Proceedings of the Twelfth International
Symposium on Combinatorial Search (SoCS 2019)

Assigning Suppliers to Meet a Deadline

Liat Cohen,[1] **Tal Grinshpoun,**[2] **Roni Stern**[1]
[1]Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel
[2]Ariel University, Ariel 40700, Israel
{liati,sternron}@post.bgu.ac.il, talgr@ariel.ac.il

Abstract

In our setting, we consider projects that consist of completing tasks, where each task needs to be executed by a single supplier chosen from a subset of suppliers. The suppliers differ in their execution times, which are stochastically taken from known distributions. The Supplier Assignment for Meeting a Deadline (SAMD) problem is the problem of assigning a supplier to each task in a manner that maximizes the chance to meet some overall project deadline. We propose an A*-based approach, along with an efficient admissible heuristic function that guarantees an optimal solution for this problem.

Introduction

We introduce the SAMD problem, in which a sequence of tasks $T = \{t_1, \dots t_N\}$ must be completed before a predefined deadline d. The tasks must be completed sequentially, i.e., if $i < j$ then t_i must be completed before starting t_j. Each task t needs to be executed by a single supplier (agent), chosen from a subset of suppliers S_t that have the required proficiency to handle that task. The suppliers differ in their execution times, which are stochastically taken from known distributions. A solution to a SAMD problem is an assignment of suppliers to tasks such that each task t is assigned a supplier that can execute it, and an optimal solution is an assignment that maximizes the probability of meeting the deadline, i.e., of completing all tasks before the deadline.

SAMD is particularly challenging because computing the probability of some assignment of suppliers to meet a deadline is equivalent to an existing NP-hard problem (Cohen, Shimony, and Weiss 2015). Our main contribution is a complete and optimal algorithm that is based on the A* algorithm (Hart, Nilsson, and Raphael 1968). The SAMD problem has some unique properties that require making several non-trivial adjustments to A*, as well as developing a novel efficient domain-specific admissible heuristic.

Problem Definition

Let S be the set of all suppliers. For ease of presentation, we assume that every supplier can perform exactly one task. The task completion time of a supplier $s \in S$ is represented by real-valued random variable X_s taking values in $[0, \infty)$.

A *supplier assignment* is a function $\varphi : T \to S$ such that $\varphi(t) \in S_t$. For a given sequence of tasks T, supplier assignment function φ, and a deadline $d > 0$, we denote by $M(T, \varphi, d)$ the probability that all tasks in T will be completed in their respective order by the suppliers assigned to do them according to φ before time d. Formally,

$$M(T, \varphi, d) = Pr\left(\sum_{t \in T} X_{\varphi(t)} \leq d\right) \quad (1)$$

Corollary 1. *The computation of $M(T, \varphi, d)$ is NP-hard.*

Corollary 1 follows from Cohen et al. (2015, Theorem 5).

Definition 1. *An SAMD problem is defined by the tuple $\langle T, S, X, d\rangle$, where T is an ordered set of tasks, S is a set of suppliers, X is the set of random variables, one for each supplier, that represents the task completion time of this suppliers, and d is the deadline. A solution to an SAMD problem is a supplier assignment φ for T and S. An optimal solution to an SAMD problem is a supplier assignment φ^* that maximizes $M(T, \varphi^*, d)$, that is, for every other solution φ it holds that $M(T, \varphi^*, d) \geq M(T, \varphi, d)$.*[1]

We conjecture that finding an optimal solution to a given SAMD problem is NP-hard because computing $M(T, \varphi, d)$ is NP-hard (Corollary 1). Nevertheless, for small enough number of tasks, it is possible to compute $M(T, \varphi, d)$ exactly. Thus, a brute-force approach to solve an SAMD problem optimally is to compute the value of $M(T, \varphi, d)$ for every possible assignment φ and return the assignment that yields the maximal value. Next, we propose a A*-based algorithm that is guaranteed to find an optimal solution to any SAMD problem and is significantly more efficient than the brute-force approach.

A* for SAMD

First, we define SAMD as a search problem in the following graph. A node in this graph represents a *partial* supplier assignment, which is function that assigns suppliers to a subset of the tasks in T. The initial node n is an empty supplier assignment, that is, a partial supplier assignment that does not assign any supplier to any tasks. Every child node of n represents a possible assignment of a supplier to T_1. Every

[1]This notion of *optimality* we introduced by Frank (1969).

child node of these nodes represents a possible assignment of a supplier to T_2, and so forth.

To apply A* to search in this graph and solve SAMD optimally, however, is not trivial. This is because the notion of paths and costs of paths in this tree is not fully defined in SAMD, since we can only compute the actual value – $M(T, \varphi, d)$ – for a complete supplier assignment. To address this, we define for every node n a single value, denoted $U(n)$, that is an upper bound over the cost of all goal nodes in the sub-tree of the search space rooted by n. Then, in every iteration of our modified A*, we pop from the open list the node with the highest $U(n)$ value and insert its children to the open list. A leaf node represents a complete supplier assignment. So, for every generated leaf node n we compute its corresponding M value, and store the leaf with the highest M value seen so far. We refer to this leaf node as n_{best} and its M value as M_{best}. The search halts when either the open list is empty or the expanded node n is such that $U(n) \leq M_{best}$. When the former condition occurs, we know that all supplier assignments have been checked, in which case n_{best} is indeed optimal. If the latter condition occurs, we know that for every node n in the open list it holds that $U(n) \leq M_{best}$, and thus n_{best} is optimal. Consequently, given that an admissible, i.e., upper bound, $U(\cdot)$ function is implemented, completeness and optimality are guaranteed (Stern et al. 2014; Holte and Zilles 2019).

Corollary 2. A* *for SAMD is complete and optimal.*

Computing an Upper Bound

Our A* can be used with any implementation of $U(\cdot)$ as long as it is admissible for maximization problems, i.e., returns a valid upper bound over all the goals in its subtree. Next, we propose a concrete admissible $U(\cdot)$ that works well in practice.

Let n be the node we wish to compute $U(n)$ for, and assume that n represents a partial assignment φ_n that assigns a supplier to tasks $t_1, \ldots t_i$ for some $i < N$. We compute $U(n)$ by assuming that in every unassigned task $t \in \{t_{i+1}, \ldots, t_N\}$ all suppliers that can perform this task do so simultaneously, so that the task is completed when the fastest one finished. Formally, for the computation of $U(n)$, we assume that the completion time of every unassigned task t to be $\min_{s \in S_t} X_s$. This is a random variable that can be computed easily given all the X_s random variables. Since in our setting only a single supplier performs each task, the real completion time of any single supplier must be larger than this value (or at least equal). Thus, it can be used as an admissible heuristic.

To summarize, we define $U(n)$ as follows:

$$U(n) = \sum_{t \in \{t_1, \ldots t_i\}} X_{\varphi_n(t)} + \sum_{t \in \{t_{i+1}, \ldots t_N\}} \min_{s \in S_t} X_s \quad (2)$$

While the above $U(\cdot)$ is clearly admissible, its computational effort is equivalent to that of the $M(T, \varphi, d)$ value of some complete supplier assignment φ, which is NP-hard (Corollary 1). For small problems it may be feasible to apply such a computation for a limited number of times, as is done in SAMPLING. However, $U(n)$ needs to be computed frequently for every node n that is added to the open list, which renders its exact computation inapplicable. Hence, we turn to an approximate computation of $U(\cdot)$.

The problem with computing the M value stems from the hardness of computation that lies in the exponential growth of the support size (2015). Consequently, after every addend in the computation of $U(\cdot)$ we apply the OPTAPPROX (X, m) operator of Cohen et al. (2018), which for a given random variable X and a requested support size m returns a new random variable X' with a support size of at most m in polynomial time. By restricting the support size to a given m, we bound the exponential growth.

Lemma 3. $U_{\text{OPTAPPROX}}(\cdot)$ *is admissible.*

Discussion and Conclusion

We performed an initial experimental evaluation that shows promising trends for our A*-based approach that on the one hand, finds better solutions than various heuristic methods, and on the other hand, significantly outperforms the mentioned above brute-force approach in terms of run-time. Nevertheless, its exponential search space prevents applying it to large-scale projects. One of the main barriers in this context is the exact computation of M for complete supplier assignments. An interesting direction for future work would be to use OPTAPPROX also there (instead of only in the computation of $U_{\text{OPTAPPROX}}(\cdot)$). This compromises the optimality of the algorithm, but will enable using the (now suboptimal) A*-based approach to much larger problems.

We believe that the presented SAMD problem is a challenging problem that can bridge the gap between the research areas of Artificial Intelligence and Project Management. In future research it will be interesting to see how SAMD can be extended to allow solving projects with parallel tasks, which are common in many real-world projects.

Acknowledgments. This research was supported by the Lynn and William Frankel Centre for Computer Science at Ben-Gurion University.

References

Cohen, L.; Grinshpoun, T.; and Weiss, G. 2018. Optimal approximation of random variables for estimating the probability of meeting a plan deadline. In *AAAI Conference on Artificial Intelligence*.

Cohen, L.; Shimony, S. E.; and Weiss, G. 2015. Estimating the probability of meeting a deadline in hierarchical plans. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Frank, H. 1969. Shortest paths in probabilistic graphs. *Operations Research* 17(4):583–599.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* 4(2):100–107.

Holte, R., and Zilles, S. 2019. On the optimal efficiency of cost-algebraic A*. In *AAAI Conference on Artificial Intelligence*.

Stern, R.; Kiesel, S.; Puzis, R.; Felner, A.; and Ruml, W. 2014. Max is more than min: Solving maximization problems with heuristic search. In *Symposium on Combinatorial Search (SoCS)*.