Grammar-based Selection Hyper-heuristics for Solving Irregular Bin Packing Problems

Alejandro Sosa-Ascencio
Tecnologico de Monterrey
Eugenio Garza Sada 2501
Monterrey, NL. México
a01061994@itesm.mx

Hugo Terashima-Marín
Tecnologico de Monterrey
Eugenio Garza Sada 2501
Monterrey, NL. México
terashima@itesm.mx

José C. Ortiz-Bayliss
Tecnologico de Monterrey
Eugenio Garza Sada 2501
Monterrey, NL. México
jcobayliss@itesm.mx

Santiago E. Conant-Pablos
Tecnologico de Monterrey
Eugenio Garza Sada 2501
Monterrey, NL. México
sconant@itesm.mx

ABSTRACT

This article describes a grammar-based hyper-heuristic model for selecting heuristics to solve the two-dimensional bin packing problem (2D-PBB) with irregular pieces and regular objects. We propose to use a genetic programming approach to generate rules for selecting one suitable heuristic according to the features that characterize the problem state. The experiments confirm the idea that the results produced by the proposed approach are able to rival those obtained by some heuristics described in the literature.

Keywords

Hyper-heuristics, Grammar-based Genetic Programming, Bin Packing

1. INTRODUCTION

Cutting and packing are general problems with many variants that include the 2D-BPP. Because of its complexity, this problem is usually solved by using heuristics. But, such heuristics tend to be very specialized to certain types or classes of instances of the problem. A recent trend to improve the performance of heuristics considers the use of hyper-heuristics [2]. In the context of our investigation, hyper-heuristics are methods that manage a set of heuristics and suggest which one to apply based on the current problem state under exploration.

We used genetic programming (GP) as the high-level strategy to generate rules that decide which heuristic to apply at each moment of the solution process.

GECCO '16 July 20-24, 2016, Denver, CO, USA

ACM ISBN 978-1-4503-4323-7/16/07.

DOI: http://dx.doi.org/10.1145/2908961.2908970

2. RELATED WORK

Several heuristics have been designed for the 2D-BPP. *Selection heuristics* determine the piece to be packaged and the object in which the current piece should be packed. The available selection heuristics for our framework are first fit decreasing (FFD), filler (FIL), best fit decreasing (BFD) and Djang and Fitch (DJD). A second type of heuristics, known as *placement heuristics*, states the way in which the piece is located inside the object. In this investigation we have used a constructive approach wit maximum adjacency (CAD) as placement heuristic. More details on these heuristics can be consulted at [6, 7].

Ross et al. [7] proposed a hyper-heuristic system for tackling one-dimensional bin packing benchmark problems using an XCS classifier that associates characteristics of the current state of a problem with a specific heuristic. Burke et al. [3] proposed a genetic programming hyper-heuristic approach that generates constructive heuristics for the two-dimensional strip packing problem.

The application of grammar-based approaches as hyper-heuristic methods is a relatively new trend. Bader-El-Den and Poli [1] developed a GP framework that evolves grammar components to generate effective incremental SAT solvers. Burke et al. [4] used grammatical evolution (GE) for evolving local search heuristics for 1D-BPP.

3. HYPER-HEURISTIC FRAMEWORK

Our framework relies on a genetic programming module that evolves rules that determine when to apply a particular heuristic during the solution process. The initial population of the evolutionary algorithm is randomly generated. Unlike other grammar-based systems, our genotype is directly represented as a tree structure that is recursively constructed when the individual is evaluated.

We designed a grammar composed of two main types of components: non-terminal functions and terminal functions. Non-terminal functions can be decision functions (if then else) or relational operators ($>$, $\geq$, $<$, $\leq$, $=$). Terminal functions may return numerical constants, selection heuristics, placement heuristics, combined heuristics, or feature extractors. Feature extractors return a value related to some feature that describes the current problem state, which may

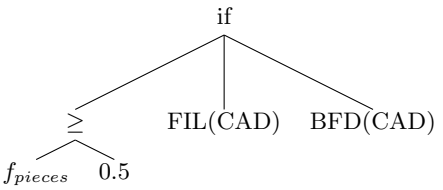

Figure 1: An example of a rule that may be generated by the proposed framework.

change every time a new piece is packed into an object. We chose eight problem extractors used in a similar work to describe the 2D-BPP instances [6] considered for this investigation. In all the cases, the values returned by the feature extractors were normalized in the range [0,1]. For example, Fig. 1 depicts a an example rule that may be generated by our framework. In this example, if at least half of the pieces in the problem are pending for packing, the algorithm will use BFD and CAD. Otherwise it will use FIL and CAD.

4. EXPERIMENTS

The solver used in this investigation is fully implemented in Java. The algorithm generates a solution from scratch by packing one piece at a time into objects that are empty at the beginning. Once a piece is packed it will not be revised for further improvements. When the solver tries to pack a piece into an object, it will try rotating the piece by multiples of 90 degrees (0, 90, 180 and 270). All the objects are squares and have the same size.

540 two-dimensional randomly generated instances [5] that contain only convex polygonal pieces were divided into a training and a testing set. The training set is composed of half of the instances while the remaining instances compose the testing set. The instances have been balanced according to their features to produce two similar sets of 270 instances each.

Each run of the hyper-heuristic framework results in one new rule for applying the heuristics. In each run, we used a steady-state evolutionary process with a population size of 30 individuals along 80 generations. The individuals have a crossover probability of 0.9 and a mutation probability of 0.05. The maximum depth of trees in the grammar-based GP system was set to six levels. We produced 15 rules as the result of 15 runs of the framework.

5. RESULTS

We used the *usage* (the number of open objects for each instance) as metric to evaluate the performance of the heuristics in this investigation. Every rule produced by our framework was compared against the best performer among the heuristics described in Sect. 2.

The rules produced by our framework tend to replicate the best heuristic per instance on the testing set. The worst rule produced was able to use as few objects as the best heuristic per instance on 95.93% of the instances. 33% of the rules generated by our approach were as competent as the best performing heuristic per instance in 97.04% of the instances in the testing set. These results confirm our initial idea that a grammar-based hyper-heuristic framework is a feasible strategy to combine the strengths of existing heuristics into one single method for covering a wider range of instances than with traditional approaches.

6. CONCLUSION

This work presents the first ideas on a grammar-based hyper-heuristic framework for tackling the 2D-BPP with irregular pieces and regular objects. The rules generated with our approach have a competitive performance compared with the best performing heuristic for each particular instance in a testing set. More investigation is needed to understand the consequences of the grammar in the quality of the results. At this point, we expect that grammars designed with a greater diversity of functions should provide a better performance than those with a more restricted set of functions. As future work, we want to explore other hyper-heuristic approaches for the 2D-BPP, for example, the approach that decomposes heuristics to produce new heuristics without the intervention of human experts.

7. ACKNOWLEDGMENTS

This research was supported in part by ITESM Research Group with Strategic Focus in Intelligent Systems and CONACyT Basic Science Project under grant 241461.

8. REFERENCES

[1] M. Bader-El-Den and R. Poli. A GP-based hyper-heuristic framework for evolving 3-SAT heuristics. *Proceedings of the 9th annual conference on Evolutionary Computation*, page 1749, 2007.

[2] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, pages 1–30, 2013.

[3] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. Woodward. A classification of hyper-heuristic approaches. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 449–468. Springer US, 2010.

[4] E. K. Burke, M. R. Hyde, G. Kendall, and J. Woodward. Automating the packing heuristic design process with genetic programming. *Evolutionary Computation*, 20(1):63–89, 2012.

[5] E. López-Camacho, H. Terashima-Marín, and P. Ross. A hyper-heuristic for solving one and two-dimensional bin packing problems. In *13th annual conference companion on Genetic and evolutionary computation*, GECCO '11, pages 257–258, New York, NY, USA, 2011. ACM.

[6] E. López-Camacho, H. Terashima-Marín, P. Ross, and G. Ochoa. A unified hyper-heuristic framework for solving bin packing problems. *Expert Syst. Appl.*, 41(15):6876–6889, 2014.

[7] P. Ross, J. G. Marín-Blázquez, S. Schulenburg, and E. Hart. Learning a procedure that can solve hard bin-packing problems: A new GA-based approach to hyper-heuristics. In *Conference on Genetic and Evolutionary Computation. Lecture Notes in Computer Science*, volume 2724, pages 1295–1306. Springer-Verlag, 2003.