A Learning Metaheuristic for the Multi Mode Resource Constrained Project Scheduling Problem

Tony Wauters[1], Jannes Verstichel[1], Katja Verbeeck[1], and Greet Vanden Berghe[1,2]

[1]KaHo St.-Lieven, Department of Engineering
Gebr. Desmetstraat 1, 9000 Gent, Belgium, Email:
{Tony.Wauters,Jannes.Verstichel,Katja.Verbeeck,Greet.VandenBerghe}@kahosl.be

[2]K.U.Leuven, Department of Computer Science
Celestijnenlaan 200A, 3001 Leuven (Heverlee), Belgium

Abstract. This abstract introduces a novel approach to intelligently select appropriate modes for each activity in the multi mode resource constrained project scheduling problem. The mode of an activity determines its duration and the resources that it requires. In order to obtain a good (or optimal) makespan, finding an appropriate mode for each activity is most important.
The approach integrates different components. It applies a serial forward/backward heuristic to construct schedules; learning automata to come up with good quality modes and a genetic algorithm to explore the search space.

Keywords: MRCPSP - Learning Automata - Genetic Algorithms

1 Introduction

The Multi Mode Resource Constrained Project Scheduling Problem (MRCPSP) is an extension to the Resource Constrained Project Scheduling Problem (RCPSP). Given a project with a certain number of scarce (non) renewable resources and a set of non preemptive activities, each with a set of predecessors, a set of successors and resource requirements, the goal is to find a feasible assignment that minimizes the makespan. The (M)RCPSP is subject to two constraints: 1) an activity should not be scheduled until all its predecessors have finished (precedence constraint) and 2) the number of assignments of a resource at any time should not be larger than the availability of that resource (resource constraint). In the MRCPSP, an activity can have a fixed number of modes. Each mode defines the number of the resources required to finish the activity within a given execution time. A feasible schedule for the MRCPSP is determined by the mode and the start time of each activity, provided that all precedence and availability constraints are satisfied.

The RCPSP is known to be NP-hard. A comprehensive survey of the project scheduling problem can be found in [2, 6]. A tabu search metaheuristic is applied in [5] for solving the MRCPSP with generalized precedence relations. In [4, 10, 7] a genetic algorithm is presented for the MRCPSP.

The contribution of this paper is the introduction of a novel approach to the MRCPSP that combines ideas from reinforcement learning and metaheuristics. In contrast with [3], which applies a genetic algorithm to tune parameters, we integrate learning automata within the genetic algorithm itself. A learning automaton is a simple decision making device, which, unlike a genetic algorithm, does not assume complex parameter selection. Moreover, learning automata show interesting convergence behaviour in stationary environments [11].

This abstract is structured as follows. Section 2 describes the components of the reactive metaheuristic. Section 3 presents experimental results on a set of benchmark problems. Finally, Section 4 concludes the paper and points out directions for further research.

2 A Hybrid Optimisation Approach

The hybrid approach combines three components to create good quality solutions. We first present the algorithm to generate a solution. In Section 2.2, we present the method that learns which modes contribute best to a small makespan. Finally, we describe how the components are integrated into a genetic algorithm.

2.1 Serial Schedule Generator

We construct a solution with the iterative forward/backward scheduling technique [9]. It applies an ordered activity-list and a mode assigned to each activity. When an activity is scheduled without violating any constraint, the algorithm continues with the next activity. In case a particular activity cannot be scheduled (e.g. when it requires more than the available number of resources), an infeasible assignment results in a very large makespan ($M + r$). M is a large number (related to the maximum scheduling horizon of the project) and r is the sum of all the resource deficits.

2.2 Learning Automata

In [11] some classical reinforcement learning algorithms for Finite Action Learning Automata (FALA) are described. These algorithms learn to take the best action out of a set of possible actions following a reinforcement signal from the environment. For learning the best modes of every given priority list we have chosen the Linear Reward-Inaction (LRI) method because of its ϵ-optimality property in all stationary environments. An action probability vector $p(k)$ as described below is maintained and updated:

$$\begin{aligned} p_i(k+1) &= p_i(k) + \lambda\beta(k)(1 - p_i(k)), && \text{if action } i \text{ was chosen at } k \\ p_j(k+1) &= p_j(k) - \lambda\beta(k)p_j(k), && \forall j \neq i \end{aligned}$$

with k the current iteration number, $\lambda \in [0, 1]$ the reward parameter and $\beta(k) \in \{0, 1\}$ the reinforcement obtained when choosing action i.

The mode learning method works as follows. For every activity in the list, a learning automaton chooses a mode out of a fixed set of predefined modes according to its probability vector. By mapping the mode to each activity, a new solution can be generated. We compare the new solution to the current best solution. In case of an improvement, we set it as the new best solution and perform a positive reinforcement ($\beta = 1$) update of all the learning automata for the mode that they have chosen. In case the new solution was not better, we update all the learning automata with a negative reinforcement ($\beta = 0$). At the end we return the schedule corresponding to the best solution found.

2.3 Genetic Algorithm

In this section we present the hybrid approach that is based on the genetic algorithm described in [8]. We integrate the mode learning method into the genetic algorithm and learn the appropriate activity modes for every individual solution in the population. The learning phase should not require many evaluations as that would increase the computation time too much. We call this the short learning phase. At the end of every generation we apply a long learning phase (with a higher number of learning iterations) to the best individual. If the long learning phase generates a better solution than the best so far, we store the new one. Experimental results are presented in Section 3.

3 Results

We performed tests on all instances of the MRCPSP J30 PSPLib library [1]. The test instances involve 30 multiple mode activities. We carried out the tests with the following parameters: population size 10, 5 generations, 5.000 iterations with a learning rate of 0.2 for the short learning phase, and 10.000 iterations with a learning rate of 0.1 for the long learning phase. We compared our resulting makespans with the best found for these instances. The results for all MRCPSP j30 instances are presented in Figure 1. Our method performs equally good as the best known method for 314 out of 552 instances.

4 Conclusion

We developed a new hybrid metaheuristic method for the MRCPSP. It integrates a learning component into an existing metaheuristic that performs well on RCPSP, in order to make it fit for the MRCPSP. State-of-the-art solutions are very often reproduced, which is very promising, given the preliminary state of the research. Future work will involve fine tuning all the components in order to improve the overall behaviour of the approach for a larger set of test problems.

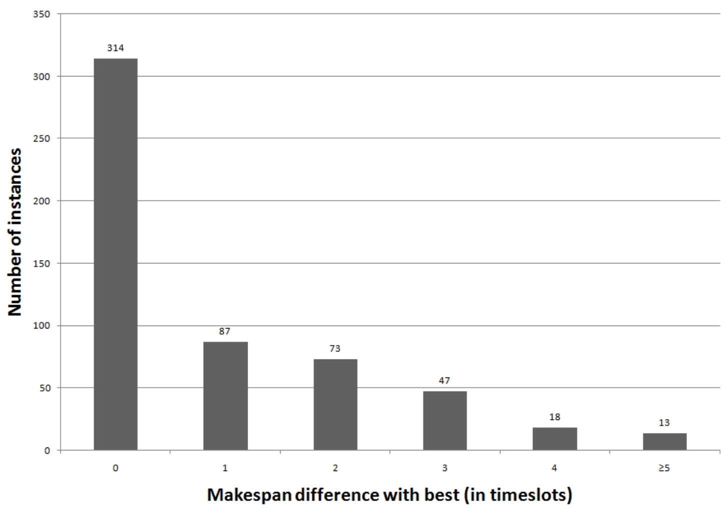

Fig. 1. Results for all MRCPSP j30 instances

References

1. Project scheduling problem library - psplib - http://129.187.106.231/psplib/.
2. P. Brucker, A. Drexl, R. Mohring, K. Neumann, and E. Pesch. Resource-constrained project scheduling: Notation, classification, models and methods. *EJOR*, 112:3–41, 1999.
3. E.K. Burke, P. De Causmaecker, G. De Maere, J. Mulder, M. Paelinck, and G. Vanden Berghe. A time-window approach to investigate improvement of robustness objectives in airline schedules. *Computers and Operations Research*. Special issue on Robustness.
4. S. Hartmann. Project scheduling with multiple modes: a genetic algorithm. *Annals of Operations Research*, 102:111–135, 1997.
5. W. Herroelen and B. De Reyck. The multi-mode resource-constrained project scheduling problem with generalized precedence relations. *EJOR*, 119:538–556, 1999.
6. W. Herroelen, B. De Reyck, and E. Demeulemeester. Resource-constrained project scheduling: a survey of recent developements. *Computers and Operations Research*, 25:297–302, 1998.
7. J.Alcaraz and C. Maroto. A new genetic algorithm for the multi-mode resource-constrained project scheduling problem. page 4, 2002.
8. R. Kolisch and S. Hartmann. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, 174:23–37, 2006.
9. K.Y. Li and R.J. Willis. An iterative scheduling technique for resource-constrained project scheduling. *EJOR*, 56:370–379, 1992.
10. M. Masao and C.C. Tseng. A genetic algorithm for multi-mode resource constrained project scheduling problem. *EJOR*, 100:134–141, 1997.
11. M.A.L. Thathachar and P.S. Sastry. *Networks of Learning Automata Techniques for Online Stochastic Optimization*. Kluwer Academic Publishers, 2004.