Adding Heuristics to Conflict-Based Search for Multi-Agent Path Finding

Ariel Felner
SISE Department
Ben-Gurion University
Israel
felner@bgu.ac.il

Jiaoyang Li
CS Department
Univ. of Southern California
USA
jiaoyanl@usc.edu

Eli Boyarski
SISE Department
Ben-Gurion University
Israel
eli.boyarski@gmail.com

Hang Ma
Liron Cohen
T. K. Satish Kumar
Sven Koenig
Univ. of Southern California
USA

Abstract

Conflict-Based Search (CBS) and its enhancements are among the strongest algorithms for the multi-agent path-finding problem. However, existing variants of CBS do not use any heuristics that estimate future work. In this paper, we introduce different admissible heuristics for CBS by aggregating cardinal conflicts among agents. In our experiments, CBS with these heuristics outperforms previous state-of-the-art CBS variants by up to a factor of five.

1 Introduction and Overview

The *Multi-Agent Path-Finding* (MAPF) problem is specified by a graph $G = (V, E)$ and a set of k agents $\{a_1 \dots a_k\}$, where agent a_i has start location $s_i \in V$ and goal location $g_i \in V$. Time is discretized into time steps, and agent a_i is in location s_i at time step t_0. Between successive time steps, each agent can either *move* to an adjacent empty location or *wait* in its current location. Both move and wait actions incur a cost of one. A *path* for agent a_i is a sequence of move and wait actions that lead agent a_i from location s_i to location g_i. A *conflict* between two paths is a tuple $\langle a_i, a_j, v, t\rangle$, meaning that agents a_i and a_j both occupy the same vertex v at the same time step t. A *solution* is a set of k paths, one for each agent. The objective is to find a *conflict-free solution*. Felner et al. (2017) and Ma and Koenig (2017) provide surveys of different settings, applications and algorithms for MAPF.

Conflict-Based Search (CBS) (Sharon et al. 2012a; 2015) is an optimal two-level search-based algorithm for MAPF for which also suboptimal variants exist (Barer et al. 2014; Cohen, Uras, and Koenig 2015; Cohen et al. 2016). It is useful for many real-world applications such as automated warehousing (Hönig et al. 2016; Ma, Kumar, and Koenig 2017) and other robotics applications (Hoenig et al. 2016). We focus here on the classic CBS variant that minimizes the *sum-of-costs* objective (Standley 2010; Standley and Korf 2011; Sharon et al. 2013; 2015) (that is, the sum of the path cost of all agents), which is NP-hard (Yu and LaValle 2013; Ma et al. 2016b).[1]

[1]There also exist CBS variants that minimize the makespan objective (that is, the largest individual path cost among all agents) but our heuristics apply only to the sum-of-costs objective.

A number of enhancements to and generalizations of CBS have been introduced (Sharon et al. 2015; Ma et al. 2016a). *Improved CBS* (ICBS) (Boyarski et al. 2015) is one of its strongest variants. However, all existing CBS variants use only the costs of the (possibly conflicting) paths in the nodes of the CBS constraint tree as the costs of the nodes. These costs can be considered to be the g-values of the nodes. Our contribution is to calculate admissible heuristics and thus add h-values to the costs of these nodes. To this end, we introduce several ways of aggregating *cardinal conflicts* (Boyarski et al. 2015) among agents. CBS with such h-values is called *ICBS plus h* (ICBS-h). It improves upon CBS in the same way that A* improves upon Dijkstra's algorithm. In our experiments, it outperforms CBS and ICBS by up to a factor of five.

2 Conflict-Based Search (CBS)

CBS has two levels. The high level of CBS searches the binary *constraint tree* (CT). Each node $N \in CT$ contains: **(1)** a set of constraints imposed on the agents ($N.constraints$), where a *constraint* imposed on agent a_i is a tuple $\langle a_i, v, t\rangle$, meaning that agent a_i is prohibited from occupying vertex v at time step t; **(2)** a single solution ($N.solution$) that satisfies all constraints; and **(3)** the cost of solution $N.solution$ ($N.cost$), that is, the sum of the path costs of all agents. The root node contains an empty set of constraints. The high level performs a best-first search on the CT, ordering the nodes according to their costs. Ties are broken in favor of nodes whose solutions have fewer conflicts.

Generating a node in the CT. Given a node N, the low level of CBS finds a shortest path for each agent that satisfies all constraints in node N imposed on the agent, for example, by using A* with h-values that are the true distances when ignoring all constraints (Sharon et al. 2015).

Expanding a node in the CT. Once CBS has chosen node N for expansion, it checks the solution $N.solution$ for conflicts. If it is conflict-free, then node N is a goal node and CBS returns its solution. Otherwise, CBS *splits* node N on one of the conflicts $\langle a_i, a_j, v, t\rangle$ as follows. In any conflict-free solution, at most one of the conflicting agents a_i and a_j can occupy vertex v at time step t. Therefore, at least one of the constraints $\langle a_i, v, t\rangle$ or $\langle a_j, v, t\rangle$ must be satisfied. Consequently, CBS *splits* node N by generating two children of node N, each with a set of constraints that adds one of

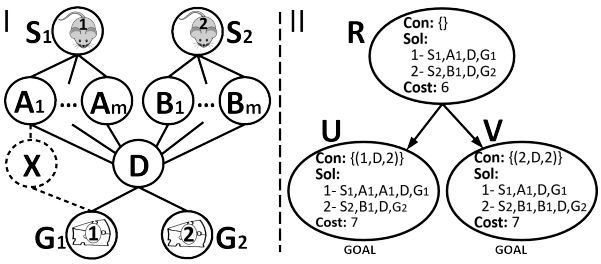


Figure 1: (I) A MAPF instance. (II) Its CT.

these two constraints to the set $N.constraints$. Thus, CBS imposes an additional constraint on only one agent for each child and thus has to re-plan the path of only that agent.

Figure 1(I) shows an example of a two-agent MAPF instance. Each agent (mouse) must plan a path to its respective goal location (piece of cheese). Figure 1(II) shows the corresponding CT. Its root node R contains an empty set of constraints, and the low level of CBS finds the shortest path $\langle S_1, A_1, D, G_1\rangle$ of length 3 for agent 1 and the shortest path $\langle S_2, B_1, D, G_2\rangle$ of length 3 for agent 2. Thus, the cost of the root node is $R.cost = 3 + 3 = 6$. The solution of the root node has conflict $\langle 1, 2, D, 2\rangle$ since agents 1 and 2 both occupy vertex D at time step 2. Consequently, CBS splits the root node. The new left child U (right child V) of the root node adds the constraint $\langle 1, D, 2\rangle$ ($\langle 2, D, 2\rangle$). In node U, the low level of CBS finds the new shortest path $\langle S_1, A_1, A_1, D, G_1\rangle$ of length 4 (that includes a wait action) for agent 1, while the shortest path of agent 2 is identical to the one in the root node since no new constraints are imposed on agent 2. Thus, the cost of node U is $U.cost = 4 + 3 = 7$. Since the solution of node U is conflict-free, it is a goal node and CBS returns its solution.

2.1 Improved CBS (ICBS)

CBS arbitrarily chooses conflicts to split and arbitrarily chooses paths in the low-level. However, poor choices can significantly increase the size of its CT and thus its runtime. *Improved CBS* (ICBS) (Boyarski et al. 2015) addresses this issue with two improvements to CBS.

Improvement 1: Splitting on cardinal conflicts. ICBS classifies conflicts into three types. A conflict C is *cardinal* iff, when CBS uses the conflict to split node N, the cost of each of the two resulting children of node N is larger than the cost of node N. Conflict C is *semi-cardinal* iff the cost of one child is larger than the cost of N and the cost of the other child is equal to the cost of N. Finally, conflict C is *non-cardinal* iff the cost of each of the two children is equal to the cost of node N. For example, in Figure 1(I), the conflict $\langle 1, 2, D, 2\rangle$ is cardinal for the root node since the costs of nodes U and V are both 7. If the dotted lines are added, then the conflict becomes semi-cardinal since the cost of node V remains 7 while the cost of node U becomes 6 since now a shortest path of length 3 via location X exists for agent 1.

ICBS must first choose a cardinal conflict (if one exists) when splitting a node N. The costs of both children of N are then larger than the cost $N.cost$ of node N, and the best-first search of the high level of CBS thus expands some other unexpanded node with cost $N.cost$ next (if available) rather than nodes in the CT subtree rooted at N. This can result in smaller CTs and thus make the search more efficient.

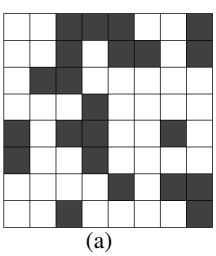
Improvement 2: Bypassing conflicts. When ICBS has to choose a semi-cardinal or non-cardinal conflict when splitting a node N, it can sometimes modify one of the paths in the solution of node N to make the conflict disappear without splitting the node. If, when splitting node N, one of the solutions of the resulting children of node N includes an alternative path for an agent with the same cost as the original path but without the conflict and with fewer conflicts overall, then this path replaces the path of the agent in node N and the node is not split. This can result in smaller CTs and thus make the search more efficient.

3 ICBS-h

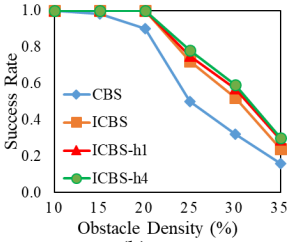
The best-first search of the high level of all existing CBS variants uses only the cost of a CT node as its priority. This value can be considered to be the g-value of the node. We want to add an admissible (that is, non-overestimating) h-value to its priority to make it more informed, resulting in ICBS with heuristics or, in short, *ICBS plus h* (ICBS-h). Our idea is simple: If the solution of a node N contains one or more cardinal conflicts, then an h-value of one is admissible for node N because the cost of any of its descendants in the CT with a conflict-free solution is at least $N.cost + 1$. The reason is that the paths in their solutions cannot be shorter than the ones in the solution of node N since the same or more constraints are imposed on the agents, and the length of the path of at least one of the conflicting agents has to increase by at least one. However, if the solution of a node contains x cardinal conflicts, then an h-value of x is not necessarily admissible for the node, which is why current CBS variants use the number of conflicts only to break ties among nodes with the same priority. We now show several ways to calculate admissible h-values by aggregating cardinal conflicts among agents. We use a *conflict graph* $G_{CF} = (V_{CF}, E_{CF})$ of node N. Each vertex $v_i \in V_{CF}$ corresponds to an agent a_i that is involved in at least one cardinal conflict. Each edge $e = (v_i, v_j) \in E_{CF}$ expresses that there is at least one cardinal conflict between agents a_i and a_j. Similar conflict graphs were used in the context of heuristic search for sliding tile puzzles (Felner, Korf, and Hanan 2004) and cost-optimal planning (Pommerening, Röger, and Helmert 2013).

3.1 Disjoint Cardinal Conflicts

Disjoint cardinal conflicts are cardinal conflicts between disjoint pairs of agents. If the solution of a node N contains x disjoint cardinal conflicts, then $h = x$ is admissible for node N since the length of the path of at least one agent of each agent pair has to increase by at least one. Thus, we can use the size of a *matching* (that is, a set of edges without common vertices) in the conflict graph of node N as its admissible h-value. ICBS-h_1 finds a greedy matching as follows: It repeatedly chooses an arbitrary edge (representing a cardinal conflict) in the conflict graph, increases the h-value of node N by one and then deletes all edges that are incident on both



(a)



(b)

Obs	Runtime (ms)				Expanded Nodes			
	CBS	ICBS	ICBS-h_1	ICBS-h_4	CBS	ICBS	ICBS-h_1	ICBS-h_4
10%	2,771	78	**62**	63	5,306	143	108	**107**
15%	2,264	1,070	561	**386**	4,263	552	264	**205**
20%	8,940	1,367	837	**764**	18,752	1,195	700	**657**
25%	95,920	10,650	3,771	**3,215**	169,000	11,232	4,568	**4,030**
30%	142,825	21,773	**6,578**	6,580	262,320	20,186	7,896	**7,767**
35%	113,673	20,276	4,539	**3,582**	223,764	25,666	6,818	**5,003**

(c)

Figure 2: Experimental results on 8×8 grids.

vertices of the chosen edge from the conflict graph, until no edge remains. ICBS-h_2 finds a maximum matching in time $O(\sqrt{|V_{CF}|}|E_{CF}|)$ (Peterson and Loui 1988) to improve the h-values. A maximum matching takes more time to compute but can potentially result in more informed h-values.

3.2 Minimum Vertex Cover of the Conflict Graph

The length of the path of at least one agent of each conflicting agent pair in a cardinal conflict has to increase by at least one. Thus, we can use the size of a minimum *vertex cover* (that is, a set of vertices such that each edge is incident on at least one vertex in the set) of the conflict graph of node N as its admissible h-value (Felner, Korf, and Hanan 2004; Pommerening, Röger, and Helmert 2013). Finding a minimum vertex cover is NP-hard (Xu et al. 2018). ICBS-h_3 thus greedily determines a lower bound on the size of a minimum vertex cover. Finally, ICBS-h_4 finds a minimum vertex cover as follows: When CBS generates a node N in the CT, it replans the path of only one agent. Consequently, only edges incident on the vertex corresponding to this agent can appear in or disappear from the conflict graph. In the former (latter) case, the size of the minimum vertex cover and thus the h-value of node N increase (decrease) by at most one. This property can be exploited to calculate the h-value of node N with an algorithm that determines in time $O(2^q n)$ whether a given graph with n vertices has a vertex cover of size q (Downey and Fellows 1995), by executing it at most twice (namely for $q = h - 1$ and, if that is unsuccessful and $h < k - 1$, also for $q = h$, where h is the h-value of the parent of node N). Note that $k - 1$ is an upper bound on the h-values since the size of the minimum vertex cover of the conflict graph (with at most k vertices) is at most $k - 1$.

4 Experimental Results

We experimented with CBS, ICBS and all four variants of ICBS-h. In practice, the different heuristics performed relatively similarly and we only provide results for ICBS-h_1 and ICBS-h_4, which are the weakest and strongest among the four. Our code is written in C#, and our experiments are conducted on a 2.80 GHz Intel core i7-7700 laptop with 8 GB RAM. We set a runtime limit of 5 minutes, as used before (Sharon et al. 2012a; 2015).

Experiment 1. We experimented with 10 agents on 4-neighbor 8×8 grids with 10% to 35% randomly-placed obstacles (Figure 2(a)). Figure 2(b) shows the *success rate* (that is, percentage of instances solved within the runtime limit) out of 100 random instances. CBS performs worst since it essentially uses zero h-values. ICBS had already been shown to significantly outperform CBS (Boyarski et al. 2015), and this trend is confirmed here. In a sense, ICBS is similar to ICBS-h with h-values that are one for nodes with cardinal conflicts and zero otherwise. ICBS-h_1 and ICBS-h_4 perform even better since they use even larger h-values. ICBS-h_4 performs best even though it solves the NP-hard minimum vertex cover problem repeatedly. A similar phenomenon was reported by Felner, Korf, and Hanan (2004) and can be explained here with the conflict graph being sparse. Figure 2(c) shows the runtime and number of expanded nodes. The numbers for the three ICBS variants are averaged over the instances solved by all three of them while the numbers for CBS are averaged over only the smaller number of instances solved by it (and thus would be much larger if the same instances had been used in both cases). The trends are similar as for the success rate. ICBS-h_4 improves the runtime and number of expanded nodes by a factor of up to 5 over ICBS.

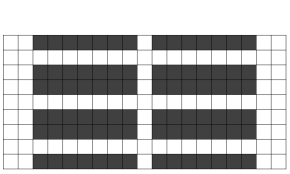
Experiment 2. We repeated Experiment 1 on a 4-neighbor warehouse grid (Figure 3(a)) and on the standard 4-neighbor Dragon Age: Origin computer game grid BRC202d (Sturtevant 2012) (Figure 3(d)). Figures 3(b,c) and 3(e,f) show similar trends as for Experiment 1. ICBS-h_4 now improves the runtime and number of expanded nodes only by a factor of up to 2-3 over ICBS, which can be explained with the environments being less congested.

Experiment 3. To show the potential of adding heuristics when problems scale up Table 1 shows the number of conflicts, the number of cardinal conflicts and the h-value of the root node calculated by ICBS-h_1 and ICBS-h_4 for the instances used so far plus 100 instances for 4-neighbor 15×15 grids with 10% obstacles. The number of cardinal conflicts and thus the h-values and the importance of using ICBS-h increase with the obstacle and agent densities and thus the difficulty of MAPF instances. For example, the h-value of the root node on 15×15 grids with 100 agents is about 16 on average for ICBS-h_4, allowing it to prune many nodes that areexpanded by CBS and/or ICBS.

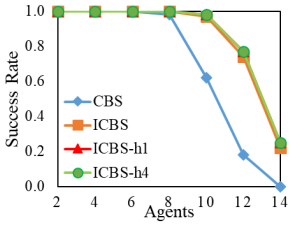
5 Possible Slowdown

Depending on tie breaking, A* with admissible h-values can expand more nodes than with zero h-values if the admissible h-values of some non-goal nodes are zero. Such zero h-values for non-goal nodes must exist if zero-cost edges are allowed and are connected to the goal.

The CT contains zero-cost edges in case some of its nodes



(a)

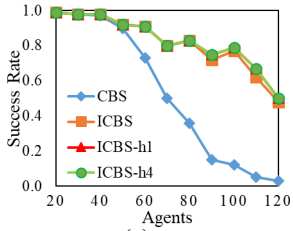


(b)

k	Runtime (ms)				Expanded Nodes			
	CBS	ICBS	ICBS-h_1	ICBS-h_4	CBS	ICBS	ICBS-h_1	ICBS-h_4
4	2.43	2.17	**1.66**	1.69	4.32	3.36	2.67	**2.63**
6	45	24	**14**	**14**	62	31	18	**17**
8	6,047	428	242	**232**	6,957	440	247	**242**
10	31,581	19,563	11,364	**11,363**	35,014	16,682	10,225	**10,027**
12	60,490	41,670	26,777	**26,347**	69,254	35,873	23,759	**23,486**
14	-	100,729	66,619	**66,205**	-	78,911	53,515	**53,125**

(c)

(d)



(e)

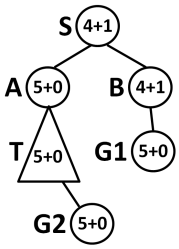
k	Runtime (ms)				Expanded Nodes			
	CBS	ICBS	ICBS-h_1	ICBS-h_4	CBS	ICBS	ICBS-h_1	ICBS-h_4
20	25	25	**24**	**24**	2.37	1.74	**1.70**	**1.70**
40	5,080	153	**120**	**120**	258	9.39	7.71	**7.70**
60	19,609	717	383	**379**	1,070	26	**17**	**17**
80	59,775	8,807	**3,953**	4,173	2,922	324	**156**	**156**
100	23,852	14,649	**8,799**	8,983	873	361	203	**200**
120	25,485	43,996	**16,953**	17,630	976	984	431	**429**

(f)

Figure 3: Experimental results on a warehouse grid (top) and a computer game grid (bottom).

	All Conflicts	Cardinal Conflicts	h_1	h_4		All Conflicts	Cardinal Conflicts	h_1	h_4
Obs	8x8 Grids				k	Warehouse Grid			
10%	6.36	1.01	0.74	0.76	4	4.34	1.33	0.41	0.45
15%	7.83	1.54	0.97	1.00	6	11.08	3.60	0.93	1.01
20%	8.95	3.84	1.78	1.95	8	18.09	5.59	1.45	1.55
25%	12.41	7.31	2.39	2.78	10	34.39	12.74	1.99	2.32
30%	14.12	9.33	2.76	3.33	12	52.93	16.69	2.42	2.78
35%	17.39	14.64	3.31	4.22	14	66.53	21.26	3.18	3.74
k	15x15 Grids				k	Computer Game Grid			
20	16.85	1.48	1.03	1.06	20	4.69	0.14	0.12	0.12
40	68.81	6.26	3.54	3.82	40	13.56	0.71	0.51	0.51
60	154.69	13.65	6.65	7.17	60	29.62	1.74	1.05	1.06
80	269.52	22.77	10.07	11.14	80	62.78	3.15	1.92	1.96
100	415.20	36.77	14.40	16.03	100	84.41	5.83	2.67	2.73

Table 1: Number of conflicts and h-value of root node.



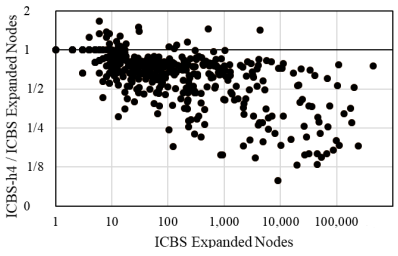


Figure 4: ICBS versus ICBS-h.

were split based on semi-cardinal or non-cardinal conflicts. Admissible h-values of such non-goal nodes have to be zero in case they are connected to goal nodes via one or more zero-cost edges. Thus, ICBS-h can expand more nodes than CBS. Figure 4(left) shows an example CT. The expressions inside the nodes are the sums of their g-values and h-values. Nodes $G1$ and $G2$ are goal nodes. CBS first expands node S. It then expands node B since it has a smaller g-value (and thus cost and priority) than node A. Finally, it expands node $G1$ (at which point it stops) since it has the same g-value as node A but a smaller number of conflicts (since the solutions in goal nodes are conflict-free while the ones in non-goal nodes are not). ICBS-h first expands node S as well. It then expands node A if it has a smaller number of conflicts than node B since it has the same sum of g-value and h-value (and thus priority) as node B. (It would also expand node A in case it broke ties in favor of nodes with smaller h-values.) It can then expand the entire CT subtree rooted at node A and finally node $G2$ (at which point it stops). In this case, ICBS-h expands more nodes than CBS. If the h-values of non-goal nodes were strictly larger than zero, ICBS-h would avoid this issue since it would expand node B (instead of node A) and finally node $G1$ (at which point it stops).

In our experiments, ICBS-h expanded more nodes than ICBS for fewer than 5% of the instances, and these cases do not significantly contribute to the average number of expanded nodes. Figure 4(right) shows the ratio of the number of expanded nodes by ICBS-h_4 and ICBS (as a function of the number of expanded nodes by ICBS) on the instances of 4-neighbor 8×8 grids that were solved by ICBS. ICBS expanded fewer nodes than ICBS-h_4 for only 22 out of 447 instances.

6 Conclusions and Future Research

We have provided first evidence that admissible h-values are beneficial for CBS. There are several possible directions for future research.

Direction 1. It is challenging to derive h-values for Meta-Agent CBS (MA-CBS) (Sharon et al. 2012b), which is a variant of CBS that can merge two conflicting agents into a meta-agent instead of using their conflict to split a node. MA-CBS treats a meta-agent as a single composite agent and reasons only about conflicts among meta-agents, which complicates the aggregation of cardinal conflicts among the individual agents that form meta-agents.

Direction 2. It might be possible to use sophisticated h-values for cost-optimal planning to develop even more informed h-values for CBS, such as linear programming-based h-values (Pommerening et al. 2015).

7 Acknowledgments

The research at Ben-Gurion University was supported by the Israel Ministry of Science and the Czech Ministry of Education and by Israel Science Foundation grant 844/17. The research at the University of Southern California was supported by National Science Foundation grants 1724392, 1409987 and 1319966. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies or the U.S. government.

References

Barer, M.; Sharon, G.; Stern, R.; and Felner, A. 2014. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *Proceedings of the Annual Symposium on Combinatorial Search*.

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

Cohen, L.; Uras, T.; Kumar, S.; Xu, H.; Ayanian, N.; and Koenig, S. 2016. Improved solvers for bounded-suboptimal multi-agent path finding. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 3067–3074.

Cohen, L.; Uras, T.; and Koenig, S. 2015. Feasibility study: Using highways for bounded-suboptimal multi-agent path finding. In *Proceedings of the Symposium on Combinatorial Search*, 2–8.

Downey, R., and Fellows, M. 1995. Parameterized computational feasibility. In *Feasible Mathematics II*, 219–244.

Felner, A.; Stern, R.; Shimony, S.; Boyarski, E.; Goldenberg, M.; Sharon, G.; Sturtevant, N.; Wagner, G.; and Surynek, P. 2017. Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges. In *Proceedings of the Annual Symposium on Combinatorial Search*, 20–37.

Felner, A.; Korf, R. E.; and Hanan, S. 2004. Additive pattern database heuristics. *Journal of Artificial Intelligence Research* 22:279–318.

Hoenig, W.; Kumar, S.; Ma, H.; Koenig, S.; and Ayanian, N. 2016. Formation change for robot groups in occluded environments. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 4836–4842.

Hönig, W.; Kumar, S.; Cohen, L.; Ma, H.; Xu, H.; Ayanian, N.; and Koenig, S. 2016. Multi-agent path finding with kinematic constraints. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 477–485.

Ma, H., and Koenig, S. 2017. AI buzzwords explained: Multi-agent path finding (MAPF). *AI Matters* 3(3):15–19.

Ma, H.; Koenig, S.; Ayanian, N.; Cohen, L.; Hoenig, W.; Kumar, S.; Uras, T.; Xu, H.; Tovey, C.; and Sharon, G. 2016a. Overview: Generalizations of multi-agent path finding to real-world scenarios. In *Proceedings of the IJCAI-16 Workshop on Multi-Agent Path Finding*.

Ma, H.; Tovey, C.; Sharon, G.; Kumar, S.; and Koenig, S. 2016b. Multi-agent path finding with payload transfers and the package-exchange robot-routing problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 3166–3173.

Ma, H.; Kumar, S.; and Koenig, S. 2017. Multi-agent path finding with delay probabilities. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 3605–3612.

Peterson, P., and Loui, M. 1988. The general maximum matching algorithm of Micali and Vazirani. *Algorithmica* 3(1–4):511–533.

Pommerening, F.; Röger, G.; Helmert, M.; and Bonet, B. 2015. Heuristics for cost-optimal classical planning based on linear programming. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 4303–4309.

Pommerening, F.; Röger, G.; and Helmert, M. 2013. Getting the most out of pattern databases for classical planning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2357–2364.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. 2012a. Conflict-based search for optimal multi-agent path finding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 563–569.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. 2012b. Meta-agent conflict-based search for optimal multi-agent path finding. In *Proceedings of the Annual Symposium on Combinatorial Search*.

Sharon, G.; Stern, R.; Goldenberg, M.; and Felner, A. 2013. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence* 195:470–495.

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

Standley, T., and Korf, R. 2011. Complete algorithms for cooperative pathfinding problems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 668–673.

Standley, T. 2010. Finding optimal solutions to cooperative pathfinding problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 173–178.

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

Xu, H.; Sun, K.; Koenig, S.; and Kumar, S. 2018. A warning propagation-based linear-time-and-space algorithm for the minimum vertex cover problem on giant graphs. In *Proceedings of the International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*.

Yu, J., and LaValle, S. 2013. Structure and intractability of optimal multi-robot path planning on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1444–1449.