A New Upper Bound for the Makespan of Cost-Optimal Solutions for Multi-Agent Path Finding (Extended Abstract)

Rodrigo López[1], **Roberto Asín-Achá** [2], **Jorge A. Baier**[1][3]
[1]Department of Computer Science, Pontificia Universidad Católica de Chile, Santiago, Chile
[2]Department of Computer Science, Universidad Tecnica Federico Santa Maria, Santiago, Chile
[3]Instituto Milenio Fundamentos de los Datos, Chile
rilopez3@uc.cl, roberto.asin@usm.cl, jabaier@ing.puc.cl

Introduction

In Multi-Agent Path Finding (MAPF), we are given a graph G and a set of k agents, each of which is associated with an initial vertex and a target vertex; the task is to find, for each agent, a path connecting its initial and target vertices. In addition, the set of k paths must be non-conflicting. Finding a cost-optimal solution to a MAPF instance, a solution that minimizes the total number of edges over the k paths, is an NP-hard task (Surynek 2010). When MAPF tasks are relatively small and dense, the approaches that perform best in practice are compilation-based. One successful strategy, applicable to SAT-, ASP-, and MIP-based solvers (Barták and Svancara 2019; Gómez, Hernández, and Baier 2021; Asín Achá et al. 2022), involves two phases: first, finding a solution Π^{min} of minimum makespan (i.e., minimum time-to-completion), and second, using a theoretical result to compute an upper bound T^* for the makespan of a cost-optimal solution. By encoding the problem for makespan T^*, a minimum-cost solution is guaranteed to be cost-optimal. However, the state-of-the-art theoretical bound used (Surynek et al. 2016; Barták and Svancara 2019; Gómez, Hernández, and Baier 2021) may significantly overestimate the actual makespan.

Background

A MAPF instance is a tuple $P = (G, A, s, t)$ where $G = (V, E)$ is a directed graph, A is a set of agents, $s : A \to V$ is such that $s(a)$ is the start vertex a, and $t : A \to V$ is such that $t(a)$ is the target vertex of a. We assume that for every vertex $v \in V$, there is an edge $(v, v) \in E$, and thus agents are allowed to "wait" at each node.

A path $\pi = u_0, u_1, \ldots, u_n$ in graph $G = (V, E)$ is a non-empty sequence of vertices in V such that for every $i \in \{1, \ldots, n\}$, $(u_{i-1}, u_i) \in E$. A sequence of vertices π is a path from u to v if π is a path whose first element is u and whose last element is v. Given a path $\pi = u_0, u_1, \ldots, u_n$, $\pi[i]$, with $i \geq 0$, denotes the i-th vertex of π; that is, $\pi[i] = u_i$. The cost of $\pi = u_0 u_1 \ldots u_n$, denoted by $C(\pi)$, is the number of edges traversed; therefore, $C(\pi) = n$.

Definition 1. *A solution to MAPF instance $P = (G, A, s, t)$ is a function Π whose domain is A such that:*

1. *For every $a \in A$, $\Pi(a)$ is a path in G from $s(a)$ to $t(a)$.*
2. *For every $a \in A$, $\Pi(a)$ is not of the form πuu, for some $u \in V$ and some sequence π of vertices in V. This means that agents do not perform a* wait *as their last action.*
3. *Π is* conflict-free*; specifically, two paths in Π do not have a vertex or an edge conflict.*

Definition 2. *A relaxed solution for a MAPF instance $P = (G, A, s, t)$ is a function Π whose domain is A that (only) satisfies Conditions 1 and 2 of Definition 1.*

The *makespan* of Π is $T(\Pi) = \max_{a \in A} C(\Pi(a))$, and the *cost* of Π is $C(\Pi) = \sum_{a \in A} C(\Pi(a))$. Solution Π is *makespan-optimal* iff Π is a solution of minimum makespan. Solution Π is *cost-optimal* iff Π is a solution of minimum cost. Makespan-optimal and cost-optimal relaxed solutions are defined analogously. Current approaches (e.g., Gómez, Hernández, and Baier 2021; Asín Achá et al. 2022), use the following theoretical result to compute T^*.

Theorem 1. *Let P be a MAPF instance. Let Π^{min} be a makespan-optimal solution, and let Π^{rel} be a cost-optimal relaxed solution to P. Then there exists a cost-optimal solution Π^* to P such that:*

$$T(\Pi^*) \leq T(\Pi^{rel}) + C(\Pi^{min}) - C(\Pi^{rel}). \qquad (1)$$

A New Upper Bound T^*

The bound we propose is based on the solution to sub-instances of the given MAPF instance to obtain a better upper bound for the makespan of a cost-optimal solution.

Definition 3. *Given a MAPF instance $P = (G, A, s, t)$, and a subset b of agents in A ($b \subseteq A$), the sub-instance of P relative to b, denoted as $P|_b$, is the tuple $P|_b = (G, b, s|_b, t|_b)$.*

Now, we present the main theoretical result of the paper, from which we derive our new upper bound.

Theorem 2. *Let $P = (G, A, s, t)$ be a MAPF instance and let Π^* be an optimal solution to P. Let a_{max} be an agent of maximum cost in Π^*; that is, such that $C(\Pi^*(a_{max})) = T(\Pi^*)$. Let B be a partition of A containing $\{a_{max}\}$, and let Π^*_b be an optimal solution to $P|_b$, for every $b \in B$. Let Π^{rel} be a cost-optimal relaxed solution to P. Finally, let Π^{min} be a makespan-optimal solution for P. Then,*

$$T(\Pi^*) \leq T(\Pi^{rel}(a_{max})) + C(\Pi^{min}) - \sum_{b \in B} C(\Pi^*_b). \qquad (2)$$

A key point to note is that partition B in Theorem 2 must contain $\{a_{max}\}$, where a_{max} is an agent with maximum cost within the cost-optimal solution, which is *still not computed*. This means that to apply our bound, we need to *guess* a_{max}. A reliable heuristic we've used is to select the agent with the highest cost in the relaxed solution. Once we compute the solution, we check if our guess is correct. If it was, the bound is correct; hence, the solution is cost-optimal (we omit the proof to this claim in this abstract for lack of space). However, if our guess is wrong, we cannot ensure the bound's accuracy and thus cannot guarantee optimality. We may use the bound of Theorem 1 to compute a solution in such cases.

Parallelization with ReBo

To guarantee that we always find a solution and to minimize execution time, our recursive bounding approach – which we call ReBo – exploits parallel computation. In the first step, we compute a relaxed solution Π^{rel}, and define a_i as the i-th agent with the highest cost in Π^{rel}. Now we define $A_0 = A$, and $A_i = A_{i-1} \setminus \{a_i\}$ for every $i \in \{1, \ldots, k\}$. We run up to m threads ($m \leq k$). Thread 0 runs the standard approach (using Theorem 1 for the upper bound). The remaining threads exploit different ways in which our bound can be used. Thread i finds a solution to $P|_{A_i}$ using the traditional approach and then uses the returned solution to sequentially compute solutions to $P|_{A_i}, \ldots, P|_{A_1}, P|_{A_0}$ using our bound. We report the solution once one thread finds a solution to P.

Experimental Evaluation

ReBo is applicable to SAT and ASP compilation-based approaches. To our knowledge, there are no other parallel cost-optimal MAPF solvers. Therefore, we compare our ReBo implementation against ASP-MAPF2 (Asín Achá et al. 2022) due to its superiority over other approaches (i.e., MDD-SAT (Surynek et al. 2016), BCP (Lam et al. 2022)) on the chosen benchmarks. To compare with ASP-MAPF2 fair, ReBo was restricted to 4 threads, assigning one core to each thread, and for ASP-MAPF2, we configured the ASP solver to use 4 cores. We used the benchmarks proposed by Gómez, Hernández, and Baier (2021) and Asín Achá et al. (2022). That includes 410 grid instances of size 20×20, 1280 grid instances of size 50×50, and 1200 warehouse-like maps instances of size 18×33 and 30×57; 2890 instances in total.

Table 1 shows the total number of instances solved for each method at a time limit. We observe that ReBo is competitive and outperforms ASP-MAPF2. Figure 1 compares the values of the bounds of Theorem 1 and Theorem 2 against the makespan of a cost-optimal solution on the instances of the benchmark sets that are solved for both methods and the new bound criterion is met, grouped by density. The values for instances with the same density are averaged. On average, the overestimation of our approach is 7.74%, while the overestimation of the previous bound is 54.93%. We evaluated the accuracy of choosing agent a_{max} and found such a guess was correct in 91.97% of the cases.

Time limit	ReBo	ASP2
1 second	49	**68**
10 seconds	**559**	506
100 seconds	**1415**	1329
600 seconds	**1926**	1838
1200 seconds	**2105**	2006

Table 1: Number of solved instances by ReBo and ASP-MAPF2 (ASP2), to a given time limit.

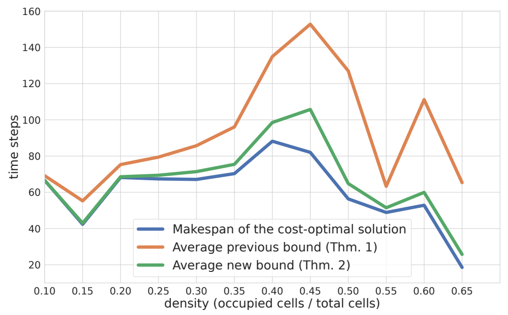

Figure 1: Average bound proposed by density (occupied cells / total cells) for the instances solved for both methods.

Summary and Future Work

We propose a new theoretical bound for the makespan of a cost-optimal solution to a MAPF instance. The bound we proposed is correct and at least as tight as the bound that has been used until now. We also proposed a way to exploit this new bound in practice which exploits parallelization.

Acknowledgments

This work was funded by the ANID Doctorado Nacional 2020 under Grant 21201864 and National Center for Artificial Intelligence CENIA FB210017, Basal ANID.

References

Asín Achá, R. J.; López, R.; Hagedorn, S.; and Baier, J. A. 2022. Multi-Agent Path Finding: A New Boolean Encoding. *JAIR*, 75: 323–350.

Barták, R.; and Svancara, J. 2019. On SAT-Based Approaches for Multi-Agent Path Finding with the Sum-of-Costs Objective. In *SoCS*, 10–17. AAAI Press.

Gómez, R. N.; Hernández, C.; and Baier, J. A. 2021. A Compact Answer Set Programming Encoding of Multi-Agent Pathfinding. *IEEE Access*, 9: 26886–26901.

Lam, E.; Bodic, P. L.; Harabor, D.; and Stuckey, P. J. 2022. Branch-and-cut-and-price for multi-agent path finding. *Computers & Operations Research*, 144: 105809.

Surynek, P. 2010. An Optimization Variant of Multi-Robot Path Planning Is Intractable. In *AAAI*, 1261–1263.

Surynek, P.; Felner, A.; Stern, R.; and Boyarski, E. 2016. Efficient SAT Approach to Multi-Agent Path Finding Under the Sum of Costs Objective. In *ECAI*, 810–818. IOS Press.