Evolving Scheduling Policies through a Genetic Programming Framework

Christos Dimopoulos
University of Sheffield
Dept. of Automatic Control & Systems Engineering
Mappin Street, S1 3JD, U.K.
cop97cd@sheffield.ac.uk

Ali MS Zalzala
Heriot-Watt University
Department of Computing & Electrical Engineering
Edinburgh, EH14 4AS, U.K.
a.zalzala@hw.ac.uk

1 INTRODUCTION

In this paper we investigate the potential use of Genetic Programming for the solution of the one-machine total tardiness problem, which has been the subject of academic research for almost four decades. To our knowledge, no previous effort has been made to solve static scheduling problems in a GP-framework, in contrast to other evolutionary computation techniques that have been extensively used for this scope.

2 EXPERIMENTAL PHASE

It is difficult to evolve a permutation representation without producing infeasible solutions when subtree crossover and mutation are employed. Two different ways of approaching the problem were tested during the experimental phase. First, a combination of dispatching rules was utilised as an indirect way of representing a permutation through a GP-framework. The problem of variable length was dealt by considering only the part of the program that had significance on the solution of the problem. The algorithm was trained on experimental set-ups comprised of a number of problems with significant differences in their parameters. Four combinations of dispatching rules were evolved as scheduling policies, each one for a particular value of *n* (total number of jobs). Their performance was largely dominated by the presence of the MON (Montagne) rule, which is a problem-specific dispatching rule. However, there was an indication that for large-sized problems an evolved combination might produce a better overall performance than the application of the MON rule itself. Secondly, a traditional GP-framework was employed as a basis for evolving a formula of a dispatching rule for the one-machine total tardiness problem. Nine different dispatching rules were evolved during the experimental phase of the algorithm. Most of the evolved rules performed quite well on a very large set of previously unseen problems (160 in total). Based on this observation we can safely say that these rules did not just fit the data of the fitness cases but they contained information that was relevant to the problem considered. In addition, the evolved dispatching rules clearly outperformed MON in the majority of cases. This paper is part of a broader research that investigates the potential use of Genetic Programming for the solution of the one-machine total tardiness problem. The algorithm described in the first part of the experimental phase has also been employed as a heuristic procedure for the solution of single one-machine total tardiness problems. Detailed results are presented in (Dimopoulos & Zalzala, 1999).

3 CONCLUSIONS

We can summarise the conclusions of our investigation in the form of the following points:

- The evolution of a direct permutation representation in a GP-framework is not straightforward, however, a problem-specific indirect representation can be sometimes constructed.
- It is sensible to employ the MON dispatching rule as a scheduling policy in a plant where tardiness problems of variable characteristics are encountered. Applying combinations of dispatching rules as scheduling policies may be beneficial in some cases, but overall the improvement is not significant.
- Evolving scheduling policies in the form of dispatching rule formulas is a procedure more natural to genetic programming. While high quality results can be achieved, evolved individuals provide us with significant insights to the nature of the scheduling problem.

Acknowledgments

The first author would like to thank Greek State Fund (I.K.Y.) for its support.

References

[Dimopoulos & Zalzala, 1999] Dimopoulos, C. and Zalzala, A.M.S., “A genetic programming heuristic for the one-machine total tardiness problem”, accepted for publication in *Congress on Evolutionary Computation, CEC ’99.*