Learning the Quality of Dispatch Heuristics Generated by Automated Programming

Andrew J. Parkes, Neema Beglou, and Ender Özcan

School of Computer Science, University of Nottingham, Nottingham, NG8 1BB, UK

Abstract. One of the challenges within the area of optimisation, and AI in general, is to be able to support the automated creation of the heuristics that are often needed within effective algorithms. Such an example of automated programming may be performed by search within a space of heuristics that will be applied to a target domain. In this, brief proof-of-concept, paper, we consider the case of online bin-packing as the target domain, and consider the potential for machine learning methods to aid the associated automated programming problem. Simple numerical 'policy matrices' are used to represent heuristics, or 'dispatch policies', controlling the placement of item into bins as they arrive. We report on an initial investigation of the potential for neural nets to analyse and classify the resulting 'policy matrices', and find strong evidence that simple nets can be trained to learn to predict which heuristics, expressed as policy matrices, exhibit better or worse fitness. This gives the potential for them to be used as a surrogate fitness function to enhance the usage of search algorithms for finding heuristics. It also supports the prospect of using machine learning to extract the patterns that lead to successful heuristics, and so generate explanations and understanding of machine-generated heuristics.

1 Introduction

The effective solution of many real-world combinatorial optimisation problems, such as scheduling, timetabling, resource assignment, etc., requires good heuristics to be used within the heart of many algorithms. Usually, these heuristics are generated by human experts - however, the rise of powerful methods in search and AI holds the potential for them to be created automatically. To further this aim, in this paper, we consider the specific and simple case of online bin packing: items arrive one at a time, and immediately, and irrevocably need to be assigned to a bin, either an existing bin with sufficient space, or by opening a new bin, and the goal is simply to minimise the number of bins are used. The decision process then requires a 'dispatch policy' or 'heuristic' to decide which bin should be used. The quality of a heuristic can be evaluated by simply using it to pack many items and then taking the average fullness of the bins - usually expressed as a percentage, and higher is better. A natural way to do this is with an 'index policy' in which a score is given to each packing option based on: the size s of the item under consideration, and the remaining space, r, within the bin, and then the highest scoring option is taken. In the 'CHAMP' approach [3, 1], Parkes and Özcan introduced a "policy matrix" representation, denoted as $M[s,r]$ to store the scores, and so a matrix constitutes a heuristic. Then standard search methods (evolutionary algorithms or metaheuristics) are used to find good values for those matrices. (Note that the associated search problem is in the space of heuristics and not the direct space of solutions to the target domain.) Earlier work [2] in this area used genetic programming (GP) to build a numeric score function $f(s, r)$ as an arithmetic tree of the relevant inputs; however, the automatically-generated matrix policies heuristics significantly improved upon the GP approaches. A surprise of the CHAMP work was that the resulting matrices were rather 'spiky', and did not have the smooth structure that was usually (implicitly) assumed in human-generated heuristics or those arising in the GP approaches. Accordingly, the table-lookup representation seems to offer the potential for search-based methods to create new kinds of heuristics. However, a disadvantage is that the resulting policy matrices can be hard to interpret – it is not immediately evident which matrices are good, and which are bad. It is then natural to ask whether machine learning methods can identify the structures or patterns in such "machine-discovered heuristics", and so give insight into what properties a heuristic ought to have. Given that the policy matrices are two-dimensional matrices, and so similar to 2-d pictures, it particularly natural to ask whether systems used in machine vision might be applicable, and in particular to try neural nets. Accordingly, in this (short proof-of-concept) paper, we show that indeed neural nets can be trained to distinguish between good and bad matrices. Furthermore, even simple nets do well, and this gives evidence for a potential for use of neural nets (or machine learning in general) to be used for the automated generation and understanding of heuristics.

2 Experimental Results

In this paper, we will use an example of online bin-packing as in previous work [3]; "UBP(20,5,10)", with bins of capacity 20, and item sizes selected uniformly at random from the range [5,. . .,10]. See [3, 1] for explicit examples of some resulting matrices. For this paper, the relevant entries in each matrix can be simply regarded as a set of integers (the size of the set happens to be 57). Furthermore, we restrict the value of each entry to be in $\{1, 2, 3\}$; this restriction still allows matrices of high fitness. The set of such numbers then define the heuristic. The standard heuristics, such as "best-fit" and "first-fit" can easily be represented as matrices and give a bin fullness of around 92%; however, using a GA much better matrices can be found with a fullness of up to about 98%. The aim here is to determine whether machine learning methods can be used to predict and aid understanding the fitness of such matrices.

For the study, we used a combination of random generation and local search to generate a diverse dataset of over 150k matrices, each with the associated fitness[1].

[1] We plan to make the dataset available at an appropriate time

g-mean 86.4%	Predicted P:[94-100]	Predicted N:[92-94]
Actual P:[94-100]	982 76.7%	298 23.3%
Actual N:[92-94]	300 2.4%	12063 97.6%

Table 1. Single Layer MLP using classes N:[92-94]–P:[94-100].

g-mean 93.3%	Predicted P:[94-100]	Predicted N:[92-93]
Actual P:[94-100]	1190 91.8%	106 8.2%
Actual N:[92-93]	161 1.8%	8781 98.2%

Table 2. Single Layer MLP using classes N:[92-93]–P:[94-100].

In order to train a neural net there were two primary decision: how to give the matrices as input to the net, and the structure of the net. Firstly, the input is done using a standard 'one-hot' scheme - for each matrix entry, each of the possible values from $\{1, 2, 3\}$ is selected using a set of 3 $\{0, 1\}$ inputs. Hence, each matrix is converted to 3*57=171 binary inputs. (Other schemes were tried but were less effective.) Secondly, for the neural net structure, we used a simple feed-forward Multi-layer Perceptron (MLPs), with one hidden layer. We followed a common heuristic that each layer should be about half the size of the previous one, so the hidden layer was give 171/2=85 nodes After some initial experiments we decided on sigmoid activation functions for the hidden layer. The system was implemented using DeepLearning4J[2] and the training parameters were fairly standard e.g. using stochastic gradient descent. Again other options were tried, but were no more effective.

This proof-of-concept paper just aims to give evidence that neural nets have the ability to detect patterns in the matrices that are correlated with their fitness. Hence, instead of doing regression we just did binary classification. It is usual to discuss results in terms of "Positive (P)" and "Negative (N)" classes, and we take the P class to correspond to the better matrices. (The net is trained so that the output gives the class memberships.) As usual, we will report confusion matrices. The positive class will usually have fewer instances, and so we need to also use performance measures that are appropriate when the class sizes are imbalanced [4]; the accuracy would be a poor measure, hence we also report the standard geometric mean 'g-mean' of the true positive and true negative rates.

Due to lack of space, we just give representative results using two pairs of classes. Since we are mostly interesting in good matrices then it is reasonable to restrict to instances with fitness of at least 92%; Table 1 shows results when the two classes are then formed from a threshold at fitness 94%. This case is arguably challenging, as the range of fitness values is small, and there will be fewer of the very bad matrices that are (presumably) easier to spot. However, the performance is still good, with 77% of those in the higher-fitness class being recognised. This case has 'adjacent classes' with a boundary at 94%, and so small differences in the fitness could lead matrices to be placed in separate classes; such boundary cases are likely to be more difficult to learn, and so results could be unduly pessimistic. Accordingly, we put a small gap between

[2] `https://deeplearning4j.org/`

the two classes by excluding the fitness range [92%-93%], giving a second case with two classes N:[92-93]–P:[94-100]. The results in Table 2 indicate that the classification performance did then improve significantly; now 92% of the higher fitness class was recognised, again with a very good value for the g-mean.

3 Conclusions and Future Work

We considered the issue of finding and identifying good heuristic dispatch policies for the online bin-packing problem. The heuristics are represented numerically by an integer matrix, with the pertinent entries being scores that are used during the decision process of where to allocate items as they arrive. We investigated whether or not a neural net can predict the quality of the matrices. On converting to binary classification problems, the results clearly indicate that nets with just one hidden layer can achieve high discriminating and predictive power. This indicates that the properties of the matrix that lead to high fitness are indeed susceptible to being learned. The first natural potential impact is that such learned model could be used as a surrogate fitness value during a search in the space of matrices; being used to select matrices more likely to perform well when evaluating their true fitness.

Arguably more importantly, a potential impact is using a learned model in order to generate understanding of the space of heuristics (matrices). Inspections of the weights in the trained nets should be at least able to indicate which of the entries of the matrix are considered most important, and deeper analysis might also reveal the conditions needed on the relevant values. For this, it is particularly interesting that even just a single hidden layer gives good results; as such simple nets are likely to be much easier to analyse. Future work will look at other methods, such as SVM (with appropriate kernels) and decision trees, as this might well give results that are easier to interpret. Such understanding will hopefully also lead to better insight of how to search for good heuristics.

Overall, the main novelty and contribution of this study is to show that (standard) machine learning can be applied in an intriguing way to study the space of 'special purpose' heuristic functions that are used in dispatch policies, or in optimisation algorithms.

References

1. Asta, S., Özcan, E., Parkes, A.J.: CHAMP: Creating Heuristics viA many Parameters. Expert Systems with Applications 63, 208–221 (2016)
2. Burke, E.K., Hyde, M.R., Kendall, G., Woodward, J.R.: The scalability of evolved on line bin packing heuristics. In: 2007 IEEE Congress on Evolutionary Computation. pp. 2530–2537 (Sept 2007)
3. Özcan, E., Parkes, A.J.: Policy matrix evolution for generation of heuristics. Proc. of the 13th ann. conf. on Genetic and evolutionary computation (GECCO) (2011)
4. V. López and A. Fernández and S. Garca and V. Palade and F. Herrera: An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. Information Sciences 250, 113 – 141 (2013)