The Thirteenth International Symposium
on Combinatorial Search (SoCS 2020)

Generalizing Multi-Agent Path Finding for Heterogeneous Agents

Dor Atzmon, Yonathan Zax, Einat Kivity,
Lidor Avitan, Jonathan Morag, Ariel Felner
Ben-Gurion University of the Negev
{dorat, sachsy, einatkiv, lidorav, moragj}@post.bgu.ac.il, felner@bgu.ac.il

Abstract

Multi-Agent Path Finding (MAPF) is the problem of finding non-colliding paths for multiple agents. The classical problem assumes that all agents are homogeneous, with a fixed size and behavior. However, in reality agents are heterogeneous. In this paper, we generalize MAPF to the case of general, heterogeneous agents (G-MAPF). We then show how two previous settings of large agents and k-robust agents are special cases of G-MAPF. Finally, we introduce G-CBS, a variant of the Conflict-Based Search (CBS) algorithm for G-MAPF, which does not cause significant extra overhead.

1 Introduction

In the *Multi-Agent Path Finding* problem (MAPF) a plan is needed for leading a set of agents from their start locations to their goal locations, without collisions. MAPF is practically applicable in real-world problems, such as traffic control, robotics, and video games (Felner et al. 2017). Finding plans that minimize some objective function (optimal solutions) is NP-hard (Surynek 2010; Yu and LaValle 2013). However, some algorithms manage to do so for dozens of agents (Surynek 2012; Felner et al. 2018; Lam et al. 2019; Gange, Harabor, and Stuckey 2019; Li et al. 2019a).

In classical MAPF the assumption is that each agent occupies only one location at each time step and that each action applied is always successful. In reality, the environment may be more complicated. In particular, agents can have different sizes (and occupy more than one location) and may behave unexpectedly. Two important attempts did not make the one-agent-per-one-location assumption. Li et al. (2019b) extended MAPF to *MAPF for large agents* (LA-MAPF). These large agents occupy multiple locations at each time step, and collide when two agents overlap, i.e., jointly occupy at least one location at the same time. However, LA-MAPF assumes that these agents have a fixed unchanged size and that they cannot rotate. Atzmon et al. (2020b) studied the problem of k-*Robust MAPF* (kR-MAPF), for agents that may experience up to k unexpected delays. As agents may be delayed, an agent may located differently than originally planed. However, in kR-MAPF the assumption is that the size of each agent is only a single location.

The two problems above were designed for specific types of agents. In this paper, we suggest a more flexible and general definition for agents of various types. We define the G-MAPF problem, which generalizes MAPF for heterogeneous agents. In G-MAPF, different agents may have different sizes or shapes, and different behaviors with regards to movements and delays. We extend *Conflict-Based Search* (CBS), the well known MAPF solver, to G-CBS for G-MAPF. We will show how G-CBS works for both LA-MAPF and kR-MAPF. We chose these two settings as they are two extreme cases of occupying locations; in LA-MAPF, multiple locations are occupied each time, and in kR-MAPF each location is occupied for multiple times (other related works are described in Section 6). While G-CBS can work with heterogeneous agents, our experiments demonstrate that it performs relatively similar to the algorithms designed for specific types of agents.

2 Definitions and Background

In the *Multi-Agent Path Finding* problem (MAPF) the input is an undirected graph $G = (V, E)$ and a set of n agents labeled $A = \{a_1, \dots, a_n\}$, where each agent $a_i \in A$ has a start location s_i and a goal location g_i ($s_i, g_i \in V$). A *solution* to the problem is a plan π consists of paths $\pi = \{\pi_1, \dots, \pi_n\}$, such that for each agent a_i the path π_i contains a sequence of adjacent locations that leads from s_i to g_i. Let $\pi_i(t)$ be the t-th location in that sequence. Between each two consecutive locations in π_i, the agent can either *move* to an adjacent location or *wait* in its current location ($\forall t : (\pi_i(t), \pi_i(t+1)) \in E \vee \pi_i(t) = \pi_i(t+1)$).

A solution π is called *valid* if it is conflict-free. Following the terminology suggested by Stern et al. (2019), we define the following conflicts. A *vertex conflict* $\langle a_i, a_j, v, t\rangle$ occurs when two agents a_i and a_j are planned to be in the same location v at the same time step t, i.e., $\pi_i(t) = \pi_j(t) = v$. A *swapping conflict* $\langle a_i, a_j, e, t\rangle$ occurs when two agents a_i and a_j are planned to traverse the same edge e in opposite directions between the same consecutive time steps, i.e., $\pi_i(t) = \pi_j(t+1) \wedge \pi_i(t+1) = \pi_j(t)$ ($e = (\pi_i(t), \pi_j(t))$).

A solution π is called *optimal* if it minimizes some objective function. Let $C(\pi_i)$ be the cost of all move/wait actions in π_i. We focus on minimizing *Sum-Of-Costs* (SOC), which is the sum of all paths costs in π, i.e., $\sum_{\pi_i \in \pi} C(\pi_i)$.

2.1 Conflict-Based Search (CBS)

CBS (Sharon et al. 2015) is an efficient, popular algorithm for finding optimal MAPF solutions. CBS searches in two levels. The high-level of CBS builds a binary tree, called *Constraint Tree* (CT). Each CT node N contains: (1) *N.constraints*, a set of constraints; (2) $N.\pi$, a solution; (3) *N.cost*, the cost of $N.\pi$ based on the selected objective function. A constraint $\langle a, v, t\rangle$ forbids agent a from being in location v at time step t (the same can be done for an edge). The low-level of CBS plans a path for one agent from its start location to its goal location that satisfies a given set of constraints. Any single-agent path finding algorithm, such as A*, can be used as the low-level algorithm. Each CT node calls the low-level for each agent along with its constraints from *N.constraints* (if such exist). CBS starts searching from a CT root node N that contains an empty set of constraints. Then, it identifies conflicts by scanning all paths in $N.\pi$. This can be done efficiently by using a hash table. A conflict $\langle a_i, a_j, v, t\rangle$ is resolved by generating two new CT child nodes N_i and N_j. CBS imposes a constraint on each of the conflicting agents, i.e., $\langle a_i, v, t\rangle$ on N_i and $\langle a_j, v, t\rangle$ on N_j. To obtain an optimal solution, CBS searches in a *best-first* search manner (lowest cost first). When a conflict-free CT node N is expanded, $N.\pi$ is returned as an optimal solution.

2.2 MAPF for Large Agents (LA-MAPF)

LA-MAPF (Li et al. 2019b) is an extension of MAPF for *large agents*. In LA-MAPF, each agent a_i is represented by a reference point (location), and has a fixed geometric shape $P_i(v) \subseteq V$ that is projected from its reference point. Each agent a_i has a start reference point s_i and a goal reference point g_i. A vertex conflict $\langle a_i, a_j, v_i, v_j, t\rangle$ between agents a_i and a_j occurs when the shapes of a_i and a_j projected by reference points v_i and v_j, respectively, at time step t overlap, i.e., $(\pi_i(t) = v_i) \wedge (\pi_j(t) = v_j) \wedge (P_i(v_i) \cap P_j(v_j) \neq \emptyset)$[1]. A solution is a conflict-free plan π, leading each agent from its start reference point to its goal reference point.

To optimally solve LA-MAPF problems, *Multi-Constraint CBS* (MC-CBS) was introduced and it works as follows. For a CT node N, conflicts are identified by simulating $N.\pi$ and detecting overlapping shapes. A vertex conflict is identified when $P_i(\pi_i(t)) \cap P_j(\pi_j(t)) \neq \emptyset$. To resolve a conflict in CT node N, MC-CBS generates two CT child nodes N_i and N_j, each with a *multi-constraint*, which is a set of constraints C_i and C_j on the reference points of the conflicting agents a_i and a_j, respectively. For a given conflict $\langle a_i, a_j, v_i, v_j, t\rangle$, there are two approaches for determining the above multi-constraint: (1) The *Asymmetric approach* adds a single constraint to one of the agents $C_i = \{\langle a_i, v_i, t\rangle\}$ and a large set of constraints to the other agent $C_j = \{\langle a_j, v', t\rangle \mid \langle a_i, a_j, v_i, v', t\rangle \text{ is a vertex conflict}\}$. (2) The *Symmetric approach* chooses one overlapping location $v \in P_i(v_i) \cap P_j(v_j)$, and sets $C_i = \{\langle a_i, v', t\rangle \,|v \in P_i(v')\}$ and $C_j = \{\langle a_j, v'', t\rangle \,|v \in P_j(v'')\}$.

[1]Similar definitions exist for swapping conflicts.

2.3 k-Robust MAPF (kR-MAPF)

kR-MAPF (Atzmon et al. 2020b) is a variant of MAPF with an additional parameter k. In kR-MAPF, each agent can be delayed up to k times. A solution to kR-MAPF is a plan π such that there are no *k-delay conflicts* in π. A k-delay conflict $\langle a_i, a_j, v, t_i, t_j\rangle$ between agents a_i and a_j occurs when both agents are planned to be in location v at time steps t_i and t_j, where $|t_i - t_j| \leq k$, i.e., $\pi_i(t_i) = \pi_j(t_j) = v$.

kR-MAPF was solved by a CBS-based algorithm, called *Improved k-Robust CBS* (IkR-CBS). IkR-CBS identifies and resolves conflicts as follows. For a CT node N, k-delay conflicts are identified in $N.\pi$ when two agents occupy the same location in time steps t_i and t_j, receptively, where $|t_i - t_j| \leq k$. To resolve k-delay conflicts, IkR-CBS uses range constraints (which are similar to the multi-constraint described above). While MC-CBS imposed multiple constraints of locations on a specific time step, IkR-CBS imposes multiple constraints of time steps (range of time steps) on a specific location. A range constraint $\langle a, v, [t_1, t_2]\rangle$ prohibits agent a to be in location v at time range $[t_1, t_2]$. For a given conflict $\langle a_i, a_j, v, t_i, t_j\rangle$, there are also two approaches for determining the constraints ranges as follows. (1) The *Asymmetric approach* imposes a constraint of a single time step on one agent $\langle a_i, v, [t_i, t_i]\rangle$, and a large range of time steps on the other agent $\langle a_j, v, [t_i - k, t_i + k]\rangle$. Of course, the range can be divided differently between the agents, in a more balanced way. (2) The *Symmetric approach* imposes the same range of time steps on both agents. Assuming that $t_i \leq t_j$, the symmetric constraints are $\langle a_i, v, [t_j - k, t_j]\rangle$ and $\langle a_j, v, [t_j - k, t_j]\rangle$ for agents a_i and a_j, respectively. There are other possible symmetric ranges that can be imposed, such as $\langle a_i, v, [t_i, t_i + k]\rangle$ and $\langle a_j, v, [t_i, t_i + k]\rangle$.

3 Generalized MAPF (G-MAPF)

In this section, we introduce G-MAPF, which generalizes MAPF for heterogeneous agents.

Definition 3.1 (State S_i)**.** A state $S_i \subset V$ is a list of locations that are occupied by agent a_i.

Definition 3.2 (Transition Function Tr_i)**.** Given a state S_i, the transition function $Tr_i(S_i)$ returns a set of m states $\{S_i^1, \ldots, S_i^m\}$, each is a state that agent a_i reaches by performing one of m applicable actions for a_i positioned at S_i.

Transition functions can define many types of actions, such as moving, waiting, rotating, jumping, and changing the shape of the agent. Moreover, a unique transition function can be defined for each agent.

The *Generalized MAPF* (G-MAPF) problem is defined on an undirected graph $G = (V, E)$ and a set of n agents labeled $A = \{a_1, \ldots, a_n\}$. Each agent a_i has a start state S_i^s, a goal state S_i^g, and a transition function Tr_i. The task is to find a *states plan* $\widehat{\pi}$ which consists of *states paths* $\widehat{\pi} = \{\widehat{\pi_1}, \ldots, \widehat{\pi_n}\}$, such that for each agent a_i the states path $\widehat{\pi_i}$ contains a sequence of states that leads from S_i^s to S_i^g. Let $\widehat{\pi_i}(t)$ be the t-th state in that sequence. Between each two consecutive states in $\widehat{\pi_i}$, there must be a transition according to Tr_i, namely, $\widehat{\pi_i}(t+1) \in Tr_i(\widehat{\pi_i}(t))$.

Similarly to classical MAPF, a solution $\widehat{\pi}$ is *valid* if it is conflict-free, namely, agents do not overlap. In G-MAPF, we

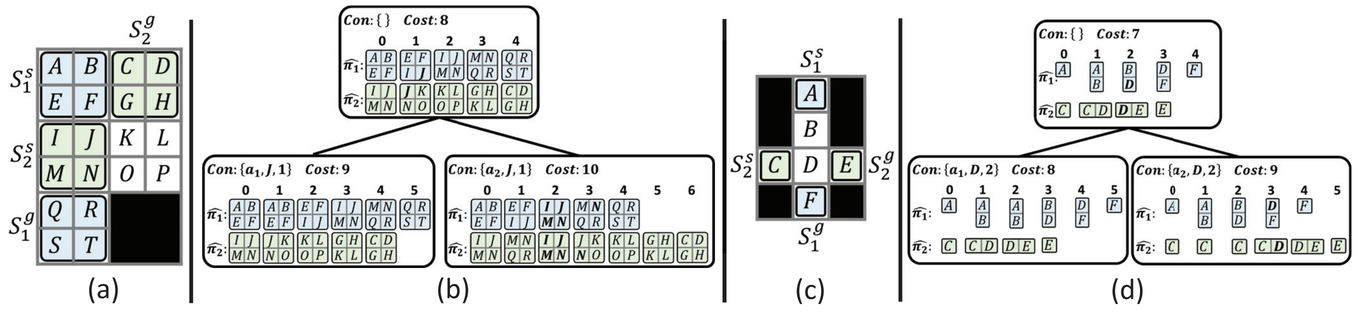

Figure 1: (a) LA-MAPF problem. (b) G-CBS's CT for LA-MAPF. (c) kR-MAPF problem. (d) G-CBS's CT for kR-MAPF.

define the following conflicts. A *Vertex conflict* $\langle a_i, a_j, v, t\rangle$ occurs when both agents a_i and a_j are planned to occupy v (v is in the overlapping area) at time step t, i.e., $v \in \widehat{\pi_i}(t) \cap \widehat{\pi_j}(t)$. A *Swapping conflict* $\langle a_i, a_j, v_i, v_j, t\rangle$ occurs when both agents a_i and a_j are planned to swap locations v_i and v_j between the same two consecutive time steps, i.e., $(v_i \in \widehat{\pi_i}(t) \cap \widehat{\pi_j}(t+1)) \wedge (v_j \in \widehat{\pi_i}(t+1) \cap \widehat{\pi_j}(t))$.

This generalization provides two contributions. First, it can generalize both LA-MAPF and kR-MAPF, as well as other types of agents. Second, while both LA-MAPF and kR-MAPF require a sophisticated method for constraining the agents to avoid collisions (multi-constraint or range constraints), this generalization defines a similar, simpler constraining method, as will be presented in Section 4.

3.1 Generalizing LA-MAPF and kR-MAPF

Next, we define LA-MAPF as a special case of G-MAPF. Recall that $P_i(v)$ denotes the geometric shape of agent a_i projected by reference point v. For each agent a_i, its start and goal states S_i^s and S_i^g can be derived from its start and goal reference points s_i and g_i as $P_i(s_i)$ and $P_i(g_i)$, respectively. For each two reference points v and v' such that agent a_i can move from v to v', the transition function Tr_i can be built easily from theirs projections, i.e., $S'_i \in Tr_i(S_i)$ where $S_i = P_i(v)$ and $S'_i = P_i(v')$.

Generalizing kR-MAPF is less intuitive, as the agents do not have shapes but rather have a unique behavior. To adjust kR-MAPF to G-MAPF, we must define the locations 'occupied' by agents as projected from their behavior. In kR-MAPF, a k-delay conflict occurs when two agents are planned to be in the same location at time steps t_1 and t_2, where $|t_1 - t_2| \leq k$. To avoid k-delay conflicts, we can say that at each time step an agent occupies its current location as well as its previous k locations (like a tail). Consequentially, each state S is a list of size $k+1$ (the current location and the previous k locations), where the first location ($S[0]$) corresponds to the agent current location, the second location ($S[1]$) corresponds to its previous location, and so on. For each agent a_i, its start state is $S_i^s = (s_i, \ldots, s_i)$ and its goal state is $S_i^g = (g_i, \ldots, g_i)$ (s_i and g_i appear in all time steps). The transition function Tr_i for a state $S_i = \{v_1, \ldots, v_{k+1}\}$ of agent a_i is defined as follows. For each edge $e = (v_1, v')$, where v_1 is the current location of the agent ($= S_i[0]$), we create a new state $S'_i = \{v', v_1, \ldots, v_k\}$ (adding v' as the current location and removing v_{k+1} from the back of the list), $S'_i \in Tr_i(S_i)$.

4 Generalized CBS (G-CBS)

Next, we define *Generalized CBS* (G-CBS), a CBS-based solver for G-MAPF. We start by defining the constraint used by G-CBS, called *occupation constraint*.

Definition 4.1 (Occupation Constraint)**.** An *occupation constraint* $\langle a_i, v, t\rangle$ prohibits *any* location in the state of agent a_i to occupy location v at time step t, i.e., $v \notin \widehat{\pi_i}(t)$.

As in CBS, G-CBS works in two levels. The low-level of G-CBS gets an agent start state, goal state, its transition function, and a set of occupation constraints, and returns a *states path* for this agent. Here also, any single-agent solver can be used as the low-level solver. The low-level solver will only generate states that satisfy all occupation constraints and it must use the corresponding transition function to generate successive states. The high-level of G-CBS searches the CT, where each CT node N contains a *states plan* $N.\widehat{\pi}$. G-CBS runs a best-first search over the CT, and it identifies and resolves conflicts as follows.

Identifying conflicts. For each CT node N, G-CBS simulates $N.\widehat{\pi}$ and detects locations that are occupied by two agents a_i and a_j at the same time t, i.e., $\widehat{\pi_i}(t) \cap \widehat{\pi_j}(t) \neq \emptyset$.

Resolving conflicts. After detecting conflicts in CT node N, G-CBS chooses one of the conflicts to resolve. For a chosen conflict $\langle a_i, a_j, v, t\rangle$, two new CT child nodes N_i and N_j are generated, with the additional occupation constraints $\langle a_i, v, t\rangle$ and $\langle a_j, v, t\rangle$, respectively. Swapping conflicts are resolved in a similar way.

Example of LA-MAPF as G-MAPF. Figure 1(a) shows an example of a LA-MAPF problem with two agents a_1 and a_2, each with a size of 2x2. Their start states are $S_1^s = (A, B, E, F)$ and $S_2^s = (I, J, M, N)$ (reference points A and I), and their goal states are $S_1^g = (Q, R, S, T)$ and $S_2^g = (C, D, G, H)$ (reference points Q and C). Figure 1(b) presents the corresponding CT generated by G-CBS. First, the root calculates a states path for each agent separately. Then, it identifies conflicts. Here, one conflict is identified – $\langle a_1, a_2, J, 1\rangle$; both agents occupy J at time step 1. The conflict is resolved by imposing an occupation constraint on each of the agents, and generating two corresponding CT child nodes. The minimal cost node is now selected next

	30x30				den502d			
	n	Rate	#Exp	Time	n	Rate	#Exp	Time
MC-CBS	2	99%	289	3,204	5	86%	6	8,189
G-CBS		97%	269	1,913		94%	7	4,498
MC-CBS	4	77%	1,155	5,188	10	51%	23	19,994
G-CBS		74%	1,207	4,626		70%	23	9,711
MC-CBS	6	40%	2,221	8,974	15	25%	39	28,437
G-CBS		37%	2,200	9,780		36%	38	15,679

Table 1: LA-MAPF experiment.

	30x30				den502d			
	n	Rate	#Exp	Time	n	Rate	#Exp	Time
IkR-CBS	5	100%	11	15	5	94%	1	1,021
G-CBS		100%	10	523		91%	1	2,541
IkR-CBS	10	96%	119	1,277	10	81%	31	3,331
G-CBS		93%	163	1,571		74%	37	5,783
IkR-CBS	15	89%	201	2,223	15	73%	50	19,287
G-CBS		87%	216	3,540		51%	52	28,285

Table 2: kR-MAPF experiment.

(the left child), and a valid solution with a cost of 9 is returned. Now, consider MC-CBS. MC-CBS with symmetric constraints would build the exact same CT. The occupation constraint imposed by G-CBS on location J is identical to the vertex constraint imposes by MC-CBS on reference points E, F, I, and J. [2]

Example of kR-MAPF as G-CBS. A kR-MAPF problem ($k = 1$) with two agents a_1 and a_2 is presented in Figure 1(c). Their start and goal states are $S_1^s = (A, A)$, $S_1^g = (F, F)$, and $S_2^s = (C, C)$, $S_2^g = (E, E)$ ($s_1 = A$, $g_1 = F$, $s_2 = C$, and $g_2 = E$). Figure 1(d) shows the CT created by G-CBS. In the CT, the root node identifies a conflict in location D at time step 2. To resolve the conflict, the occupation constraints $\langle a_1, D, 2\rangle$ and $\langle a_2, D, 2\rangle$ are imposed on the agents. Next, the minimal cost CT node is expanded (the left child). No conflicts are identified and a solution with a cost of 8 is returned. Now, consider IkR-CBS for solving the same problem. IkR-CBS with symmetric constraints will impose the constraints $\langle a_1, D, [1-2]\rangle$ and $\langle a_2, D, [1-2]\rangle$ on the reference point. This is identical to the imposed occupation constraint $\langle a_1, D, 2\rangle$ done by G-CBS.[2]

Theorem 4.1. *G-CBS is sound, complete, and optimal.*

Proof outline. G-CBS determines that a CT node is a CT goal node only if it has no conflicts. Thus, the returned solution is a valid G-MAPF solution. As agents cannot occupy the same location at the same time step, by generating two CT child nodes N_1 and N_2 for CT node N, each valid solution that satisfies $N.constraints$ also satisfies either $N_1.constraints$ or $N_2.constraints$. Therefore, by performing a best-first search over the CT, G-CBS is guaranteed to return an optimal solution. □

5 Experimental Results

To evaluate G-CBS, we compared it to both MC-CBS and IkR-CBS with symmetric constraints, over 30x30 grids with 10% obstacles and over the den502d map from the movingai repository (Sturtevant 2012). In each experiment we created 70 instances with n randomly allocated agents and measured the average success rate (1min timeout; denoted by Rate), high-level expansions (denoted by #Exp), and time (in ms). The expansions and time were averaged over all instances that were solved by both solvers.

Table 1 shows the results for the comparison between MC-CBS and G-CBS. In this experiment we randomly allocated agents (n), each with a square of size of 3x3. As can be seen, the results are relatively similar for both G-CBS and MC-CBS, with a clear advantage for G-CBS in the den502d map. This advantage is a result of the compact constraining method used by G-CBS. However, this is probably a matter of the implementation of MC-CBS.

The comparison between IkR-CBS and G-CBS is presented in Table 2. Here, each randomly allocated agent has a robustness of $k = 2$. We can see that IkR-CBS was a slightly faster than G-CBS, with a better success rate, and with almost the same number of expansions, on both 30x30 grids and on the den502d map. This is a result of the runtime of the low-level. While IkR-CBS identifies low-level nodes as similar based on a single location, G-CBS identifies similar nodes based on their entire occupation to allow for general occupation constraints, and hence consumes more time.

6 Related Work

Sharon et al. (2012) solved MAPF using CBS with *Meta-agents*. Agents that conflicted more than B times during the search were merged and solved together as one meta-agent. G-CBS can define a meta-agent as one agent that occupies the multiple locations occupied by the meta-agent.

Agents can also be defined in MAPF as train-agents (Atzmon, Diei, and Rave 2019), i.e., agents that occupy a sequence of locations. G-CBS can also define agents as trains by setting their states as the occupied sequence of locations.

MAPF was also extended for a set of convoys (Thomas, Deodhare, and Murty 2015). Each convoy was defined by a set of edges. Thus, two agents can cross the same location without causing a collision. Adjusting G-CBS for such convoys require defining the states and constraints on edges, instead of locations, and hence left for future work.

7 Conclusions

In this paper we generalized MAPF for heterogeneous agents (G-MAPF). We showed how both LA-MAPF and kR-MAPF are special cases of G-MAPF. For solving G-MAPF, we introduced G-CBS, and showed experimentally that although G-MAPF is flexible, G-CBS performs relatively close to MC-CBS and IkR-CBS, which are hand tailored for one type of agent. Future work can suggest policies for selecting conflicts to resolve (Boyarski et al. 2015) and can adjust G-CBS for the case of probabilistic environments (Wagner and Choset 2017; Atzmon et al. 2020a).

8 Acknowledgements

This research was supported by the Israel Science Foundation (ISF) under grant number 844/17 and by the United

[2]Occupation constraints can be adjusted to be identical to asymmetric constraints. However, it will need to consider the specific shape of the agent. Thus, G-CBS does not directly support it.

States-Israel Binational Science Foundation (BSF) under grant number 2017692.

References

Atzmon, D.; Stern, R.; Felner, A.; Sturtevant, N. R.; and Koenig, S. 2020a. Probabilistic robust multi-agent path finding. In *the International Conference on Automated Planning and Scheduling (ICAPS)*.

Atzmon, D.; Stern, R.; Felner, A.; Wagner, G.; Barták, R.; and Zhou, N. 2020b. Robust multi-agent path finding and executing. *Journal of Artificial Intelligence Research* 67:549–579.

Atzmon, D.; Diei, A.; and Rave, D. 2019. Multi-train path finding. In *the International Symposium on Combinatorial Search (SoCS)*, 125–129.

Boyarski, E.; Felner, A.; Stern, R.; Sharon, G.; Tolpin, D.; Betzalel, O.; and Shimony, S. E. 2015. ICBS: improved conflict-based search algorithm for multi-agent pathfinding. In *the International Joint Conference on Artificial Intelligence (IJCAI)*, 740–746.

Felner, A.; Stern, R.; Shimony, S. E.; Boyarski, E.; Goldenberg, M.; Sharon, G.; Sturtevant, N. R.; Wagner, G.; and Surynek, P. 2017. Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges. In *the International Symposium on Combinatorial Search (SoCS)*, 29–37.

Felner, A.; Li, J.; Boyarski, E.; Ma, H.; Cohen, L.; Kumar, T. K. S.; and Koenig, S. 2018. Adding heuristics to conflict-based search for multi-agent path finding. In *the International Conference on Automated Planning and Scheduling (ICAPS)*, 83–87.

Gange, G.; Harabor, D.; and Stuckey, P. J. 2019. Lazy CBS: implicit conflict-based search using lazy clause generation. In *the International Conference on Automated Planning and Scheduling (ICAPS)*, 155–162.

Lam, E.; Bodic, P. L.; Harabor, D.; and Stuckey, P. J. 2019. Branch-and-cut-and-price for multi-agent pathfinding. In *the International Joint Conference on Artificial Intelligence (IJCAI)*, 1289–1296.

Li, J.; Harabor, D.; Stuckey, P. J.; Ma, H.; and Koenig, S. 2019a. Disjoint splitting for multi-agent path finding with conflict-based search. In *the International Conference on Automated Planning and Scheduling (ICAPS)*, 279–283.

Li, J.; Surynek, P.; Felner, A.; Ma, H.; Kumar, T. K. S.; and Koenig, S. 2019b. Multi-agent path finding for large agents. In *the AAAI Conference on Artificial Intelligence (AAAI)*, 7627–7634.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2012. Meta-agent conflict-based search for optimal multi-agent path finding. In *the International Symposium on Combinatorial Search (SoCS)*.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artif. Intell.* 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 *the International Symposium on Combinatorial Search (SoCS)*, 151–159.

Sturtevant, N. R. 2012. Benchmarks for grid-based pathfinding. *Computational Intelligence and AI in Games* 4(2):144–148.

Surynek, P. 2010. An optimization variant of multi-robot path planning is intractable. In *the AAAI Conference on Artificial Intelligence (AAAI)*, 1261–1263.

Surynek, P. 2012. Towards optimal cooperative path planning in hard setups through satisfiability solving. In *the Pacific Rim International Conference on Artificial Intelligence (PRICAI)*, 564–576.

Thomas, S.; Deodhare, D.; and Murty, M. N. 2015. Extended conflict-based search for the convoy movement problem. *IEEE Intelligent Systems* 30:60–70.

Wagner, G., and Choset, H. 2017. Path planning for multiple agents under uncertainty. In *the International Conference on Automated Planning and Scheduling (ICAPS)*, 577–585.

Yu, J., and LaValle, S. M. 2013. Structure and intractability of optimal multi-robot path planning on graphs. In *the AAAI Conference on Artificial Intelligence (AAAI)*, 1444–149.