Conflict-Free Multi-Agent Meeting

Dor Atzmon, Shahar Idan Freiman, Oscar Epshtein, Oran Shichman, Ariel Felner
Ben-Gurion University of the Negev
{dorat, freimans, oscarep, shichman}@post.bgu.ac.il, felner@bgu.ac.il

Abstract

Multi-Agent Meeting (MAM) is the problem of finding a meeting location for multiple agents and paths to that location. Recently, a *Multi-Directional Heuristic Search* algorithm, called MM*, was introduced. MM* is a state-of-the-art MAM optimal solver that searches from multiple directions (one for each agent) and is guided by a heuristic function. Practically, a solution to MAM may contain conflicting paths. A related problem that plans conflict-free paths to a given set of goal locations is the *Multi-Agent Path Finding* problem (MAPF). In this paper, we solve the *Conflict-Free Multi-Agent Meeting* problem (CF-MAM). In CF-MAM, we find a meeting location for multiple agents (as in MAM) as well as conflict-free paths (as in MAPF) to that location. We introduce two novel algorithms, which combine MAM and MAPF solvers, for optimally solving CF-MAM. We compare both algorithms experimentally, showing the pros and cons of each algorithm.

Conflict-Free Multi-Agent Meeting

While MAM finds a meeting location and paths for multiple agents to that meeting location, it ignores conflicts between the agents. Here, we define the problem of finding a meeting location and conflict-free paths to the meeting location.

The *Conflict-Free Multi-Agent Meeting* problem (CF-MAM) gets as input the tuple $\langle \mathcal{G}, S \rangle$, where $\mathcal{G}$ is an undirected connected graph and S is a set of start locations. We assume that all edges have a unit cost, which is common in other conflict-free MAPF problems (Stern et al. 2019). A *solution* to CF-MAM is a meeting location m and a set of conflict-free paths Π to the meeting location m. Let $N(v)$ represent the neighbors of v, i.e., $\forall v' \in N(v), (v, v') \in \mathcal{E}$. Each path $\pi_i \in \Pi$ for agent $a_i \in A$ consists of a sequence of locations, such that each two consecutive locations $v_1, v_2 \in \pi_i$ must either satisfy $v_2 \in N(v_1)$ (move action) or $v_1 = v_2 \in \mathcal{V}$ (wait action).

The cost of path π_i is denoted by $C(\pi_i)$ and equals the number of edges traversed in π_i ($C(\pi_i) = |\pi_i| - 1$). We use $\pi_i(t)$ to denote the t-th location in π_i. Thus, $\pi_i(0) = s_i$ and $\pi_i(|\pi_i|) = m$. Often, a set of paths that minimizes some objective function is preferred. We focus on minimizing the *Sum-Of-Costs* (SOC) objective function, which equals the sum of the costs of all paths in Π ($C(\Pi) = \sum_{\pi_i \in \Pi} C(\pi_i)$). We use C^* to denote the cost of the optimal solution.

In this research, we focus on two main types of conflicts (Stern et al. 2019): *vertex conflict* and *swapping conflict*. A vertex conflict $\langle a_i, a_j, v, t \rangle$ occurs between two paths π_i and π_j if the same vertex $v \in \mathcal{V}$ is occupied by both agents a_i and a_j at the same timestep t, i.e., $\pi_i(t) = \pi_j(t) = v$. A swapping conflict $\langle a_i, a_j, e, t \rangle$ occurs between two paths π_i and π_j if the same edge $e \in \mathcal{E}$ is traversed in opposite directions by both agents a_i and a_j between the same two consecutive timesteps t and $t+1$, i.e., $(\pi_i(t), \pi_i(t+1)) = (\pi_j(t+1), \pi_j(t)) = e$.

While agents must avoid conflicts on their path, naturally, we define that agents can arrive at m at the same timestep. This is practical, for example, in the case that agents disappear at goal (Stern et al. 2019) (e.g., robots entering a charging station or autonomous vehicles entering a garage).

In the full paper of this research, we prove that given a set of paths that only contains swapping conflicts, a set of conflict-free paths (without vertex conflicts and without swapping conflicts) with the same cost can be constructed. In the next two sections, we introduce two algorithms for CF-MAM. As a (conflict-free) solution can be constructed from a set of paths that only contains swapping conflicts, both algorithms only consider vertex conflicts.

CBS-Based Solution for CF-MAM

Conflict-Based Search (CBS) (Sharon et al. 2015) is a prominent, state-of-the-art MAPF solver. It plans a set of paths that may contain conflicts and iteratively resolve them by imposing constraints on the agents and replanning new paths for the constrained agents. Here, we introduce the CFM-CBS algorithm for solving CF-MAM, which uses the framework of the CBS algorithm. CFM-CBS has two levels. The high level of CFM-CBS searches the binary *constraint tree* (CT). Each node $N \in CT$ contains: (1) a set of constraints imposed on the agents ($N.constraints$); (2) a set of paths ($N.\Pi$) that satisfies all constraints in $N.constraints$; and (3) the cost of $N.\Pi$ ($N.cost$). A *constraint* is a tuple $\langle a_i, v, t \rangle$ that prohibits agent a_i to occupy location v at timestep t. We use such constraints for resolving conflicts, as explained below. The root node of CT contains an empty set of constraints. The high level searches the CT in a best-first manner, prioritizing nodes with lower cost.

Generating a CT node. Given a node N, the low level of CFM-CBS solves the given CF-MAM problem instance as a MAM problem that satisfies all constraints of node N. Such a solution can be achieved using any MAM solver, such as MM* (Atzmon et al. 2020). However, to support constraints, as well as wait actions, MM* needs to be slightly modified to CF-MM*. Thus, instead of the pair (a_i, v), a node in CF-MM* is a tuple of (a_i, v, t) representing an agent and its location at timestep t. In CF-MM* an *invalid node* (a_i, v, t) is a node that violates the constraint $\langle a_i, v, t\rangle$. CF-MM* may generate invalid nodes as such nodes may be meeting locations. However, if an invalid node N is chosen for expansion, CF-MM* discards N and does not expand it.

Expanding a CT node. Once CFM-CBS has chosen a node N for expansion, it checks its paths $N.\Pi$ for conflicts. If it is conflict-free, then node N is a goal node and CFM-CBS returns its solution. Otherwise, CFM-CBS *splits* node N on one of the conflicts $\langle a_i, a_j, v, t\rangle$ by generating two children for node N. Each child node has a set of constraints that is the union of $N.constraints$ and a new constraint. One of the two children adds the new constraint $\langle a_i, v, t\rangle$ and the other child adds the new constraint $\langle a_j, v, t\rangle$.

Iterative Meeting Search for CF-MAM

The *Iterative Meeting Search* algorithm (IMS) for CF-MAM also has two levels. The high level of IMS iteratively examines possible meeting locations until the optimal meeting location can be determined. This is done by a best-first search on possible meeting locations. The low level (not fully described here) sets each possible meeting location as a meeting location and applies a reduction to Network Flow.

First, IMS initializes OPEN, CLOSED, and an upper bound on the cost of the optimal solution U ($U \geq C^*$) with infinity. The high level performs a best-first search, starting from only one of the start locations s_i ((a_i, s_i) is inserted to OPEN). While OPEN is not empty, an expansion cycle is performed. Each expansion cycle starts by extracting the node (a_i, v) with the lowest f-value (the same f-value as in MM*) from OPEN. As MAM is a relaxed problem of CF-MAM, for the same input $\langle \mathcal{G}, S\rangle$, the cost C' of the optimal solution for MAM is a lower bound on the cost C^* of the optimal solution for CF-MAM, i.e., $C' \leq C^*$. Thus, since f is a lower bound on C', it is also a lower bound on C^*. For each node (a_i, v) selected for expansion, the high level calls the low level to calculate the cost of meeting at location v. Then, U is updated with the lowest cost found. As U is an upper bound on the cost of the optimal solution C^*, if $fmin \geq U$ then IMS halts and the optimal solution is found ($C^* = U$), where $fmin$ is the lowest f in OPEN. Otherwise, in case the optimal solution is still not found, for each neighbor v' of v, the high level inserts (a_i, v') to OPEN and moves (a_i, v) to CLOSED. The node (a_i, v') is not inserted to OPEN in case it is either in CLOSED or in OPEN with a lower or equal cost.

Experiments

For CFM-CBS, we used CF-MM* as a low-level solver. For IMS, we used for solving the Minimum-Cost Flow problem (MCFP) an efficient implementation of the cost–scaling algorithm of Goldberg and Tarjan (1990; 1997), which runs in polynomial time. For both, we used the clique heuristic as an admissible heuristic for a meeting location and Manhattan distance as admissible heuristic between two locations.

#Agents	Solver	10×10		50×50		
		Cost	Time	Succ.	Cost	Time
3	CFM-CBS	11	**0.0**	**50**	59	**0.0**
	IMS		**0.0**	**50**		0.3
5	CFM-CBS	23	**0.0**	**50**	106	**0.5**
	IMS		**0.0**	**50**		8.3
7	CFM-CBS	34	**0.1**	**50**	155	**3.9**
	IMS		**0.1**	49		40.9
9	CFM-CBS	45	**0.3**	**49**	204	**26.3**
	IMS		0.4	46		126.4
11	CFM-CBS	56	9.5	**39**	245	**50.5**
	IMS		**0.8**	23		200.2
13	CFM-CBS	67	30.2	**29**	-	-
	IMS		**1.2**	3		
15	CFM-CBS	79	57.3	**21**	-	-
	IMS		**1.7**	1		

Table 1: Results for 10x10 and 50x50 grids with 20% Obs.

We compared CFM-CBS and IMS on 10x10 and 50x50 grids with 20% randomly placed obstacles, and $3, 5, 7, 9, 11, 13$, and 15 randomly allocated agents. We created 50 problem instances for each combination and measured the success rate (for timeout of 5min for each instance), average cost, and average time (seconds). Table 1 presents the results for this experiment. Each row shows the number of agents. The results for the 10x10 grids and for the 50x50 grids are in columns 3-4 and 5-7, respectively.

For the 10x10 grids, both CFM-CBS and IMS solve all problem instances and hence we do not present the success rate in the table. As expected, larger number of agents increases the average cost and the average time for both solvers. However, the influence of this increase is greater for CFM-CBS than for IMS. For example, for 7 agents, both solvers ran for $\sim 0.1s$, and for 15 agents, CFM-CBS ran $\sim 57.3s$ while IMS ran only $\sim 1.7s$. The runtime of CBS-based solutions is exponential in the number of conflicts it resolves. Thus, CFM-CBS does not perform well in dense environments, such as small grids with many agents.

For the 50x50 grids, as the number of agents increased, both solvers solved fewer instances within the 5min timeout. However, CFM-CBS solved more instances than IMS. For 13 agents, CFM-CBS solved 29 problem instances while IMS only solved 3. The same trend can be seen for the time: CFM-CBS was faster than IMS. For 11 agents, CFM-CBS and IMS ran $\sim 50.5s$ and $\sim 200.2s$, respectively. The average cost and average time in the table were calculated from instances that were solved by both solvers. Here, the environment is sparser and fewer conflicts occur. Thus, CFM-CBS can perform better than observed above.

Our experiments provide the following general rule. CFM-CBS should be used in sparse environments while IMS should be used in dense environments.

References

Atzmon, D.; Li, J.; Felner, A.; Nachmani, E.; Shperberg, S. S.; Sturtevant, N.; and Koenig, S. 2020. Multi-Directional Heuristic Search. In *IJCAI*, 4062–4068.

Goldberg, A. V. 1997. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of algorithms* 22(1): 1–29.

Goldberg, A. V.; and Tarjan, R. E. 1990. Finding minimum-cost circulations by successive approximation. *Mathematics of Operations Research* 15(3): 430–466.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-based search for optimal multi-agent pathfinding. *AIJ* 219: 40–66.

Stern, R.; Sturtevant, N. R.; Felner, A.; Koenig, S.; Ma, H.; Walker, T. T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T. K. S.; Barták, R.; and Boyarski, E. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *SoCS*, 151–159.