Multi-Agent Path Finding with Deadlines: Preliminary Results*

Extended Abstract

Hang Ma*  
USC  
hangma@usc.edu

Glenn Wagner  
CSIRO

Ariel Felner  
Ben-Gurion University

Jiaoyang Li*  
T. K. Satish Kumar*  
Sven Koenig*

ABSTRACT

We formalize the problem of multi-agent path finding with deadlines (MAPF-DL). The objective is to maximize the number of agents that can reach their given goal vertices from their given start vertices within a given deadline, without colliding with each other. We first show that the MAPF-DL problem is NP-hard to solve optimally. We then present an optimal MAPF-DL algorithm based on a reduction of the MAPF-DL problem to a flow problem and a subsequent compact integer linear programming formulation of the resulting reduced abstracted multi-commodity flow network.

ACM Reference Format:

1 INTRODUCTION

Multi-agent path finding (MAPF) is the problem of planning collision-free paths for multiple agents in known environments from their given start vertices to their given goal vertices. MAPF is important, for example, for aircraft-towing vehicles [20], warehouse and office robots [30, 34], and game characters [19]. The objective is to minimize the sum of the arrival times of the agents or the makespan. The MAPF problem is NP-hard to solve optimally [36] and even to approximate within a small constant factor for makespan minimization [18]. It can be solved with reductions to other well-studied combinatorial problems [5, 21, 28, 35] and dedicated optimal [2, 6, 8, 22, 23, 25, 26, 32], bounded-suboptimal [1, 3], and suboptimal MAPF algorithms [4, 12, 24, 27, 31, 33], as described in several surveys [7, 15].

The MAPF problem has recently been generalized in different directions [9, 10, 13–17] but none of them capture an important characteristic of many applications, namely the ability to meet deadlines. We thus formalize the multi-agent path finding problem with deadlines (MAPF-DL problem). The objective is to maximize the number of agents that can reach their given goal vertices from their given start vertices within a given deadline, without colliding with each other. In previously studied MAPF problems, all agents have to be routed from their start vertices to their goal vertices, and the objective is with regard to resources such as fuel (sum of arrival times) or time (makespan). In the MAPF-DL problem, on the other hand, the resources are the agents themselves. We first show that the MAPF-DL problem is NP-hard to solve optimally. We then present an optimal MAPF-DL algorithm based on a reduction of the MAPF-DL problem to a flow problem and a subsequent compact integer linear programming formulation of the resulting reduced abstracted multi-commodity flow network.

2 MAPF-DL PROBLEM

We formalize the MAPF-DL problem as follows: We are given a deadline, denoted by a time step $T_{end}$, an undirected graph $G = (V, E)$, and $M$ agents $a_1, a_2 \ldots a_M$. Each agent $a_i$ has a start vertex $s_i$ and a goal vertex $g_i$. In each time step, each agent either moves to an adjacent vertex or stays at the same vertex. Let $l_i(t)$ be the vertex occupied by agent $a_i$ at time step $t ∈ \{0 \ldots T_{end}\}$. Call an agent $a_i$ successful iff it occupies its goal vertex at the deadline $T_{end}$, that is, $l_i(T_{end}) = g_i$. A plan consists of a path $l_i$ assigned to each successful agent $a_i$. Unsuccessful agents are removed at time step zero and thus have no paths assigned to them.1 A solution is a plan that satisfies the following conditions: (1) For all successful agents $a_1, l_i(0) = s_i$ [each successful agent starts at its start vertex]. (2) For all successful agents $a_i$ and all time steps $t \geq 0$, $(l_i(t) − 1, l_i(t)) \in E$ or $l_i(t − 1) = l_i(t)$ [each successful agent always either moves to an adjacent vertex or does not move]. (3) For all pairs of different successful agents $a_i$ and $a_j$ and all time steps $t$, $l_i(t) \neq l_j(t)$ [two successful agents never occupy the same vertex simultaneously]. (4) For all pairs of different successful agents $a_i$ and $a_j$ and all time steps $t > 0$, $l_i(t − 1) \neq l_j(t)$ or $l_i(t − 1) \neq l_j(t)$ [two successful agents never traverse the same edge simultaneously in opposite directions]. Define a collision between two different successful agents $a_i$ and $a_j$ to be either a vertex collision $(a_i, a_j, v, t)$ iff $v = l_i(t) = l_j(t)$ (corresponding to Condition 3) or an edge collision $(a_i, a_j, u, v, t)$ iff $u = l_i(t) = l_j(t + 1)$ and $v = l_i(t) = l_j(t + 1)$ (corresponding to Condition 4). The objective is to maximize the number of successful agents $M_{succ} = |\{a_i|l_i(T_{end}) = g_i\}|$.

Theorem 1. It is NP-hard to compute a MAPF-DL solution with the maximum number of successful agents.

The proof of the theorem reduces the ≤3-SAT problem [29], an NP-complete version of the Boolean satisfiability problem, to the MAPF-DL problem. The reduction is similar to the one used for proving the NP-hardness of approximating the optimal makespan for the MAPF problem [18]. It constructs a MAPF-DL instance with

---

1 Depending on the application, the unsuccessful agents can be removed at time step zero, wait at their start vertices, or move out of the way of the successful agents. We choose the first option in this paper. If the unsuccessful agents are not removed, they can obstruct other agents. However, our proof of NP-hardness does not depend on this assumption, and our MAPF-DL algorithm can be adapted to other assumptions.
There is a one-to-one correspondence between all solutions of a MAPF-DL instance with the maximum number of successful agents and all maximum integral flows on the corresponding flow network.

The proof of the theorem is similar to the one for the reduction of the MAPF problem to the multi-commodity flow problem [35]. Figure 1(a) shows a MAPF-DL instance with deadline $T_{end} = 2$. Agents $a_1$ and $a_2$ have start vertices $s_1$ and $s_2$ and goal vertices $g_1$ and $g_2$, respectively. The number of successful agents is at most $M_{succ} = 1$ because only agent $a_2$ can reach its goal vertex in two time steps. Figure 1(c) shows the corresponding flow network with a maximum flow (in color) that corresponds to a solution with unsuccessful agent $a_1$ and successful agent $a_2$ with path $(v_2, v_4, v_5)$.

**Abstracted Flow Network and Compact ILP Formulation**

We construct a compact ILP formulation based on an abstraction of the flow network $N = (V, E)$ and additional linear constraints to prevent vertex and edge collisions. We obtain the abstracted flow network $N' = (V', E')$ by (1) contracting each $(v_i^{in}, v_i^{out}) \in E$ and replacing $v_i^{in}$ and $v_i^{out}$ with a single vertex $v_i$ for all $v \in V$ and $t = 1 \ldots T_{end}$ (and $v_0^{out}$ with $v_0$); and (2) replacing the gadget for each $(u, v) \in E$ and each $t = 0 \ldots T_{end} - 1$ with a pair of edges $(u_t, v_{t+1})$ and $(v_t, u_{t+1})$. Figure 2 (left) shows an example. Then, we use the standard ILP formulation of this abstracted network augmented with the constraints shown in red:

\[
\text{maximize } M_{succ} = \sum_{i=1}^{M} \sum_{e \in E} x_{i,e} \text{ subject to }
\]

\[
0 \leq \sum_{e \in E} x_{i,e} \leq \delta(e) \quad (\text{subsumed by the top red constraints})
\]

\[
\sum_{e \in \delta^{-1}(v)} x_{i,e} - \sum_{e \in \delta^{-1}(v)} x_{i,e} = 0 \quad i = 1 \ldots M, v \in V' \backslash \{(s_1, g_1)_{T_{end}}\}
\]

\[
\sum_{e \in \delta^{-1}(v_0^{out})} x_{i,e} = \sum_{e \in \delta^{-1}(v_T)} x_{i,e} \quad i = 1 \ldots M
\]

\[
\sum_{i=1}^{M} \sum_{e \in \delta^{-1}(w)} x_{i,e} \leq 1 \quad w \in V'
\]

\[
\sum_{i=1}^{M} \sum_{e \in \delta^{-1}(g_2)} x_{i,e} = 0 \quad (u_t, v_{t+1}) \in E', (v_t, u_{t+1}) \in E',
\]

where the 0/1 variable $x_{i,e}$ represents the amount of flow of commodity type $i$ on edge $e \in E'$ and the sets $\delta^{-1}(v)$ and $\delta^{-1}(v)$ contain all outgoing and incoming, respectively, edges of vertex $v$. The top red constraints prevent vertex collisions of the form $(v, *, u, t)$, and the bottom red constraints prevent edge collisions of the forms $(v, *, u, v, t)$ and $(u, *, v, u, t)$.

**Reduced Abstracted Flow Network**

We can remove all vertices and edges from the abstracted flow network that are not on some path from at least one start vertex to the corresponding goal vertex in the abstracted flow network. This can be done by performing one complete forward breadth-first search from each start vertex and one complete backward breadth-first search from each goal vertex and then keeping only those vertices and edges that are part of the search trees associated with at least one start vertex and the corresponding goal vertex. Figure 2 (right) shows an example. We then use the compact ILP formulation of the resulting reduced abstracted flow network.

**Experimental Evaluation**

We tested our optimal MAPF-DL algorithm on a 2.50 GHz Intel Core i5-2450M laptop with 6 GB RAM, using CPLEX V12.7.1 [11] as the ILP solver. We randomly generated MAPF-DL instances with different numbers of agents (ranging from 10 to 100 in increments of 10) on 40 x 40 4-neighbor 2D grids with deadline $T_{end} = 50$. We blocked all grid cells independently at random with 20% probability each. We generated 50 MAPF-DL instances for each number of agents. We placed the start and goal vertices of each agent randomly at distance 48, 49, or 50. The following table shows the percentage of instances that could be solved within a runtime limit of 60 seconds per instance.

<table>
<thead>
<tr>
<th>agents</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>success rate</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>98%</td>
<td>88%</td>
<td>76%</td>
<td>12%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

**References**


