments exclusively containing items not yet represented by the offspring are added. The second phase considers all remaining segments in the increasing order of their cost, and adds any segment that contains at least one new item. All other items within that segment already present in the offspring are replaced by the most similar items not yet in the offspring. The similarity of items is measured according to their difference in weight. If there are no more items missing from the offspring, the remaining duplicates are simply deleted and the crossover event terminates. A simple scenario of the crossover event, using a bin size of 100, is depicted in figure 1: (a) Two parents are chosen from the population. (b) Their combined segments are sorted in increasing order of their cost. Phase 1: All mutually exclusive segments are added to the offspring. (c) Phase 2: All segments containing at least one new item are added to the offspring. All duplicates are replaced with the closest possible item that is still missing.

The crossover operator ensures that not only does the offspring inherit the most tightly packed bins from both parents, but also that the bins with the least amount of free space are preserved the most. Although the segments are considered as independent units, there is an interrelationship between the segments because any individual must contain all variables of the problem's instance exactly once. This is guaranteed by the replacement procedure, which is also loosely inspired by known properties of the genetic code. Natural systems evolved such that similar amino acids, the building blocks of proteins, are encoded by similar segments of RNA (codons). Mutations to a segment of RNA are therefore likely to result in a similar amino acid, minimising the resulting phenotypic change. We therefore replace an item with the most similar available one and thus guarantee that even modified segments retain their original composition to the maximum degree possible. The penalty term applied to infeasible bin arrangements ensures that those bins are less likely to be considered first during crossover. This will tend to purge all infeasible bins over time if the penalty term is chosen appropriately. The need for an explicit repair procedure is thus avoided. The fitness of the offspring follows directly from the crossover event and is simply the combined amount of free space over all the bins.

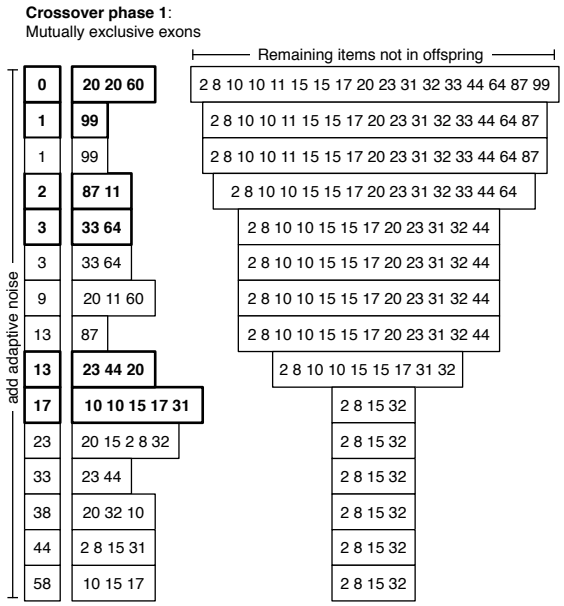
The crossover operator we are proposing here is clearly greedy in the sense that it always considers the seemingly best (most tightly packed) bins first. This approach will consequently be susceptible to entrapment in local optima, particularly in cases where the optimal solutions contain bins with moderate amounts of free space. In order to counteract this, noise can be added to the cost of each bin such that the ordering of segments is only approximate. The amount of noise is measured as a percentage of the bin's capacity and is made part of the individual's encoding: A binary sequence is used whose integer value determines the percentage of noise added to the segment's true cost. The control sequence is limited to 4 bits, allowing a range of 0-15% to be expressed, which is subsequently shifted to encode noise values in the range [5, 20]. Experiments with a fixed level of noise indicated this to be the best range, while attempts to evolve noise in the full range [0,100] failed to produce any noteworthy improvements.

5. EXPERIMENTAL SETUP

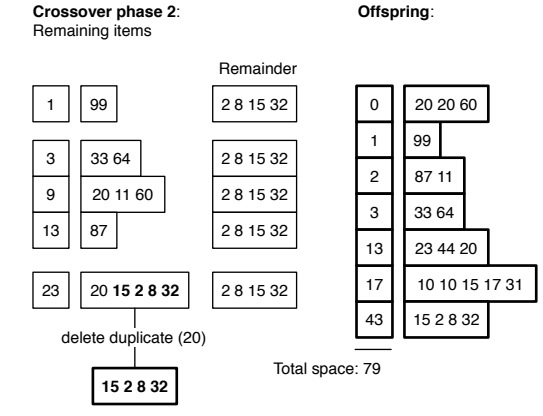
The new algorithm proposed here was tested on a suite of benchmark problems due to A. Scholl and R. Klein which is



(a)



(b)



(c)

Figure 1: Two-phase exon-shuffling crossover: (a) parent selection, (b) sorting, (c) combination. Refer to main text for details.

available online[1]. This suite is made up of three sets of problem instances, with the third set being the most difficult. It contains 10 instances, each of which consists of 200 items with weights in the range [20000, 35000] and bins of capacity 100,000. We will focus exclusively here on this set as it is deemed the most challenging. Our algorithm is executed 50 times on each instance, with a maximum limit of 50,000 function evaluations. We record the number of times each instance has been solved optimally and how many function evaluations on average are required to do so. As a measure of accuracy, the average number of bins is given alongside an indication of how many bins, on average, are overfilled. As it is presently implemented, the algorithm may return invalid solutions which contain overfilled bins. A trivial extension to the algorithm may be used to repair such encodings by removing items from overfilled bins and placing them into new ones using a first fit or similar heuristic. However, in order to demonstrate the algorithm's ability to purge infeasible sub-solutions, we do not add such a repair procedure at this stage.

The GA used for all the experiments is a standard steady-state GA with a population of size 150. Parents are selected at random and the offspring is placed into the population depending on a binary tournament with a randomly chosen individual. The previously discussed crossover operator is applied with a probability of 0.8. In the event that crossover is not applied, a randomly chosen parent is reproduced asexually (i.e. cloned). With equal probability, the mutation operator either swaps two items which are in different bins or places one item from one bin into another. The mutation operator's overall probability is set to $1/m$ where m is the current number of segments in an individual. The control sequence is mutated by flipping a bit with probability $1/p$ where p is the length of the control sequence (4 bits in this case). These parameters have been established systematically, although by no means exhaustively, by a series of experiments.

6. RESULTS AND ANALYSIS

The results from all the experiments are summarized in table 1. It shows that the algorithm has successfully found the global optimum in 8 of the 10 problem instances. In 5 cases the optimum was found in all trials, while in 3 cases the optimum was found in over 90% of trials. There are two cases, hard2 and hard3, where the global optimum has not been found at all. However, all the final solutions for these two cases are exactly a single bin away from the globally optimal solution. The same holds true for the other instances which have not been solved with 100% reliability – the unsuccessful trials were all at most one bin away from the global optimum. Interestingly, there is only a single trial in which an invalid solution was returned (for hard5), highlighting the algorithm's ability to purge infeasible sub-solutions.

The algorithm's overall behaviour is further examined by looking at its convergence over time. This is done by focussing on the population's best individual at each generation. There are three factors to consider: Fitness (bin slack), number of bins, and number of constraint violations. We select three representative instances for this analysis, one of which has been solved 100% (hard0), one which has

[1]`http://www.wiwi.uni-jena.de/Entscheidung/binpp/`.

Instance	Opt	%	Avg FE	Min FE	Max FE	Accuracy	Avg over
hard0	56	100	7177.38	2488	17186	56	0
hard1	57	100	5826.58	2384	15996	57	0
hard2	56	0	-	-	-	57	0
hard3	55	0	-	-	-	56	0
hard4	57	98	13474.10	6362	35809	57.02	0
hard5	56	92	15361.15	6316	44610	56.06	0.04
hard6	57	100	5501.98	1924	10143	57	0
hard7	55	100	7899.92	2275	22461	55	0
hard8	57	100	5265.58	1830	21475	57	0
hard9	56	98	13931.55	5079	30045	56.02	0

Table 1: Summary of the experimental results: Each instance is listed by its name and optimal solution (Opt). The third column (%) indicates the percentage of solved trials, complemented by the average number of function evaluations required (Avg FE). In addition, the minimum and maximum number of function evaluations required are shown (Min FE, Max FE). The last two columns show the overall accuracy of the algorithm (Accuracy), and the average number of overfilled bins in the final solution (Avg over).

never been solved (hard2), and one that has been solved but not all the time (hard5). The graphs are presented in figure 2. They indicate that the algorithm quickly converges towards the right number of bins, although several of them are initially overfilled. In fact, the number of overfilled bins roughly peaks when the optimal number of bins is first found. The best individual's fitness also seems to stagnate around that point. The subsequent change is then very gradual, reducing the number of overfilled bins without affecting the actual number of bins. The overall change in fitness is fractional compared to the initial change and hardly visible in the graphs. Although there is a penalty factor of 10 being applied to infeasible bin arrangements, the penalty only applies at the level of the bin and hence may not make a significant contribution to the overall fitness of an individual.

Naturally, any new algorithm, however good, needs to be compared against existing algorithms. A comparison of different approaches for this particular set of benchmark problems is shown in table 2. It shows that our new biologically inspired exon shuffling GA ranks highly in terms of success rate. In fact, only the approach due to Alvim et al. [1] performs better. However, this comparison simply compares the number of instances successfully solved without any regard to running time or computational complexity. It is therefore impossible to conclude on the performance of the algorithm presented here in regard to the previous approaches from the literature. However, the table clearly highlights the difficulty of solving the chosen set of benchmark problems and more than half of the approaches considered fail to solve any of the instances at all. The notable exception is HI_BP, which is superior to our exon shuffling GA in terms of success rate. A proper comparison of the run-time properties will need to be established in the near future to reach a conclusive verdict on the competitiveness of these two approaches. Nevertheless, HI_BP requires several pre-processing steps and undergoes multiple phases exploiting several mathematical properties of the BPP. The algorithm presented here, on the other hand, is very simple to implement and could be applicable to other problems that exhibit similar structural properties.

Approach	Reference	Solved
MBS	[6] in [5]	0
MBS'	[5]	0
Perturbation MBS'	[5]	0
Sampling MBS'	[5]	0
FFD	[13] in [5]	0
BFD	[13] in [5]	0
WFD	[13] in [5]	0
B2F	[13] in [5]	0
FFD-B2F	[13] in [5]	0
Relaxed MBS'	[5]	2
VNS	[5]	2
Perturbation MBS' & VNS	[5]	2
Genetic Algorithm	[10]	3
BISON	[13] in [10]	3
Dual Tabu	[13] in [5]	3
Exon Shuffling GA	This paper	8
HI_BP	[1]	10

Table 2: A comparison of several approaches in terms of reliability on the same set of 10 benchmark instances: For each approach, the number of instances solved is shown.

7. CONCLUSION

The genetic algorithm (GA) presented in this paper has been applied successfully to a benchmark suite of hard bin packing problems (BPPs). This GA uses a biologically inspired group based encoding to achieve an appropriate modularisation of the search space. This allows the assignment of cost values to individual sub-solutions. The crossover event is based loosely upon the theory of exon shuffling, and combines parental segments in a greedy fashion to produce a single offspring. A control sequence is used to introduce noise during the crossover event to prevent stagnation at local optima and to increase the population's diversity. The resulting algorithm exhibits a very high success rate at finding optimal solutions, solving 8 out of 10 problem instances. Even for the two unsolved instances solutions to within a single bin of the optimal solution are found. This success rate is superior to most other approaches in the literature.

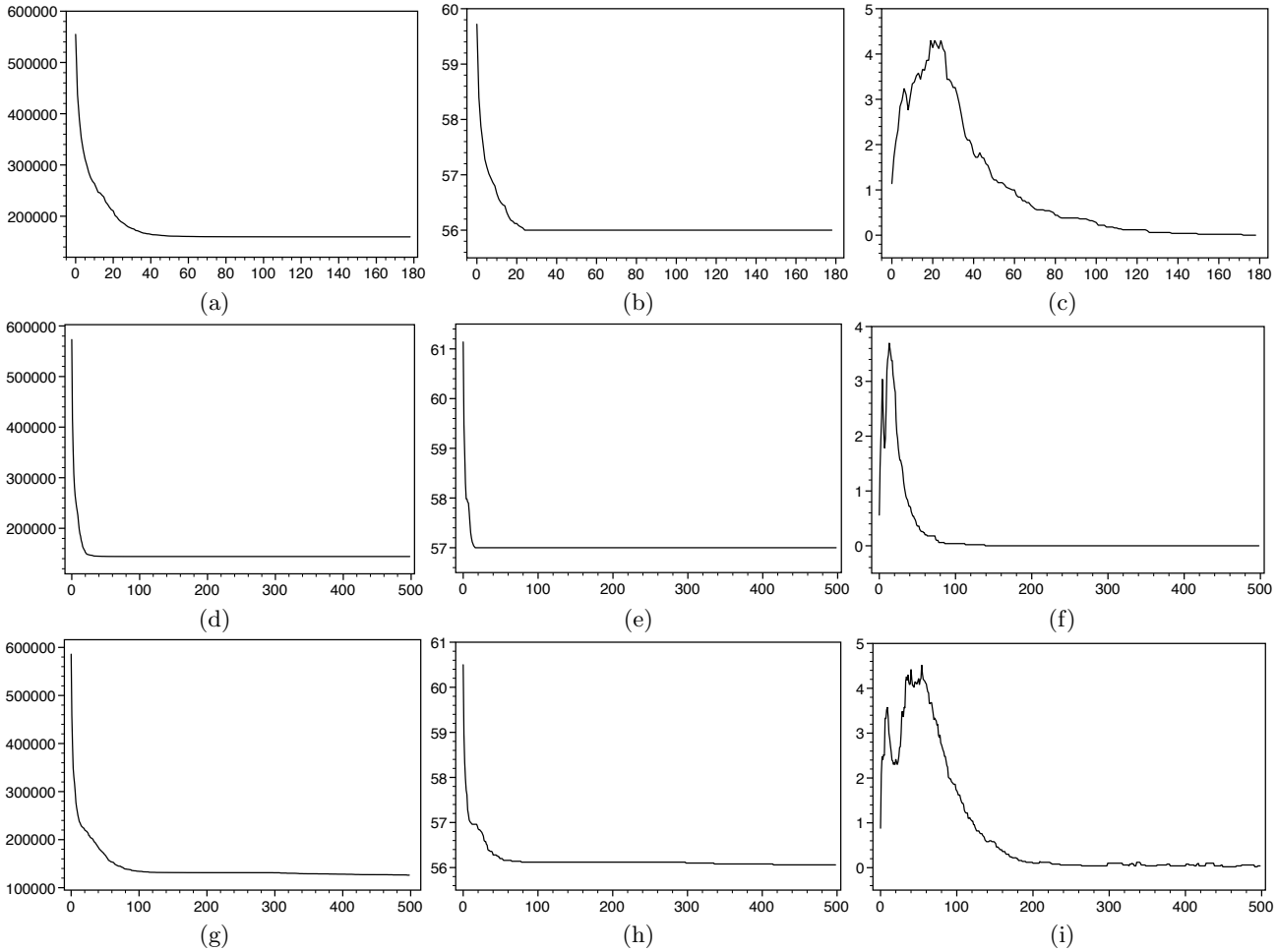


Figure 2: Graphs showing the algorithm's convergence for three selected problems: hard0 (a,b,c), hard2 (d,e,f) and hard5 (g,h,i). The first graph in each row depicts the best fitness over time as measured by the space available across all bins. The second graph shows the number of bins of the population's best solution over time. The third graph shows the number of overfilled bins in the best solution over time.

Moreover, unlike previous evolutionary approaches, the suggested algorithm does not use any additional heuristics and is therefore very simple to implement. Also, the number of variables to be considered prior the algorithm's execution is relatively small compared to other approaches in the literature.

The results presented in this paper are encouraging, but further testing is required to fully identify the algorithm's strengths and weaknesses. Further testing will address the remaining two benchmark suites due to A. Scholl and R. Klein containing 720 and 480 instances respectively. In addition, the algorithm's behaviour needs to be analysed more carefully to determine why it fails to find the global optimum in some cases but not in others. A control sequence is used to add noise during the greedy crossover event, but further (adaptive) elements may be required to achieve a greater success rate across a wider selection of bin packing problems. These, and further potential refinements, form the basis of ongoing investigations, upon which we hope to report in the future.

8. ACKNOWLEDGMENTS

This work was supported by a Paul and Yuanbi Ramsay Scholarship.

9. REFERENCES

[1] A. C. F. Alvim, C. C. Ribeiro, F. Glover, and D. J. Aloise. A hybrid improvment heuristic for the one-dimensional bin packing problem. *Journal of Heuristics*, 10:205–229, 2004.

[2] E. G. Coffman, M. R. Garey, and D. S. Johnson. An application of bin-packing to multimachine scheduling. *Journal of Computing*, 7:1–17, 1978.

[3] J. M. Daida, S. P. Yalcin, P. M. Litvak, G. A. Eickhoff, and J. A. Polit. Of metaphors and darwinism: Deconstructing genetic programming's chimera. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 453–462, 1999.

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

[5] K. Fleszar and K. S. Hindi. New heuristics for one-dimensional bin-packing. *Computers & operations research*, 29:821–839, 2002.

[6] J. N. D. Gupta and J. C. Ho. A new heuristic algorithms of the one-dimensional bin-packing problem. *Production planning & control*, 10(6):598–603, 1999.

[7] J. H. Holland. *Adaptation in Natural and Artifical Systems*. University of Michigan Press, Ann Arbor, MI, 1975.

[8] C.-Y. Kao and F.-T. Lin. A stochastic approach for the one-dimensional bin-packing problems. In *Systems, Man and Cybernetics, 1992*, volume 2, pages 1545–1551, 1992.

[9] J. A. Kolkman and W. P. C. Stemmer. Directed evolution of proteins by exon shuffling. *Nature Biotechnology*, 19:423–428, 2001.

[10] H. Lima and T. Yakawa. A new design of genetic algorithm for bin packing. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on Evolutionary Computation*, volume 2, pages 1044–1049, 2003.

[11] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1996.

[12] B. Modrek and C. J. Lee. Alternative splicing in the human, mouse and rat genomes is associated with an increased frequency of exon creation and/or loss. *Nature Genetics*, 34(2):177–180, 2003.

[13] A. Scholl, R. Klein, and C. Juergens. Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Computers & Operations Research*, 24(7):627–645, 1997.

[14] M. Silvano and T. Paolo. *Knapsack Problems, Algorithms and Computer Implementations*, chapter Bin-packing problem, pages 221–245. John Wiley and Sons Ltd., England, 1990.