Using Swap-Based Search Trees to obtain Solutions for Resource Constrained Project Scheduling Problems

Maximilian Bügler[1]* and **André Borrmann**[1]

[1] Chair of Computational Modelling and Simulation, Arcisstraße 21, 80333 München, Germany

The high level of complexity in modern construction projects causes the problem of scheduling the involved tasks under precedence and resource constraints to be a great challenge. Additionally applications on real construction sites require non linear and non continuous constraints and objective functions. In order to optimize large scale construction sites in practice, this paper proposes a new optimization approach based on creating search trees from swapping and delaying tasks. The convergence of the proposed algorithm is analyzed and its complexity is compared to other available algorithms.

1 Introduction

The planning of construction projects is a very ill-structured, knowledge-intensive task due to the complicated, interactive, and dynamic nature of construction processes [1]. Deterministic approaches based on linear programming require very large numbers of constraints for a complete representation of the problem, especially when large numbers of resources are available [2]. Therefore, those algorithms are not feasible for large scale problems. Furthermore non-linear constraints or objective functions additionally increase the complexity. Heuristic methods for this type of problem include genetic algorithms [3], simulated annealing [4], ant colony optimization [5], and particle swarm optimization [6]. We introduce a new heuristic optimization technique that is based on *discrete-event simulation* [7] and utilizes *feasible task swaps and delays*, that cause changes to the sequence of the tasks within a schedule [8].

2 Discrete-Event Simulation

In order to obtain initial solutions and to apply and evaluate task swaps and delays a discrete-event simulation model following the Activity-based construction approach [9] is used. This simulation model in its default configuration only creates non-delay schedules, where each task is executed at the earliest feasible time instance. As this does not necessarily cover the full space of possible schedules the notation of delays is introduced. A delayed task is skipped at the earliest possible time instance. Multiple delays for one task are possible while tasks that do not overlap with any other tasks in time cannot be delayed. An example for a problem where a delay is required to reach the optimum is illustrated in Figure 1. In this example task D can be executed after task A has finished, which then blocks task C from being executed right after task B has finished. In order to reach the optimum it is required to wait for task B to finish, then start task C right away.

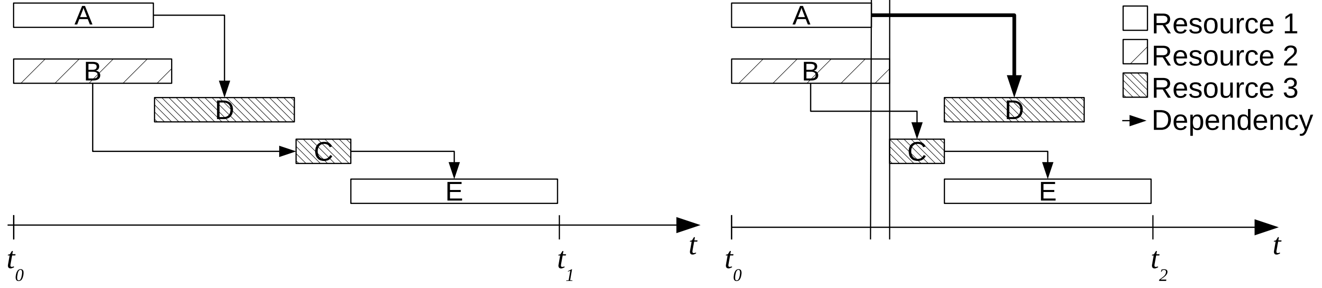

Fig. 1: Left: Only possible non-delay schedule for this problem; Right: Shorter schedule after delaying task D.

Once an initial schedule has been created using predetermined priorities or at random, task swaps and delays can be applied to create different schedules. The feasible swaps and delays are collected during the simulation of each schedule. A swap of two tasks is feasible if and only if the execution of the first task blocks the second task from being executed and vice versa such that a concurrent execution is not possible at the respective time instance. Lemma 2.1 implies that searching the space of possible schedules by applying swaps and delays to an initial schedule is complete.

* Corresponding author: e-mail max.buegler@tum.de, phone +49 89 289 25023, fax +49 89 289 25051

Lemma 2.1 *Every feasible schedule can be reached from any other feasible schedule by applying a series of swaps and delays.*

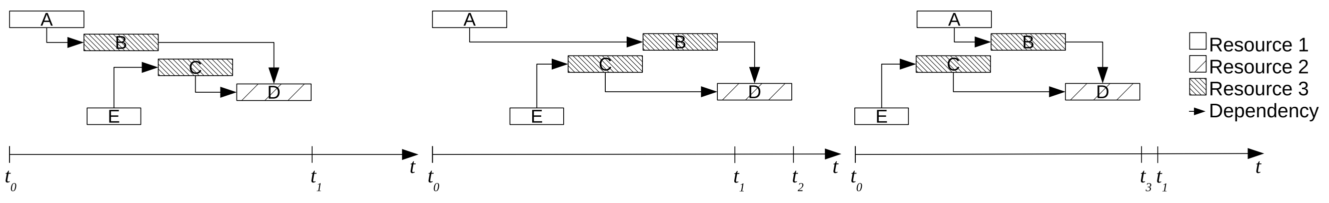

Fig. 2: Left: Initial schedule for illustrative problem; Center: Longer schedule after swapping tasks B and C; Right: Optimal schedule after additionally swapping tasks A and E.

Figure 2 illustrates that strict hill climbing cannot be guaranteed to find the global optimum, but in the illustrated case would not proceed after the first schedule was found. During several case studies it has been observed though, that the total number s of successive non beneficial swaps required to find the optimum is small. In all practical cases that were analyzed the number did not exceed $s = 3$. Though, artificially it is possible to create problems where 4 or more swaps were needed.

3 Tree Search Algorithm

In order to exploit this property a specialized tree search algorithm was developed. The nodes in the search tree represent complete schedules and each edge corresponds to the application of a swap or a delay. As the number of possible swaps and delays is expected to be large for complex problems, the depth of the search tree should be minimized. Therefore the algorithm was designed as an iterative limited depth search where the parameter s determines the search depth at each iteration. The tree is constructed with the current optimum at the root node. Then it is expanded by applying swaps and delays up to a depth of s. The tree is then scanned for a new optimum. If a new optimum was found a new tree is created using the new optimum as the new root node. If no new optimum was found the algorithm terminates and returns the current optimum.

4 Convergence

If a sufficiently large value for s is chosen Lemma 2.1 implies that the algorithm is guaranteed to converge to the global optimum. A lower value for s will speed up the algorithm, but reduce the odds of finding the optimum. The parameter can therefore be viewed as a trade-off setting between speed and quality.

5 Conclusions

This paper presented a novel approach for finding optimal schedules for the resource constrained project scheduling problem that is capable of handling non-linear and non-continuous objective functions while being guaranteed to converge to the global optimum. The algorithm was tested on several real and artificial problems and performed much faster than commonly used procedures. Though in order to have certainty to find the global optimum a sufficiently large value for s should be selected. As there is currently no known way for determining the minimum required value for this parameter, this remains an open issue for further research. Possible approaches for this problem include graph theoretical analysis and pattern matching similar to the work published by K. Shapir [10].

References

[1] D. Halpin and L. Riggs, Planning and analysis of construction operations (John Wiley & Sons, 1992).
[2] J. E. Gomar, C. T. Haas, and D. P. Morton, Journal of Construction Engineering and Management **128**(2), 103–109 (2002).
[3] A. Senouci and N. Eldin, Journal of Construction Engineering and Management **130**(6), 869–877 (2004).
[4] K. Bouleimen and H. Lecocq, European Journal of Operational Research **149**(2), 268–281 (2003).
[5] D. Merkle, M. Middendorf, and H. Schmeck, IEEE Transactions on Evolutionary Computation **6**(4), 333–346 (2002).
[6] H. Guo, K. Zhu, C. Ding, and L. Li, Expert Systems with Applications **37**(2), 1294–1301 (2010).
[7] M. König, Robust construction scheduling using discrete-event simulation, in: Proceedings of the 2011 ASCE International Workshop on Computing in Civil Engineering, (American Society of Civil Engineers, 2011), pp. 446–453.
[8] M. Bügler, G. Dori, and A. Borrmann, Swap Based Process Schedule Optimization using Discrete-Event Simulation, in: Proceedings of the International Conference on Construction Applications of Virtual Reality, London, United Kingdom, (2013).
[9] J. J. Shi, Journal of construction engineering and management **125**(5), 354–360 (1999).
[10] K. Shapir, Feature-basierte Mustererkennung in Bauablaufplänen, in: Forum Bauinformatik 2013, (2013), pp. 311–322.