Human-Derived Heuristic Enhancement of an Evolutionary Algorithm for the 2D Bin-Packing Problem

Nicholas Ross[1(✉)][0000-0003-2970-8312], Ed Keedwell[1][0000-0003-3650-6487], and Dragan Savic[1,2][0000-0001-9567-9041]

[1] University of Exeter, Exeter, EX4 4QF, United Kingdom
[2] KWR, Nieuwegein, Netherlands
nr339@exeter.ac.uk

Abstract. The 2D Bin-Packing Problem (2DBPP) is an NP-Hard combinatorial optimisation problem with many real-world analogues. Fully deterministic methods such as the well-known Best Fit and First Fit heuristics, stochastic methods such as Evolutionary Algorithms (EAs), and hybrid EAs that combine the deterministic and stochastic approaches have all been applied to the problem. Combining derived human expertise with a hybrid EA offers another potential approach. In this work, the moves of humans playing a gamified version of the 2DBPP were recorded and four different Human-Derived Heuristics (HDHs) were created by learning the underlying heuristics employed by those players. Each HDH used a decision tree in place of the mutation operator in the EA. To test their effectiveness, these were compared against hybrid EAs utilising Best Fit or First Fit heuristics as well as a standard EA using a random swap mutation modified with a Next Fit heuristic if the mutation was infeasible. The HDHs were shown to outperform the standard EA and were faster to converge than – but ultimately outperformed by – the First Fit and Best Fit heuristics. This shows that humans can create competitive heuristics through gameplay and helps to understand the role that heuristics can play in stochastic search.

Keywords: Genetic algorithms, Heuristics, Hybridization.

1 Introduction

1.1 Background

There are many real-world cutting and packing problems that have been translated into operational research problems in order to find better solutions. One such problem is the two-dimensional finite bin-packing problem (2DBPP) [1]. This problem requires that a selection of boxes of assorted size are fit into the least number of identically finite-sized bins. Boxes and bins are sized in two dimensions, the boxes may not be cut or overlapped, and the bin's fixed capacity may not be exceeded. Most versions of the problem start with empty bins and the ability to add additional bins as needed. The simplest solution (but least efficient) would be to place every box in a new bin.

More effective heuristics have been developed from other deterministic approaches, such as First Fit, Next Fit, and Best Fit amongst many others [2].

First Fit is perhaps the simplest of these heuristics, in which the selected box is simply placed into the first bin in which it fits. Both First Fit and First Fit Decreasing (in which the boxes are sorted by size before placement) have been found to be competitive approaches to solving the problem [3]. Next Fit functions the same way as First Fit, except that the heuristic starts where the previous iteration finished i.e. if the heuristic places a box in the sixth bin, then the heuristic would start by looking in the seventh bin for a place to put the next box.

The Best Fit heuristic is another competitive approach to the problem [4]. This heuristic searches through all bins to place the box in the bin where the space remaining most closely matches the dimensions of the selected box without violating the bin capacity in either dimension. Other competitive heuristics have been developed by researchers such as the adaptive sequence-based heuristic of Oliveira & Gamboa [5] and the two-dimensional version of the Djang and Finch heuristic developed by López-Camacho et al. [6].

Stochastic methods such as Genetic Algorithms (GAs) have also been applied to the 2DBPP and other bin-packing problems. However, as general-purpose algorithms they struggle to be as competitive as the simpler deterministic heuristic techniques [7, 8]. Grouping Genetic Algorithms [9] applied to the simpler one-dimensional bin-packing problem outperformed the regular GA, while other researchers have used Multi-Objective techniques in a generalised framework to more easily compare against other heuristics and allow the possibility of combining techniques [10].

Combining the deterministic and stochastic approaches into a hybrid EA, hyper-heuristic, or other hybrid heuristic has proven to be a very effective approach. The many different hybrid approaches taken to tackle the 2DBPP include combining the GRASP and VND algorithms [11], combining iterative simulated annealing with binary search [12], combining an improved heuristic with the Variable Neighbourhood Search algorithm [13], combining chaos search with a firefly algorithm [14], and using adversarial self-play for reinforcement learning making use of neural nets and Markov Decision Processes [15].

Hyper-heuristics offer another way of easily combining search-based methods, heuristics, algorithms, and metaheuristics [16]. The work of López-Camacho et al. [17] directly combines a selection of deterministic methods such as Best Fit and First Fit with a GA, while other researchers make use of multi-objective EAs [18] or use an automated approach to design hybrid metaheuristics with a GA [19].

A hybrid EA was formed in Blum & Schmid [20] by combining an EA with a randomised one-pass heuristic, while hybrid GAs have been created by combining a GA with the Best Fit Decreasing heuristic [21], multiple local search heuristics [22], the Crow Search Algorithm [23], or Human-Derived Heuristics (HDHs) [24, 25], all with promising results.

In Ross et al. [25], participants played a gamified version of the 2DBPP and their moves were recorded. Machine learning was then applied to this dataset to obtain decision tree regressor models which provided the HDH. While heuristics are normally either “rules of thumb” employed by those with domain-specific experience or

algorithms built from theory-backed research, HDH are created by learning from how a human interacts with a problem.

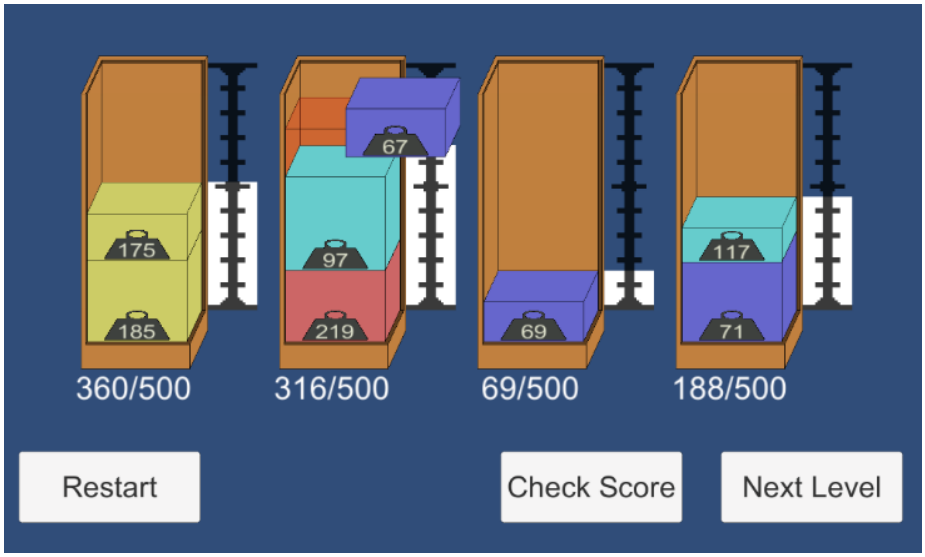

Fig. 1. The bin-packing game being played

1.2 Proposed Approach

Several different HDHs were derived from experimental data and each of them combined in turn to form a hybrid GA. These would be compared with a standard GA and hybrid GAs making use of either the Best Fit or First Fit heuristics. Each HDH would take the form of a decision tree obtained by machine learning on subsets of the data. Four different subsets of data were selected based on the moves made by the humans solving the problem during bin-packing gameplay.

The first and most obvious move set to learn from consisted of All Moves in the data set (HDH_{ALL}). This heuristic learns from moves that improved, worsened, or left the solution unchanged, and it could reasonably be expected to help find good solutions but might take more iterations to do so.

Taking a greedy approach, the second move set to learn from consists of only moves that improved the solution. The Improving Moves heuristic (HDH_{IMP}) should converge faster than the HDH_{ALL} heuristic, though there is a greater danger of getting stuck in a local optimum.

To test the hypothesis that the selection of learning moves has an influence on the performance of the different hybrid GAs, the third set of moves that were learned from consists of only moves that make the solution worse. The Deteriorating Moves heuristic (HDH_{DET}) is expected to be outperformed by the other HDHs, and possibly by the standard GA as well.

The last approach is based on Composite Moves (HDH_{COM}), which are moves that make the solution worse followed by moves that improve the solution. This heuristic

should perform similarly to the HDH_{IMP} heuristic, but with a reduced chance of getting stuck in a local optimum. Each of these heuristics will take the place of the mutation operator in the hybrid GA.

These will be compared against hybrid GAs making use of First Fit heuristics or Best Fit heuristics in place of their mutation operators, and a standard GA that uses a feasible-only random swap mutation based on Next Fit. This latter mutation is necessary to give the standard GA a fair chance when competing with the other heuristics, as all the others will only create feasible mutations. This mutation operates by initially attempting a random swap mutation, but if that would make the solution infeasible then it will try to fit the box into the next bin and repeat until it succeeds.

In Ross et al. [25] it was found that better results were obtained by combining the standard mutation with the HDH mutation in the same algorithm. For this experiment several different proportions of the standard mutation are tested with the HDH, First Fit, and Best Fit heuristics. Each hybrid GA will be tested with proportions of 100%, 99%, 40%, 10%, and 1% heuristic mutation with the balance made up of the standard mutation.

2 Experimental and Computational Details

2.1 Gamification

The 2DBPP problem used in this paper has a fixed number of finite bins and begins with the boxes already randomly distributed between them. The problem contains 10 bins and 20 boxes. The 20 boxes were created by splitting 5 bins, meaning the global optimum is reached by fitting all 20 boxes into just 5 bins. No new bins can be added, and empty bins are not removed.

To achieve a finer detail of scoring than counting the number of used bins, the problem was instead scored by giving a maximum score for each completely empty or exactly full bin, and adding a score for each other bin based on its closeness to being full or empty. The score for each bin was calculated as follows:

$$\begin{aligned} &If\ Bin = Empty\ or\ Bin = Full, \qquad Score = Dim1Max + Dim2Max \\ &Else, \qquad Score = \left|\frac{Dim1Max}{2} - BinDim1\right| + \left|\frac{Dim2Max}{2} - BinDim2\right| \end{aligned} \quad (1)$$

Where $Dim1Max$ and $Dim2Max$ are the maximum size respectively of the first and second dimensions of the bins, and $BinDim1$ and $BinDim2$ represent the current filled proportion of the bin in the first and second dimensions respectively. In the problem used both $Dim1Max$ and $Dim2Max$ were set to the same value of 500, meaning that each empty or full bin scored 1000 and the maximum score was equal to the number of bins in the problem multiplied by 1000. With 10 bins this gives a global optimum score of 10,000. Infeasible solutions were not scored, but their total capacity violation was recorded for the sake of the fitness function.

The data used in this study was obtained through the gamification of a bin-packing problem. Gamification is the process of turning something into a game or making use

of game-like features such as scoring and victory conditions. It has been successfully used to keep player attention and focus on mundane or repetitive tasks [26, 27, 28].

In this case a game was created from a 2DBPP which participants played while their moves were recorded. The game is described in more detail in [25], but essentially consisted of a simple problem with 4 bins and 8 boxes (Fig. 1).

The players were told the objective and shown how the game worked and were then encouraged to compete against each other in solving the problem in the least number of moves. Each move would see the player select a single box in the problem and move it to whichever bin they chose, after which their score would be updated.

Data was gathered from every player that finished the game and solved the problem, regardless of individual performance or the number of moves taken to do so. A total of 10 players completed the simple 4-bin problem. This game data was passed through a method that selected moves for each data set for the machine learning, based on which heuristic was being implemented. This created a data set for each of the four HDHs.

Table 1. Inputs and output for the machine learning. The colour of each input and output matches the colours of their respective nodes in the decision trees.

Inputs				Output
X[0]: Size of Selected Box	X[1]: Maximum Remaining Bin Space	X[2]: Minimum Remaining Bin Space	X[3]: Mean Remaining Bin Space	Target Remaining Bin Space

2.2 Deriving Human Heuristics

Every move that a human player made could be split into two parts; the selection of the box to move and the selection of the bin to place it in. As the First Fit and Best Fit heuristics have no set box selection method but do have a deterministic bin selection, the fairest comparison was to leave the box selection as random for all heuristics and just investigate the moves with regards to target bin selection. This fixed the machine learning output as a quality of the target bin, which was best represented by the amount of space remaining in that bin.

A key step in the process is the determination of the input variables for use in the machine learning approach. The size of the selected box was an obvious variable, and then a number of more general inputs that described the problem space could be included. These would allow the heuristics to be somewhat generalisable across different bin-packing and related problems. The preliminary experimentation, a total of four inputs were selected, each as a sum of the two dimensions.

The first input was the size of the box that the player had selected (X[0]). The second input was the maximum available space remaining in any non-empty bin (X[1]). The third input was the minimum available space in any non-full bin (X[2]), and the last input was the mean space remaining across all partially full bins (X[3]). The output was the available space remaining in the target bin (Table 1).

Machine learning was carried out in Python using *Scikit-learn* [29]. Decision trees were chosen as they give a human-readable insight into the workings of the heuristic. For each HDH a data set with the selected moves was loaded in and an *sklearn* decision tree regressor was trained. To aid readability, each tree was constrained to a minimum leaf node size of 5, a maximum of 12 leaf nodes, and a maximum depth of 6.

The resultant decision trees are shown in Fig. 2. The HDH_{ALL} decision tree that learned from all the moves in the data set is unique in not using the box size (X[0]) input in any of its calculations. The HDH_{IMP} decision tree makes less use of the mean space remaining input (X[3]) than HDH_{ALL}, while the HDH_{DET} decision tree doesn't use it at all. The HDH_{DET} tree is also the only tree that will seek out completely empty bins.

The last tree generated was that for the HDH_{COM} heuristic, which makes more use of the box size input (X[0]) and less use of the minimum bin space remaining input (X[2]) than the others. All four decision trees were exported into a program to be used as mutation operators in a GA.

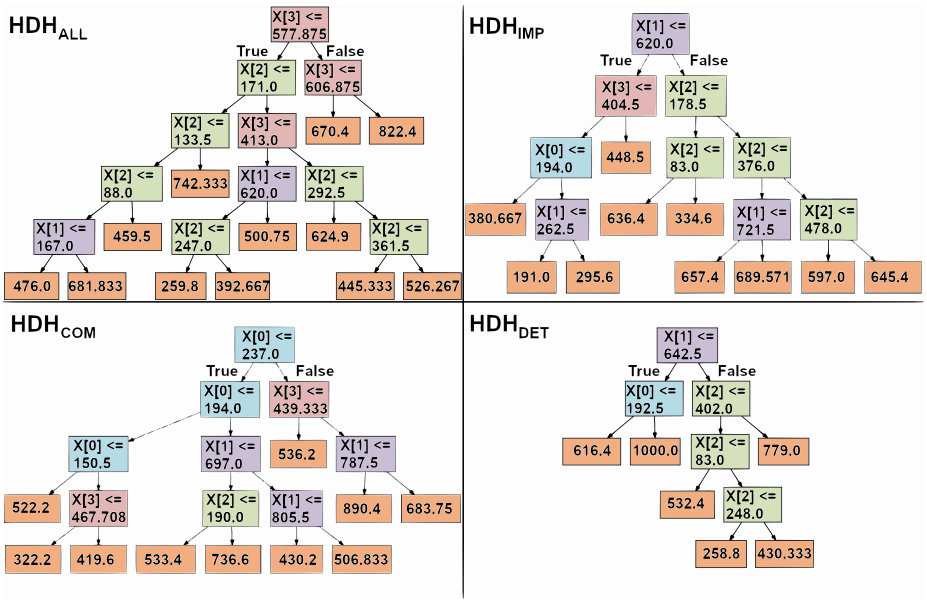

Fig. 2. The decision trees generated for the four different Human-Derived Heuristics. The box colours correspond to the inputs in Table 1.

2.3 Experimental Setup

The GA used for this experiment was a steady-state GA. The problem was represented using *k*-ary encoding with a *k* of 10 and a length equal to 20, the number of boxes in the problem. Each integer value in the encoding represented which bin the box at that index position was in.

The parameters and crossover type were tuned for the standard GA, without using the heuristics. From this initial testing a population size of 100 and a tournament selection method with a tournament size of 2 was chosen. Uniform crossover performed the best, so it was selected, along with a mutation rate of 0.1. As the GA was steady-state, two children would be created and added to the population each generation, and the two least fit members of the population would then be removed.

The fitness function scored both *feasible* solutions and *infeasible* solutions. This worked by first checking if the selected solution exceeded bin capacity and was therefore infeasible; if it was infeasible it would be scored 0 for fitness and then each infeasible bin would be scored based on how much it exceeded the bin capacity limit and added to a violation score. If the solution was feasible it would be scored as mentioned in the introduction in Eq. 1 (with an optimum score of 10,000).

The mutation selected boxes from the child problems of the crossover with a probability of 0.1. For the standard GA the mutation moved the selected box to a bin at random, but if that made that bin infeasible it would then attempt to place the box in the next bin along instead. This would be repeated as needed, looping back round to the start of the bins until the mutation found an appropriately sized bin. This approach was adopted to provide a fairer benchmark for standard mutation. The human players were not permitted to make moves that resulted in infeasible solutions and this mutation operator performs the same function for the standard GA.

For the HDH mutation the decision tree determines the bin into which the randomly selected box should be placed, based on the closest match to the determined space remaining. For First Fit and Best Fit mutations their respective heuristics were applied, with First Fit searching from the start until it found the first bin that could fit the selected box, and Best Fit searched the entire problem space for the bin that had the closest space remaining to match the box size. The hybrid GA mutation had a chance of implementing either the heuristic mutation or the standard mutation.

The GA for each condition was run for 200,000 iterations on 30 different instances of the 10-bin problem, with each different problem instance being repeated 30 times for a total of 900 runs per condition.

3 Results and Discussion

The fittest result in the population at each iteration of the GA was recorded and then averaged across all 900 runs. These mean fittest values were then plotted for each HDH against the values from the standard GA (Fig. 3).

3.1 Human-Derived Heuristics vs Standard GA

3.1.1 HDH_{ALL}. 100% HDH_{ALL} converged early to a local optimum and failed to progress further, being quickly overtaken by the other percentages and the standard GA. 99% HDH_{ALL} achieved the highest fitness by the end of the run and showed faster convergence than the standard GA, with 40% HDH_{ALL} following a similar pattern. 10% HDH_{ALL} was slower to converge but ended close behind 40% HDH_{ALL} and still

ahead of the standard GA. 1% HDH_{ALL} outperformed the standard GA and ended at a similar level of fitness as the other percentages, though it converged slower.

3.1.2 HDH_{IMP}. The 100% HDH_{IMP} condition quickly converged to a suboptimal solution and was outperformed by the other conditions including the standard GA while the 99% HDH_{IMP} and 40% HDH_{IMP} conditions converged fastest and ended with the fittest solutions. 10% HDH_{IMP} and 1% HDH_{IMP} performed similarly, though 10% HDH_{IMP} converged the faster of the two. All HDH_{IMP} heuristics except for 100% HDH_{IMP} outperformed the standard GA.

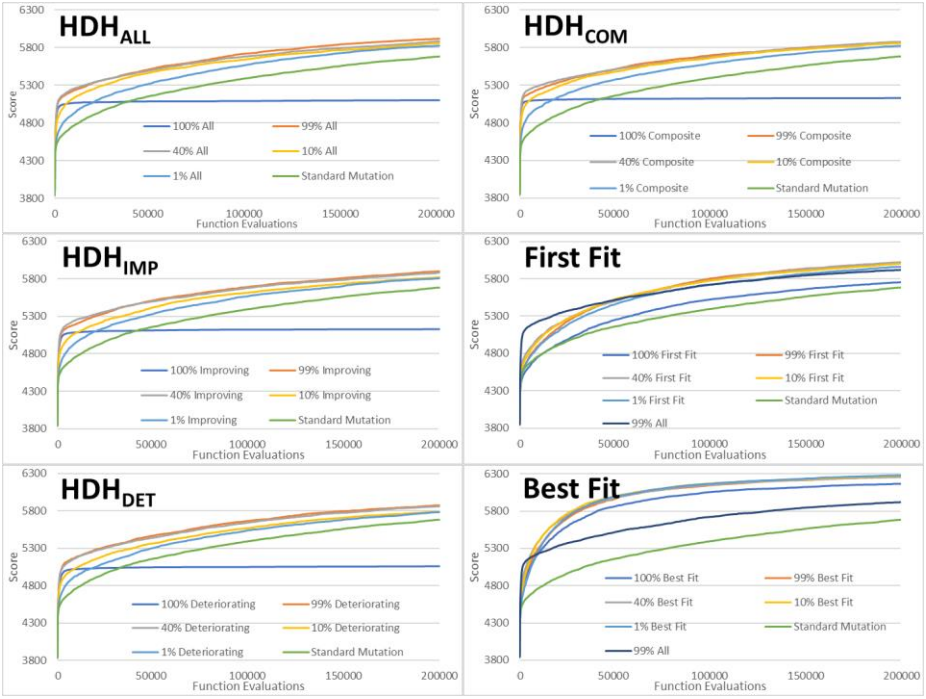

Fig. 3. Mean fittest solution per generation for each condition.

3.1.3 HDH_{DET}. Though the 100% HDH_{DET} converged quickly it was still outperformed by the standard GA. All other HDH_{DET} conditions managed to outperform the standard GA, with the 99% HDH_{DET} heuristic performing best with 40% HDH_{DET} a close second. The 10% HDH_{DET} and 1% HDH_{DET} conditions performed at an intermediate level between the standard GA and the 40% and 99% HDH_{DET} heuristics, with the 10% condition performing slightly better.

3.1.4 HDH_{COM}. The 100% HDH_{COM} heuristic also converged very early to a local optimum and then failed to progress further. The 99% HDH_{COM}, 40% HDH_{COM}, and 10% HDH_{COM} performed almost identically, except that the 40% HDH_{COM} heuristic converged faster and the 10% HDH_{COM} heuristic converged slightly slower. The 1% HDH_{COM} heuristic lagged slightly behind the others, but all HDH_{COM} percentages except for 100% outperformed the standard GA.

Table 2: Mean and maximum fittest scores attained over the 900 runs. The mean values that differ significantly from the standard GA are denoted by asterisks (*) and the highest value for mean and maximum fittest score are highlighted in bold and underlined.

		Mean Fittest Score	Statistical Comparison vs Standard GA	Maximum Fittest Score
Standard GA		5683	N/A	7374
HDH_{ALL}	100%	5102*	t(1762)=24.0 p<.001	6516
	99%	**5920***	t(1774)=-11.2 p<.001	7362
	40%	5886*	t(1789)=-9.4 p<.001	7718
	10%	5860*	t(1788)=-8.2 p<.001	7724
	1%	5827*	t(1792)=-6.6 p<.001	7310
HDH_{IMP}	100%	5129*	t(1774)=23.2 p<.001	6640
	99%	5901*	t(1792)=-10.0 p<.001	7574
	40%	5882*	t(1788)=-9.2 p<.001	7392
	10%	5822*	t(1792)=-6.4 p<.001	7186
	1%	5807*	t(1798)=-5.7 p<.001	7124
HDH_{COM}	100%	5130*	t(1778)=23.3 p<.001	6572
	99%	5876*	t(1798)=-8.8 p<.001	**8208**
	40%	5875*	t(1788)=-8.9 p<.001	7824
	10%	5859*	t(1785)=-8.2 p<.001	7238
	1%	5824*	t(1798)=-6.5 p<.001	7504
HDH_{DET}	100%	5060*	t(1775)=26.2 p<.001	6690
	99%	5876*	t(1780)=-9.0 p<.001	7706
	40%	5864*	t(1787)=-8.4 p<.001	7626
	10%	5788*	t(1798)=-4.8 p<.001	7734
	1%	5786*	t(1798)=-4.6 p<.001	7042

The final fittest score results for all 900 runs of each HDH condition were then compared statistically against the standard GA. For every HDH percentage condition an F-Test was performed comparing the variance with the standard GA results, followed by a two-factor t-Test. These results can be found in Table 2.

Every condition differed significantly from the standard GA, with the 100% HDH_{ALL}, 100% HDH_{IMP}, 100% HDH_{DET}, and 100% HDH_{COM} all performing significantly worse, and all other HDH percentages performing significantly better. 99% HDH_{ALL} achieved the highest mean fittest score and 99% HDH_{COM} found the highest maximum fittest score across all runs. Within each HDH condition the 99% heuristic reached the highest mean fittest score while the 100% heuristic performed the worst.

The 99% HDH_{ALL} result did not perform significantly better than 40% HDH_{ALL} ($t(1798)=-1.6$, $p = .10$), but did perform significantly better than 10% HDH_{ALL} ($t(1798)=-3.0$, $p = .003$) and 1% HDH_{ALL} ($t(1792)=-4.5$, $p < .001$).

The 99% HDH_{IMP} heuristic saw no significant difference compared against 40% HDH_{IMP} ($t(1798)=-0.9$, $p=.38$), but significantly outperformed the 10% HDH_{IMP} ($t(1798)=-3.7$, $p<.001$) and 1% HDH_{IMP} heuristics ($t(1798)=-4.4$, $p<.001$).

The 99% HDH_{DET} heuristic saw no significant difference in performance against 40% HDH_{DET} ($t(1798)=-0.6$, $p=.55$) but performed significantly better than 10% HDH_{DET} ($t(1791)=-4.2$, $p<.001$) and 1% HDH_{DET} ($t(1785)=-4.2$, $p<.001$).

The 99% HDH_{COM} heuristic did not significantly outperform either the 40% HDH_{COM} ($t(1798)=0.04$, $p=.96$) or 10% HDH_{COM} ($t(1798)=0.83$, $p=0.41$), but performed significantly better than 1% HDH_{COM} ($t(1798)=2.45$ $p=.01$).

Comparing the mean highest scoring heuristics from each condition against each other found the 99% HDH_{ALL} to perform significantly better than the 99% HDH_{DET} ($t(1798)=-2.2$, $p=0.03$) and 99% HDH_{COM} ($t(1787)=-2.1$, $p=0.04$) heuristics but not the 99% HDH_{IMP} ($t(1792)=-0.9$, $p=.35$) heuristic.

Table 3: Mean and maximum fittest scores attained over the 900 runs. The mean values that differ significantly from the standard GA (*) and 99% HDH_{ALL} (†) are marked. Highest values for mean and maximum fittest score are highlighted in bold and underlined.

		Mean Fittest Score	Statistical Comparison		Maximum Fittest Score
			vs Standard GA	vs HDH_{ALL}	
Standard GA		5683	N/A		7374
HDH_{ALL}	99%	5920*	t(1774)=-11.2 p<.001	N/A	7362
First Fit	100%	5754*†	t(1760)=-2.9 p=.003	t(1684)=-7.1, p<.001	8456
	99%	6011*†	t(1729)=-16.1 p<.001	t(1785)=4.8, p<.001	7718
	40%	6022*†	t(1782)=-15.8 p<.001	t(1798)=5.1, p<.001	8678
	10%	6004*†	t(1784)=-15.0 p<.001	t(1798)=4.2, p<.001	8214
	1%	5961*†	t(1789)=-12.8 p<.001	t(1798)=2.0, p=.04	7644
Best Fit	100%	6165*†	t(1798)=-22.1 p<.001	t(1791)=11.9, p<.001	**10000**
	99%	6268*†	t(1701)=-29.1 p<.001	t(1770)=18.5, p<.001	8218
	40%	6256*†	t(1685)=-28.7 p<.001	t(1760)=18.1, p<.001	8208
	10%	6271*†	t(1744)=-28.5 p<.001	t(1792)=18.1, p<.001	**10000**
	1%	**6281*†**	t(1733)=-29.1 p<.001	t(1787)=18.8, p<.001	**10000**

3.2 Human-Derived Heuristics vs First Fit

The results for the First Fit heuristic can be seen in Fig. 3, plotted against the standard GA and 99% HDH_{ALL} heuristic. All percentage conditions of First Fit performed sig-

nificantly better than the standard GA (see Table 3), though the 100% First Fit heuristic did not perform as well as the others and was outperformed by 99% HDH_{ALL}. The First Fit heuristic converged slower than 99% HDH_{ALL}, but all except 100% First Fit eventually reached a higher fitness. 100% First Fit scored significantly less than 99% $HDH_{ALL,}$ but the others all scored significantly higher than 99% HDH_{ALL} (Table 3).

3.3 Human-Derived Heuristics vs Best Fit

The results for the Best Fit heuristic can be seen in Fig. 3, with the standard GA and 99% HDH_{ALL} heuristic for comparison. 100% Best Fit was the worst performing of them, though the Best Fit heuristics were faster to converge than the First Fit heuristics, and highest scoring of all the heuristics tested. Every Best Fit heuristic significantly outperformed both the standard GA and the 99% HDH_{All} condition as well (see Table 3), though the Human-Derived Heuristic was faster to converge than the others.

3.4 Discussion

Although there were minor differences in performance between them, each HDH, regardless of the dataset source appeared to perform in a similar manner. In terms of application level, 100% HDH rapidly converged to a local optimum and was overtaken by the others as expected, with the 99% and 40% conditions performing the best followed by the 10% condition and finally the 1% condition.

The 99% condition showed that employing even a small amount of random mutation was enough to prevent the HDH getting stuck in a local optimum. Conversely, and surprisingly, the 1% condition showed that employing only a small amount of deterministic mutation was enough to significantly improve the standard GA.

The Deteriorating Moves (HDH_{DET}) heuristic performed better than expected but still achieved the lowest mean fittest scores of the four HDHs. It is likely that amongst the moves that the decision tree learned were useful moves that made the solution temporarily worse but created an opening for better moves.

The Composite Moves (HDH_{COM}) heuristic was the fastest to converge, but both the HDH_{ALL} and the HDH_{IMP} heuristics achieved better results (although the Composite Moves heuristic achieved the highest maximum score from any of the HDHs). The Composite Moves heuristic used in this paper was a single decision tree, but a true Composite Moves heuristic might be better represented by two trees; one tree to make the moves that make the solution temporarily worse and a second to improve it.

The Improving Moves heuristic performed almost as well as the All Moves heuristic, with very little difference between them except that the All Moves heuristic reaching a higher mean fittest fitness score by the end of the run.

When compared to the First Fit heuristic the HDHs performed competitively at the start of the run but were eventually outperformed by all but 100% First Fit. The 100% First Fit heuristic performed only a little better than the standard GA, and it was surprising to see the other First Fit percentages performing significantly better than it.

The Best Fit heuristics performed very well against the HDHs, though again the 100% Best Fit condition performed poorest. The Best Fit heuristics were the only ones to reach the global optimum, though on isolated runs and not for all conditions.

The 1% Best Fit heuristic had both the highest mean fittest score of any of the heuristics and found the global optimum. This result shows that just a small amount of a competitive deterministic heuristic can have a strong effect on a GA.

The only advantage enjoyed by the HDHs when compared to the Best Fit heuristic is a slightly faster convergence rate. This fast convergence could be useful if the HDH was combined with another heuristic or was incorporated into a hyper-heuristic that could take advantage of the different capabilities at its disposal.

Although HDHs were outperformed by the established heuristics, an area where HDHs would be useful is on problems that don't have existing established heuristics such as Best Fit. Learning a heuristic from human interactions with a previously unseen problem is easier than attempting to create new rules of thumb. Furthermore, many real-world problems will not have accompanying heuristics and so the HDH methodology might be used to create them.

Future work could see these heuristics tested on other problems, and other heuristics developed from similar problems compared against these. Combining several Human-Derived Heuristics and more traditional heuristics into a hyper-heuristic might yield even more promising results.

4 Conclusions

In this study machine learning was used on four different data sets to create four different Human-Derived Heuristics (HDHs). Each of them was developed from human players solving a gamified version of a small 2D bin-packing problem.

The four HDHs were then in turn combined with a hybrid GA as the mutation operator, with the GA utilising the HDH either 100%, 99%, 40%, 10%, or 1% of the time during the run and the remainder of the time using a random swap mutation modified with a Next Fit heuristic. The First Fit and Best Fit heuristics were then executed in the same process and the results compared.

For the HDHs the 100% heuristics performed poorly, but the other conditions all performed significantly better than the standard GA. Several of the HDHs also outperformed the 100% First Fit heuristic, but the other First Fit and all the Best Fit conditions were able to outperform the HDHs.

Surprisingly, using either 1% of a stochastic mutation or 1% of a deterministic mutation with 99% of the other resulted in better results than 100% of either alone.

Acknowledgments

This work was supported by Skipworth Engelhardt Asset Management Strategists Limited (SEAMS) and the Human-Computer Optimisation for Water Systems Planning and Management (HOWS) project funded by the Engineering and Physical Sciences Research Council (EPSRC) – grant EP/P009441/1.

References

1. G. Wäscher, H. Haußner, and H. Schumann, ‘An improved typology of cutting and packing problems’, European Journal of Operational Research, vol. 183, no. 3, pp. 1109–1130, Dec. 2007.
2. J. O. Berkey and P. Y. Wang, ‘Two-Dimensional Finite Bin-Packing Algorithms’, J Oper Res Soc, vol. 38, no. 5, pp. 423–429, May 1987, doi: 10.1057/jors.1987.70.
3. G. Dósa and J. Sgall, ‘First Fit bin packing: A tight analysis’, in 30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013), Dagstuhl, Germany, 2013, vol. 20, pp. 538–549, doi: 10.4230/LIPIcs.STACS.2013.538.
4. G. Dósa and J. Sgall, ‘Optimal Analysis of Best Fit Bin Packing’, in Automata, Languages, and Programming, Berlin, Heidelberg, 2014, pp. 429–441, doi: 10.1007/978-3-662-43948-7_36.
5. Ó. Oliveira and D. Gamboa, ‘Adaptive Sequence-Based Heuristic for the Two-Dimensional Non-guillotine Bin Packing Problem’, in Hybrid Intelligent Systems, Cham, 2020, pp. 370–375, doi: 10.1007/978-3-030-14347-3_36.
6. E. López-Camacho, G. Ochoa, H. Terashima-Marín, and E. K. Burke, ‘An effective heuristic for the two-dimensional irregular bin packing problem’, Ann Oper Res, vol. 206, no. 1, pp. 241–264, Jul. 2013, doi: 10.1007/s10479-013-1341-4.
7. E. Falkenauer and A. Delchambre, ‘A genetic algorithm for bin packing and line balancing’, in Proceedings 1992 IEEE International Conference on Robotics and Automation, 1992, pp. 1186–1192 vol.2, doi: 10.1109/ROBOT.1992.220088.
8. G. T. Lam, V. A. Ho, D. Logofatu, and C. Badica, ‘Considerations on using genetic algorithms for the 2D bin packing problem: A general model and detected difficulties’, in 2017 21st International Conference on System Theory, Control and Computing (ICSTCC), 2017, pp. 303–308, doi: 10.1109/ICSTCC.2017.8107051.
9. T. Kucukyilmaz and H. E. Kiziloz, ‘Cooperative parallel grouping genetic algorithm for the one-dimensional bin packing problem’, Computers & Industrial Engineering, vol. 125, pp. 157–170, Nov. 2018, doi: 10.1016/j.cie.2018.08.021.
10. F. Luo, I. D. Scherson, and J. Fuentes, ‘A Novel Genetic Algorithm for Bin Packing Problem in jMetal’, in 2017 IEEE International Conference on Cognitive Computing (ICCC), 2017, pp. 17–23, doi: 10.1109/IEEE.ICCC.2017.10.
11. F. Parreño, R. Alvarez-Valdes, J. F. Oliveira, and J. M. Tamarit, ‘A hybrid GRASP/VND algorithm for two- and three-dimensional bin packing’, Ann Oper Res, vol. 179, no. 1, pp. 203–220, Sep. 2010, doi: 10.1007/s10479-008-0449-4.
12. S. Hong, D. Zhang, H. C. Lau, X. Zeng, and Y.-W. Si, ‘A hybrid heuristic algorithm for the 2D variable-sized bin packing problem’, European Journal of Operational Research, vol. 238, no. 1, pp. 95–103, Oct. 2014, doi: 10.1016/j.ejor.2014.03.049.
13. D. Zhang, Y. Che, F. Ye, Y.-W. Si, and S. C. H. Leung, ‘A hybrid algorithm based on variable neighbourhood for the strip packing problem’, J Comb Optim, vol. 32, no. 2, pp. 513–530, Aug. 2016, doi: 10.1007/s10878-016-0036-6.
14. C. Zhao, L. Jiang, and K. L. Teo, ‘A hybrid chaos firefly algorithm for three-dimensional irregular packing problem’, Journal of Industrial & Management Optimization, vol. 16, no. 1, p. 409, 2020, doi: 10.3934/jimo.2018160.
15. A. Laterre, Y. Fu, M. K. Jabri, A.--S. Cohen, D. Kas, and K. Hajjar, ‘Ranked Reward: Enabling Self-Play Reinforcement Learning for Bin packing’, p. 10, 2019.
16. N. Pillay and R. Qu, ‘Packing Problems’, in Hyper-Heuristics: Theory and Applications, N. Pillay and R. Qu, Eds. Cham: Springer International Publishing, 2018, pp. 67–73.

17. E. López-Camacho, H. Terashima-Marín, and P. Ross, ‘A hyper-heuristic for solving one and two-dimensional bin packing problems’, in Proceedings of the 13th annual conference companion on Genetic and evolutionary computation, Dublin, Ireland, 2011, pp. 257–258, doi: 10.1145/2001858.2002003.
18. J. C. Gomez and H. Terashima-Marín, ‘Evolutionary hyper-heuristics for tackling bi-objective 2D bin packing problems’, Genet Program Evolvable Mach, vol. 19, no. 1, pp. 151–181, Jun. 2018, doi: 10.1007/s10710-017-9301-4.
19. A. Hassan and N. Pillay, ‘Hybrid metaheuristics: An automated approach’, Expert Systems with Applications, vol. 130, pp. 132–144, Sep. 2019, doi: 10.1016/j.eswa.2019.04.027.
20. C. Blum and V. Schmid, ‘Solving the 2D Bin Packing Problem by Means of a Hybrid Evolutionary Algorithm’, Procedia Computer Science, vol. 18, pp. 899–908, Jan. 2013, doi: 10.1016/j.procs.2013.05.255.
21. M. A. Kaaouache and S. Bouamama, ‘Solving bin Packing Problem with a Hybrid Genetic Algorithm for VM Placement in Cloud’, Procedia Computer Science, vol. 60, pp. 1061–1069, Jan. 2015, doi: 10.1016/j.procs.2015.08.151.
22. M. Beyaz, T. Dokeroglu, and A. Cosar, ‘Hybrid Heuristic Algorithms for the Multiobjective Load Balancing of 2D Bin Packing Problems’, in Information Sciences and Systems 2015, Cham, 2016, pp. 209–220, doi: 10.1007/978-3-319-22635-4_19.
23. S. Laabadi, M. Naimi, H. El Amri, and B. Achchab, ‘A Crow Search-Based Genetic Algorithm for Solving Two-Dimensional Bin Packing Problem’, in KI 2019: Advances in Artificial Intelligence, Cham, 2019, pp. 203–215, doi: 10.1007/978-3-030-30179-8_17.
24. M. B. Johns, H. A. Mahmoud, D. J. Walker, N. D. F. Ross, E. C. Keedwell, and D. A. Savic, ‘Augmented evolutionary intelligence: combining human and evolutionary design for water distribution network optimisation’, in Proceedings of the Genetic and Evolutionary Computation Conference, Prague, Czech Republic, 2019, pp. 1214–1222, doi: 10.1145/3321707.3321814.
25. N. D. F. Ross, M. B. Johns, E. C. Keedwell, and D. A. Savic, ‘Human-evolutionary problem solving through gamification of a bin-packing problem’, in Proceedings of the Genetic and Evolutionary Computation Conference Companion, Prague, Czech Republic, 2019, pp. 1465–1473, doi: 10.1145/3319619.3326871.
26. A. Darejeh and S. S. Salim, ‘Gamification Solutions to Enhance Software User Engagement—A Systematic Review’, International Journal of Human–Computer Interaction, vol. 32, no. 8, pp. 613–642, Aug. 2016, doi: 10.1080/10447318.2016.1183330.
27. B. Morschheuser, J. Hamari, and J. Koivisto, ‘Gamification in Crowdsourcing: A Review’, in 2016 49th Hawaii International Conference on System Sciences (HICSS), 2016, pp. 4375–4384, doi: 10.1109/HICSS.2016.543.
28. A. Suh, C. Wagner, and L. Liu, ‘Enhancing User Engagement through Gamification’, Journal of Computer Information Systems, vol. 58, no. 3, pp. 204–213, Jul. 2018, doi: 10.1080/08874417.2016.1229143.
29. F. Pedregosa et al., ‘Scikit-learn: Machine Learning in Python’, Journal of Machine Learning Research, vol. 12, p. 2825−2830, Oct. 2011.