Dimension Reduction in the Search for Online Bin Packing Policies

Shahriar Asta Ender Özcan Andrew J. Parkes

School of Computer Science
University of Nottingham
Nottingham, NG8 1BB, U.K.
{sba,exo,ajp} @cs.nott.ac.uk

ABSTRACT

In online bin-packing problems, a policy must be found for assigning items, according their size, immediately upon their arrival to bins with known initial capacities. In previous work of Özcan and Parkes (GECCO 2011), a policy was represented as a 2-dimensional 'matrix' (array) and good matrices were then evolved using a genetic algorithm (GA). Here, we consider a form of dimensional reduction in which variables in the matrix are grouped into elements taken from one-dimensional vectors. We find that with the right form of grouping, the GA then typically finds such 'vector policies' significantly more quickly, and yet suffers little loss of overall quality.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: ARTIFICIAL INTELLIGENCE—*Problem Solving, Control Methods, and Search*

General Terms

Algorithms

Keywords

cutting and packing, representations, hyper-heuristic

1. INTRODUCTION

In the online bin-packing problem decisions must be made for assigning items of various sizes immediately upon their arrival to bins with known initial capacity, but possibly already containing previous items. In work by Özcan and Parkes [1] online bin packing was considered for the case in which all item sizes and bin capacities were integer. (We refer the reader to that paper for background and notations). This allowed the decisions to be made using an index policy represented as a 2-dimensional array or 'policy matrix'. The fitness of the matrix was simply determined by using it to explicitly pack a (long) stream of items and recording the overall average fractional fullness of the bins. Good matrices were then evolved using a genetic algorithm (GA). The matrix method had the advantage of completeness as all index policies can be represented without any bias to any particular functional form, and their work showed that it was possible to use a GA to find policies that could beat standard heuristics. However, the resulting number of entries in the matrix, and the associated number of variables in the chromosome of the GA, was typically quadratic in the maximum capacity of bins. Consequently, the matrix method has the potentially severe disadvantage of needing a large number of independent variables, and a consequence of this could be that the large search space increased the time needed for the GA to find good solutions. In this paper, we consider a form of dimensional reduction in which variables in the matrix are aggregated into elements of one-dimensional vectors in various forms and so we refer to this as using 'policy vectors'. Good vectors are then evolved using a GA in a similar fashion to that used for the full matrices. This reduces the number of variables from quadratic to linear in the bin capacity, and so has the potential advantage to reduce the search time. However, it also imposes strong constraints on the policies that can be represented, thus the potential disadvantage to significantly lower the quality of the policies that are achievable. Here we briefly report on the resulting time versus quality tradeoff.

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.
ACM 978-1-4503-1964-5/13/07.

2. EVOLUTION OF POLICY VECTORS

In deciding the form of the dimensional reduction from a matrix 'M' to a vector or vectors here we use two main decisions. The first is the class 'A', 'B' or 'C' which differ depending on whether or not the entries for the empty bin are fixed, connected to the other entries or are allowed to evolves separately.

$$\begin{aligned} A: M[r,i] &= V_P[p(r,i)] \quad r < C \\ M[r,i] &= 1 \qquad\quad r = C \\ B: M[r,i] &= V[p(r,i)] \quad \forall r \\ C: M[r,i] &= V_P[p(r,i)] \quad r < C \\ M[r,i] &= V_e[i] \quad r = C \end{aligned} \tag{1}$$

The other decision is represented by the function 'p' giving the way that vectors are mapped to a matrix, and which we take to be one of four mappings: row-wise, column-wise, diagonal and anti-diagonal. (We assume that some needed offset in the ranges is done implicitly to convert them to standard 0-based arrays).

$$
\begin{aligned}
row : p(r,i) &= i \\
col : p(r,i) &= r \\
dia : p(r,i) &= r' = r - i \\
adi : p(r,i) &= r + i
\end{aligned} \tag{2}
$$

This gives a total of $3 * 4 = 12$ choices for dimensional reduction that were studied.

3. EXPERIMENTAL DESIGN AND RESULTS

We use the standard train and test system, with 12 rounds of train/test experiments being performed. Each experimental round consists of a single-run training session on each of the 10 UBP instances (where a UBP instance is a combination of bin capacity and item size ranges are described in [1]) and with the largest being UBP$(150, 20, 100, 10^5)$. This is then followed by a testing phase in which the trained vector mapped into a matrix policy is taken and applied to the corresponding bin-packing instance for 101 trials, and this is used to evaluate its quality - which is simpler the average bin fullness achieved. Different seeds are considered for the training as well as each of the test sessions resulting in different sequences of instances.

It should be noted that with 10^5 items being packed the variation between different runs is insignificant - variations arise primarily between different runs of the training, but again are generally small. Experiments showed that, given the same amount of training time, almost always the diagonal policy vector belonging to class C performs better than the policy matrix.

In order to clarify this we performed experiments to study the quality of the results as functions of the number of iterations or the time spent during the training. This also allowed to determine whether given a lot more time, the matrices produce results that are far better than any vector can achieve.

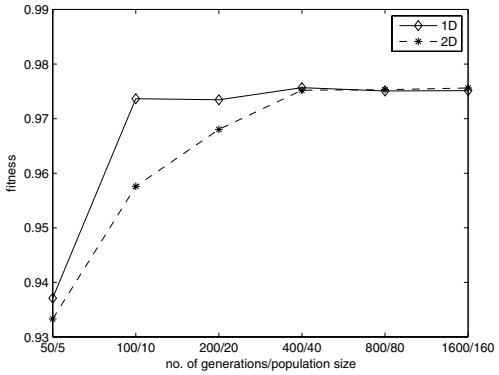
We omit the experimental details, but just show some results for a single instance ($UBP(40, 10, 20, 10^5)$). in figure 1(a). We see that the vector clearly outperforms the matrix in terms of the solution quality for all except the very longest runs. This clearly demonstrates the potential computational advantage of using vector policies. It is somewhat more surprising that even in very long runs the matrix does not improve significantly on the restriction to a vector policy.

We also found an interesting subtlety: Whilst evolving the vectors, some individuals are so inferior in terms of quality that it takes a much longer time to evaluate the generated solution. This is due to the fact that such a solution often prioritizes opening new bins which in turn forces the algorithm to examine an increasingly large number of open bins prior to decision making.

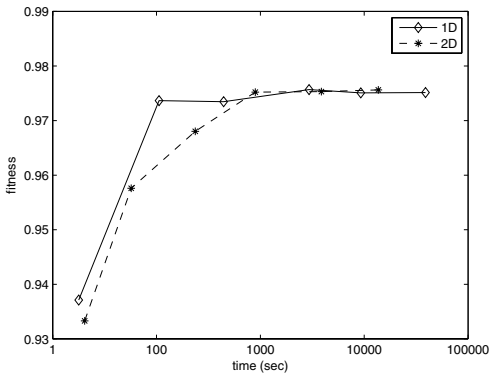
Based on this observation (arguably) a better comparison between the fitness values acquired by the matrix and the vector is provided as a function of time (figure 1(b)). In this case we see the advantage of the vector policy is not as strong, but is still significant.

4. CONCLUSIONS

The online 1-d bin-packing problem has been studied, following the work of [1] where the index policies were represented as a simple direct lookup matrix of values. This allowed a GA to produce good solutions, however had the disadvantage of requiring a large number of variables; typically quadratic in the bin capacities. A way to reduce the number of variables is simply to no longer allow all the matrix entries to be individual variables, but instead have many matrix entries be generated from the same variable. In this study, we investigated into different forms of aggregation of the policy matrix elements. Though there are many ways to aggregate policy matrix elements the empirical results show that even simple strategies have great potential in that with no, or minimal, loss of solution quality it can be possible to generate powerful solutions.



(a)



(b)

Figure 1: (a) fitness comparison (x axis: generations / population size). (b) fitness comparison as a function of time (x axis: seconds on a log-scale).

5. REFERENCES

[1] E. Özcan and A. J. Parkes. Policy matrix evolution for generation of heuristics. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 2011–2018, New York, NY, USA, 2011. ACM.