Ant Colony Optimisation for the Bin-Packing Problem

Candidate Number: 029459

1 Introduction

This report explores the use of a simple ant-colony optimisation (ACO) algorithm to solve the Bin-Packing Problem (BPP). The BPP is the problem of putting n objects, $\{a_1, ..., a_n\}$, with finite weights $w_n \in W$, into one of k bins such that the total weight in each bins is as similar as possible. Specifically, the fitness of a packing solution will be measured as the difference between the heaviest and lightest bin, where a small fitness constitutes a good solution. An individual bin error, $(bin\ weight) - (\sum_{i=1}^{n} \frac{w_n}{k})$, is also sometimes useful.

While this problem will be referred to as the BPP throughout this report, the BPP discussed in literature involves placing n objects with finite weights into an infinite number of bins with finite weight capacities, where the goal is to minimise the number of bins used [12]. This report's version of the BPP is more akin to the Multi-Way Number Partitioning Problem (MNPP), where, given a finite set of numbers S and a positive integer k, the task is to partition S into k subsets where the sum of each subset is as similar as possible. The optimisation objective can vary, but a common goal is to minimise the difference between the largest and smallest sum [13].

Most setups of the MNPP, which will be referred to as the BPP, are provably NP-hard; we believe generating and verifying optimal BPP solutions are intractable problems [3]. However, the BPP has many important real-world applications in areas such as scheduling, election manipulation, and data centre file placement [4][14][18]. Thus, developing approximate polynomial time algorithms, many of which are nature-inspired, is an incredibly worthwhile task.

This report begins with a literature review exploring the development of nature inspired BPP algorithms starting in 1991, when metaheuristic algorithms were first applied to the problem. A chronological structure was chosen, and articles were collected using keywords: multi-way number partitioning; equal piles; grouping problem; metaheuristic; nature inspired; evolutionary algorithm; swarm optimisation; tabu search; and annealing. This is followed by the description, explanation, and exploration of results from the application of different ACO implementations on the BPP.

2 Literature Review

Johnson et al. were the first to apply a stochastic metaheuristic algorithm to the BPP [8]. Specifically, a simulated annealing algorithm was developed for the Number Partitioning Problem (NPP), the binary case of the BPP with $k = 2$ bins only, k_1 and k_2. Until this point, only standard heuristic algorithms had been explored for the NPP, the most successful of which was Karmarkar and Karp's self-named Karmarkar-Karp algorithm, based on the greedy construction of a binary tree with nodes representing items [11].

Johnson et al. used an n-long binary string to represent solutions, with 0s and 1s representing items in k_1 and k_2 respectively. A solution's fitness was $|\sum_{a_n \in k_1} w_n - \sum_{a_n \in k_2} w_n|$. A graph of all the solutions was constructed, with edges between solutions which differed by 1 bit. This is called a neighbourhood. Their algorithm starts by picking a random solution in the neighbourhood, then recursively randomly chooses a neighbour with d degrees of separation from the current solution in the neighbourhood. If the neighbour's fitness is less than or equal to the current solution's, the neighbour becomes the current solution. Otherwise, the neighbour becomes the current solution based on a probability parameter.

The annealing algorithm was run on $n = 200$ and $n = 500$ instances of the NPP along with the Karmarkar-Karp algorithm. In the $n = 200$ instance, the $d = 1$ and $d = 2$ annealing algorithms produced solutions of equal quality. Furthermore, the $d = 1$ algorithm, the faster of the two, took over 10,000 times longer than the Karmarkar-Karp algorithm. Results were even less promising for the $n = 500$ problem. However, these poor results were expected by Johnson et al.. They had already mathematically shown the neighbourhood's terrain was "exceedingly mountainous", and thus expected the algorithm to find locally instead of globally optimal solutions. For this reason, they believed improved representations with smoother landscapes were required for metaheuristic algorithms to compete with their heuristic counterparts.

Landscapes also posed a challenge when Jones and Beltramo became the first to apply a Genetic Algorithm (GA) to the BPP in 1991 [10]. GAs were relatively new at the time, so their implementation was to that of Holland, the first person to popularise the GA concept [7]. In their paper, Jones and Beltramo experiment with a total of nine different algorithms using three different encodings. The first encoding, called "group-numbers", is an N-string who's i^{th} element is the group of item i: 122133 represents $\{AD\}\{BC\}\{EF\}$. The second encodes a partition as a permutation of the N objects with $k-1$ dividers, all numbered consecutively from $N+1$ to $n+k-1$. For example, the partition $\{AD\}\{BC\}\{EF\}$ is represented with dividers, $\{AD\}|\{BC\}|\{EF\}$, or 14723856 in computer form. The final, and most novel, is an indirect encoding using a greedy adding heuristic. Solutions are encoded as a permutation of the N objects, and decoded such that the first k items initialise the k bins, and the remaining items are added, in order, to the bin whose resulting total weight is closest to this ideal weight: $\sum_{x_i \in S} \frac{x_i}{k}$.

Different crossover operators were trialled on a BPP with $k = 10$, $n = 34$, and $\sum_{i=1}^{n} w_n = 10000$. All GAs used linear rank bias parent selection with a population of 1000, terminating at complete convergence. Single-point, uniform, and edge-based crossover were all used for the group-numbers encoding, with additional modifications made to single-point and uniform crossover to ensure children contained the same subsets and number of subsets as their parents. Of these, edge-based crossover produced the fastest convergence, but its $O(k^4)$ complexity makes single-point crossover the preferred choice for large BPPs [9]. Order crossover and partially mapped crossover (PMX), two operators designed specifically for permutation-based encodings, were trialled with the two remaining representations. The indirect greedy heuristic encoding, when using PMX crossover, produced the best results of any GA trialled. This is to be expected; heuristics generally increases performance by providing global problem information. PMX is also well suited to grouping problems as it maintains the relative order of genes from parents through to children. This GA setup also outperformed

the switching heuristic, a common BPP solver, achieving a mean bin absolute error of 171 versus the 3610 achieved by the heuristic.

However, all the encodings used by Jones and Beltramo contain significant redundancy, providing many ways to represent the same solution. For example, the chromosomes 122133 and 311322, using the group-number encoding, both represent the same partition. In fact, chromosomes using the group-number encoding contain exponentially more redundancy as their length increases. In Ruml's testing, he found that this redundancy created an excessively large and rugged landscape, decreasing the convergence speed and increasing the chance of converging to a local maximum [17]. For this reason, in the $k = 2$ BPP, neither Johnson et al.'s nor Jones and Beltramo's metaheuristic algorithm was competitive with Karmarkar and Karp's heuristic method.

With the issue of redundancy in mind, Falkenauer argues that for grouping problems, such as the BPP, genes in a chromosome should represent subsets (the bins) rather than individual items, with operators manipulating only subsets where possible [1]. To achieve this, he created the Grouping Genetic Algorithm (GGA), which uses chromosomes made up of a group-numbers encoded "item part", such as 122133, and adds a "group part" to the end which labels the subsets, in this case 123, making 122133 : 123. Crossover is performed on the group part, mixing the parents' subsets to form children. Mutation involves removing a random subset, forming a new subset with the deleted subset's items and random items from other subsets, and finally greedily distributing the remaining items. For all these operations, the order of the subsets in the chromosome is irrelevant, which eliminates redundancy and thus shrinks the search space significantly. Falkenauer trialled his algorithm on the same BPP attempted by Jones and Beltramo. Over 30 trials with a population size of 50, the GGA found the perfect solution every time using 10 times fewer generations [2].

Greene provides a final GA design by taking Falkenauer's idea of subset-based chromosomes further [5]. Greene sacrifices memory efficiency for a time gain, encoding solutions as k binary n-strings representing subsets. The n^{th} bit of string k represents the presence of item n in subset (bin) k. The subsets are kept in ascending order of absolute error to aid Greene's crossover operator, called "Eager Breeder". This operator aggressively accumulates good subsets in children by pointing to each parent's first subset, picking the one with lowest absolute error, and incrementing that parent's pointer until k subsets are chosen. In the resulting child, items in multiple subsets are removed from their most erroneous subset. Missing items are greedily readded. Mutation is performed by randomly moving items between subsets, with less fit solutions undergoing more mutation with a higher probability. The GA also implements elitism and roulette wheel parent selection. This GA was tested on Jones and Beltramo's BPP setup, using 40 generations of 250 individuals. The perfect solution was found in 29 of 30 trial, comparable to Falkenauer's GGA while using 33.7% of the resources.

Finding suitable solution encodings is evidently a common roadblock when producing effective metaheuristic BPP algorithms. Despite this, the alternate avenue of swarm optimisation algorithms has very limited research; recent literature reviews list GAs and their predecessors as the sole nature inspired algorithm applied to the BPP [15][16]. Given that swarm algorithms facilitate more flexible representations, unconstrained by operators, this is surprising. Thus, motivated by the recurrent issues of mountainous landscapes and redundant encodings, this report seeks to explore the promising alternative of ACOs.

3 Description of Results

A simple ACO algorithm, using pheromones but no heuristic, was developed to solve BPPs. BPPs were represented as graphs of n layers of k nodes, along with start and end layer with one node. Neighbouring layers were fully connected, with random pheromone assigned to each edge. A population of p ant paths were generated before pheromone updates equal to $\frac{100}{path\,fitness}$ were made to the graph. Pheromone was then multiplied by an evaporation constant e. Populations were recurrently generated until 10,000 fitness evaluations were performed (i.e. $p = 10$ implies 10 times more generations that when $p = 100$). Two BPPs were examined:

- *BPP*1**:** $k = 10$, $n = 500$, $W = \{x : x \in \mathbb{N}, 1 \leq x \leq 500\}$
- *BPP*2**:** $k = 50$, $n = 500$, $W = \{\frac{x^2}{2} : x \in \mathbb{N}, 1 \leq x \leq 500\}$

Results for BPPs 1 and 2 are displayed in tables 1 and 2 respectively.

	$p = 10$		$p = 100$	
	$e = 0.6$	$e = 0.9$	$e = 0.6$	$e = 0.9$
Trial 1	3622	4274	3761	8016
Trial 2	3646	3476	4667	7204
Trial 3	4343	4884	5039	6561
Trial 4	4981	5048	5548	6544
Trial 5	5580	3633	8222	8887
Mean	**4434.4**	**4263**	**5447.4**	**7442.4**

Table 1: Fitness achieved by an ACO algorithm on BPP1 for different values of p and e

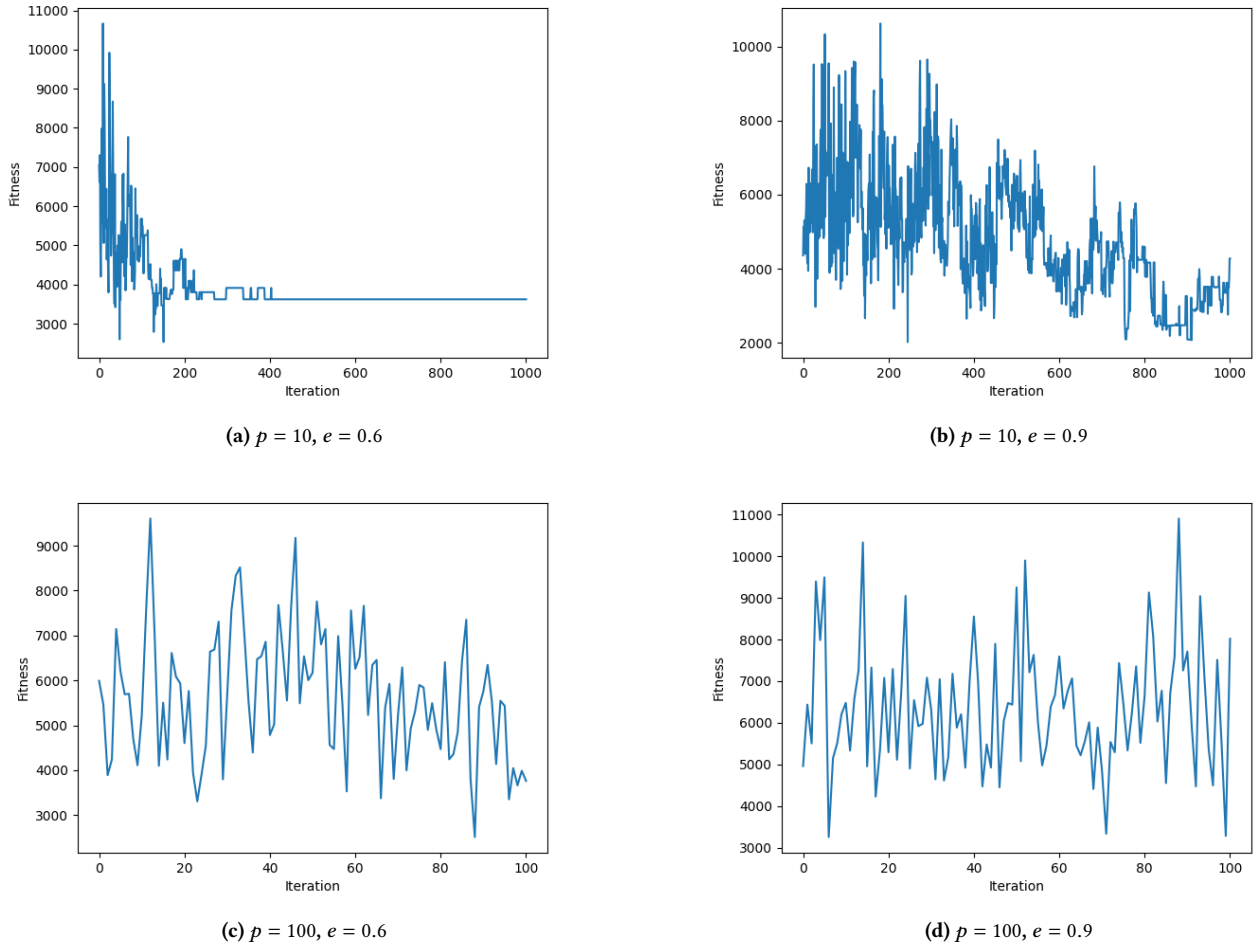
	$p = 10$		$p = 100$	
	$e = 0.6$	$e = 0.9$	$e = 0.6$	$e = 0.9$
Trial 1	693362	594659.5	600363.5	683785.5
Trial 2	650165.5	554008.5	643592	742080
Trial 3	601025.5	567115	856660	804340
Trial 4	585233	620738.5	742246	681950
Trial 5	582338.5	610232.5	704363.5	794110.5
Mean	**622424.9**	**589350.8**	**709445**	**741253.2**

Table 2: Fitness achieved by an ACO algorithm on BPP2 for different values of p and e

Graphs were also generated for trial 1 of each BPP1 parameter set (Figure 1). The graphs show the fitness of the ACO solution against the number of ACO iterations. The graph shapes are representative of the other trials with the same parameter sets for both BPPs.

4 Discussion and Further Work

The results demonstrate that an ACO with $p = 10$ and $e = 0.9$ produces the best results. The parameter p has the greatest effect on solution fitness; $p = 10$ performed significantly better than $p = 100$. $e = 0.6$ performed better than $e = 0.9$ only when $p = 100$,



(a) $p = 10$, $e = 0.6$

(b) $p = 10$, $e = 0.9$

(c) $p = 100$, $e = 0.6$

(d) $p = 100$, $e = 0.9$

Figure 1: Solution fitness against ACO iterations for trial 1 using different parameter sets on BPP1

not when $p = 10$. These patterns were the same for BPP1 and BPP2. The fitnesses of BPP2 solutions was roughly 100 times larger than BPP1 solutions, likely due to the range of item weights in BPP2 being 250,000, 500 times larger than BPP1's range.

p dictates the number of landscape points the ACO can explore before the pheromones are updated and the set of the explorable solutions shrinks. When p is small, the ACO performs a local search; when p is large, many unrelated solutions are created and evaluated. This is often desirable if the landscape is relatively smooth with an obvious optimum. However, as previously established, the BPP has an "exceedingly mountainous" multimodal terrain. Thus, exploring many solutions simultaneously likely results in no obvious optimum, forcing the ACO to explore all solutions in a brute force search [8]. This is clearly seen in Figures 1c and 1d, which show a stochastic fitness progression with little or no convergence.

The parameter e dictates the number of potential solutions "remembered" by the ACO. If e is low, only recently generated solutions are ever considered, causing aggressive optimisation of whatever previous solution was best, even if this is a local optimum. Exploration of other solutions is limited. Thus, we expect a low e to cause rapid convergence to a solution unlikely to be the global optimum. This is seen in Figure 1a, where convergence occurs with over 60% of the iterations remaining and many potential solutions unexplored. Inversely, a high e means all the previous solutions will be fully explored - a brute force approach. This is what causes the Figure slight downward trend in 1c instead of the completely stochastic Figure 1d.

Thus, we can order the parameter sets from creating local search to brute force ACO algorithms: $(p = 10, e = 0.6)$, $(p = 10, e = 0.9)$, $(p = 100, e = 0.6)$, and finally $(p = 100, e = 0.9)$. The optimal set depends largely on the computational resources available. Given just 2000 fitness evaluations, the aggressive $(p = 10, e = 0.6)$ set finds an acceptable solution quickly, while the $(p = 100, e = 0.9)$ set will, given sufficient time, brute force the optimum solution. In

this case, with a termination criterion of 10,000 fitness evaluations, $(p = 10, e = 0.9)$ balances exploration and optimisation to converge upon the final solution as termination is reached (Figure 1b).

To test the effect of search time, the ACO was allowed 100,000 fitness evaluations rather than 10,000 to solve BPP1. As expected, the parameter set $(p = 100, e = 0.6)$, now with sufficient time to search its larger search space, and outperformed $(p = 10, e = 0.9)$ which converged with 50% of the available iterations left (Table 3). It is speculated that the results would be similar for BPP2, but computational limitations prevented that test. Similarly, given sufficient fitness evaluations, $(p = 100, e = 0.9)$ will likely perform the best.

	$p = 10$		$p = 100$	
	$e = 0.6$	$e = 0.9$	$e = 0.6$	$e = 0.9$
Trial 1	2933	2582	1502	6861
Trial 2	4599	1656	2029	3994
Trial 3	4614	3555	1707	5918
Trial 4	4448	4440	1798	5434
Trial 5	4032	3119	2037	5188
Mean	**4125.2**	**3070.4**	**1814.6**	**5479**

Table 3: Fitness achieved by an ACO algorithm on BPP1 for different values of p and e with 100,000 fitness evaluations

This ACO algorithm struggles with several of the key issues encountered by algorithms in the literature review, namely the large amount of redundancy in solution encoding. Like Johnson et al.'s and Jones and Beltramo's representations, the ACO algorithm's graphing representation contains many ways to represent the same partition, with redundancy increasing exponentially with problem size. Furthermore, most of the algorithms discussed in the review incorporate a greedy heuristic which gave preference to the bin choice which resulted in the smallest error. Heuristics are generally beneficial to nature inspired algorithms, as they incorporate global information about the problem's nature to an algorithm operating on a local level. Therefore, algorithms which reduce redundancy and incorporate a heuristic, namely Falkenauer's and Greene's methods, will likely outperform the ACO algorithm which has a larger search space and no access to problem specific information.

It is common for ACO algorithms to incorporate both a heuristic and pheromone element to determine path desirability [6]. As it was anticipated that a heuristic would improve results, a new ACO algorithm which integrated a greedy heuristic was developed. Given a set of possible bin choices, every choice was ranked in ascending order of the resulting error, with the exception of negative error bins which were ranked higher than all other bins to disincentivise over-packing. Bias was added to edges according to the formula $\frac{k-r+1}{k}$, where r is the rank of the corresponding bin, τ is the average pheromone across the set of k available edges, h is and a heuristic weight parameter. The ACO algorithm then proceeded as normal, choosing an edge with a random weighted bias.

The results of this new algorithm on BPP1 for different values of h between 0.25 and 0, parameters $(p = 10, e = 0.9)$, and 10,000 fitness evaluations are shown in Table 4. Contrary to the hypothesis, the addition of a greedy heuristic resulted in significantly inferior solutions, generally worsening with larger heuristic weight, h. Furthermore, as h increased, the results became more stochastic implying degradation to brute force. This suggests that the heuristic was working against the influence of the pheromones. The reason for this is unclear, but, to speculate, the heuristic goal may incentivising rapid growth of one bin to towards the target without growing the others simultaneously. Given unfortunate random pheromone, the rapidly grown bin could become overloaded, despite the heuristic disincentivising such behaviour. Nevertheless, the basic hypothesis that incorporating a heuristic will improve the algorithm's performance remains unchanged, but a better choice of heuristic more suited to ACO is required.

h	0.25	0.20	0.15	0.10	0.05	0.00
Trial 1	4223	8421	6954	8258	6311	4066
Trial 2	6366	5535	8814	5122	7479	5103
Trial 3	7586	7534	6979	9434	3242	4969
Trial 4	5971	7993	7413	4596	5569	3910
Trial 5	10156	5301	10849	5528	6158	3992
Mean	**6860.4**	**6956.8**	**8201.8**	**6587.6**	**5751.8**	**4408**

Table 4: Fitness achieved by a greedy heuristic ACO algorithm with $(p = 10, e = 0.9)$ on BPP1 for different values of h

5 Conclusion

In conclusion, this report produced a literature review exploring the development of nature inspired algorithms for the MNPP, referred to as the BPP. Initial attempts at annealing and GAs were unsuccessful in comparison to the best heuristic algorithm available, the Karmarkar-Karp algorithm. This was due to the significant redundancy in the solution encodings, accompanied with a neighbourhood landscape shown both mathematically by Johnson et al. and anecdotally by Ruml to be "exceedingly mountainous". Later genetic algorithms found greater success by reducing redundancy with chromosomes based on the sets in a solution rather than the items themselves. A distinct lack of research into the application of swarm algorithms to the BPP was also noted, motiving this report.

A simple ant colony optimisation algorithm was produced and trialled with different parameter values for the population p and evaporation e. It was shown that larger values of p and e encourage brute force behaviour, while smaller values led to a local search approach. It was also demonstrated that p was significantly more impactful than e. For the termination criterion of 10,000 fitness evaluations, the parameter set $(p = 10, e = 0.9)$ proved the most effective, converging at termination. It was also demonstrated that for fewer and more fitness evaluations the parameter sets $(p = 10, e = 0.6)$ and $(p = 100, e = 0.6)$ respectively proved more effective, fully utilising the search time available.

Finally, inspired by other work explored in the literature review, an ACO was developed which incorporated a greedy heuristic, one which encouraged bringing bin weights as close to the ideal weight as possible. While it was hypothesised that the inclusion of global information through a heuristic would improve the performance of the ACO, the opposite was found, with result quality decreasing as the influence of the heuristic increased. However, it is still hypothesised that a heuristic more suitable to ACO would facilitate a marked improvement on the pure pheromone algorithm.

References

[1] Emanuel Falkenauer. 1994. A New Representation and Operators for Genetic Algorithms Applied to Grouping Problems. *Evolutionary Computation* 2 (06 1994), 123–144. https://doi.org/10.1162/evco.1994.2.2.123

[2] Emanuel Falkenauer. 1995. Solving Equal Piles with the Grouping Genetic Algorithm. In *Proc. 6th Intnl. Conf. on Genetic Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 492–497.

[3] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, NY, 238.

[4] R. L. Graham. 1969. Bounds on Multiprocessing Timing Anomalies. *SIAM J. Appl. Math.* 17, 2 (1969), 416–429. https://doi.org/10.1137/0117039

[5] William A. Greene. 2000. Partitioning Sets with Genetic Algorithms. In *Proc. 13th Intnl. Florida Artificial Intelligence Research Society Conf.*, James N. Etheredge and Bill Z. Manaris (Eds.). AAAI Press, Orlando, Fl, 102–106.

[6] Frederic Guinand, Patrick Siarry, and Nicolas Monmarché. 2010. *Artificial Ants: from collective intelligence to real-life optimization and beyond*. Wiley.

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

[8] David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, and Catherine Schevon. 1991. Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning. *Operations Research* 39, 3 (1991), 378–406.

[9] Donald R. Jones and Mark A. Beltramo. 1990. *Clustering with genetic algorithms*. Research Report GMR-7156. General Motors Research Laboratories, Warren, MI.

[10] Donald R. Jones and Mark A. Beltramo. 1991. Solving Partitioning Problems with Genetic Algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithm*. Morgan Kaufmann Publishers, San Fransisco, CA, 442–449.

[11] Narendra Karmarkar and Richard M. Karp. 1982. *"The differencing method of set partitioning"*. Technical report UCB/CSD 82/113. University of California, Berkeley, Berkeley, CA.

[12] Bernhard Korte and Jens Vygen. 2006. *Bin-Packing*. Springer Berlin Heidelberg, Berlin, Heidelberg, 426–441. https://doi.org/10.1007/3-540-29297-7_18

[13] Germany Magdeburg. 2005. *The Easiest Hard Problem: Number Partitioning*. Oxford University Press, Magdeburg, Germany, 125. https://doi.org/10.1093/oso/9780195177374.001.0001

[14] Stephan Mertens. 2003. The Easiest Hard Problem: Number Partitioning. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 324–329.

[15] Octavio Ramos-Figueroa, Marcela Quiroz, and Efrén Mezura-Montes. 2022. *Efficient Heuristic Strategies for the Parallel-Machine Scheduling Problem with Unrelated Machines and Makespan Minimization*. Ph. D. Dissertation. Universidad Veracruzana. https://doi.org/10.13140/RG.2.2.17915.90400

[16] Octavio Ramos-Figueroa, Marcela Quiroz-Castellanos, Efrén Mezura-Montes, and Rupak Kharel. 2021. Variation Operators for Grouping Genetic Algorithms: A Review. *Swarm and Evolutionary Computation* 60 (2021), 100796. https://doi.org/10.1016/j.swevo.2020.100796

[17] Wheeler Ruml. 1993. *Stochastic Approximation Algorithms for Number Partitioning*. Technical report TR-17-93. Harvard University, Cambridge, MA.

[18] Jilian Zhang, Kaimin Wei, and Xuelian Deng. 2020. Heuristic algorithms for diversity-aware balanced multi-way number partitioning. *Pattern Recognition Letters* 136 (2020), 56–62. https://doi.org/10.1016/j.patrec.2020.05.022