Feasibility Study: Using Highways for Bounded-Suboptimal Multi-Agent Path Finding

Liron Cohen
Department of Computer Science
University of Southern California
lironcoh@usc.edu

Tansel Uras
Department of Computer Science
University of Southern California
turas@usc.edu

Sven Koenig
Department of Computer Science
University of Southern California
skoenig@usc.edu

Abstract

Multi-agent path-finding (MAPF) is important for applications such as the kind of warehousing done by Kiva systems. Solving the problem optimally is NP-hard, yet finding low-cost solutions is important. Bounded-suboptimal MAPF algorithms, such as enhanced conflict-based search (ECBS), often do not perform well in warehousing domains with many agents. We therefore develop bounded-suboptimal MAPF algorithms, called CBS+HWY and ECBS+HWY, that exploit the problem structure of a given MAPF instance by finding paths for the agents that include edges from user-provided highways, which encourages a global behavior of the agents that avoids collisions. On the theoretical side, we develop a simple approach that uses highways for MAPF and provides suboptimality guarantees. On the experimental side, we demonstrate that ECBS+HWY can decrease the runtimes and solution costs of ECBS in Kiva-like domains with many agents if the highways capture the problem structures well.

Introduction

The multi-agent path finding (MAPF) problem is defined as follows: Given a graph and a set of agents with unique start and goal vertices, find collision-free paths for all agents from their respective start vertices to their respective goal vertices. Our objective is to minimize the sum of the time steps required for all agents to reach their respective goal vertices.

MAPF is important for applications such as the kind of warehousing done by Kiva systems, which are semi-automated warehouse management systems with potentially hundreds of autonomous robots (Wurman, D'Andrea, and Mountz 2008). Each robot is capable of lifting and carrying shelving units and bringing them to workers at pick-pack-and-ship stations. A typical Kiva installation is arranged on a grid with storage zones in the center and pick-pack-and-ship stations around the perimeter. See Figure 1 for an illustration.

Solving MAPF problems optimally is NP-hard (Yu and LaValle 2013). Optimal MAPF algorithms, such as Conflict-Based Search (CBS) (Sharon et al. 2015) and M^* (Wagner and Choset 2015), therefore can be used only for low numbers of agents. Yet, finding low-cost solutions is important because the same throughput can then be achieved with fewer agents. It is therefore common to use bounded-suboptimal MAPF approaches for higher numbers of agents. In general, search algorithms are called w-*suboptimal* for a user-provided suboptimality bound w if and only if they return solutions of cost at most $wopt$, where opt is the cost of an optimal solution. Their solution costs typically increase with w while their runtimes typically decrease with w although this is not guaranteed (Wilt and Ruml 2012). w-suboptimal search algorithms are also called bounded-suboptimal search algorithms.

Two bounded-suboptimal MAPF approaches have been proposed in the literature, namely 1) a family of algorithms that extend CBS, such as Weighted-CBS and Enhanced-CBS (ECBS) (Barer et al. 2014); and 2) a family of algorithms that extend M* such as inflated-M*, inflated-rM*, inflated-ODrM*, inflated-EPEM*, and EPERM* (Wagner and Choset 2015). ECBS is faster than all other bounded-suboptimal MAPF algorithms compared in (Barer et al. 2014). Yet, we present experimental results in this paper that it is slow in Kiva-like domains with many agents.

We therefore develop bounded-suboptimal MAPF algorithms, called CBS+HWY and ECBS+HWY, that exploit the problem structure of a given MAPF instance by finding paths for the agents that include edges from user-provided sets of edges (called highways). Highways are used in the context of transportation as a means to provide guidance for vehicles to avoid collisions with vehicles that travel in the opposite direction. Our algorithms use highways in the context of a simple inflation scheme based on the ideas behind experience graphs (Phillips et al. 2012) to derive new heuristic values that encourage path finding to return paths that include the edges of the highways, which encourages a global behavior of the agents that avoids collisions. Highways are a convenient way for users to influence the heuristic values and, since the runtimes of ECBS increase with the number of collisions, have the potential to speed it up. On the theoretical side, we develop a simple approach that uses highways for MAPF and provides suboptimality guarantees. On the experimental side, we demonstrate that ECBS+HWY can decrease the runtimes and solution costs of ECBS in Kiva-like domains with many agents if the highways capture the problem structures well.

The idea of using highways for suboptimal MAPF is not new. (Wang and Botea 2008) proposed to use flow restric-

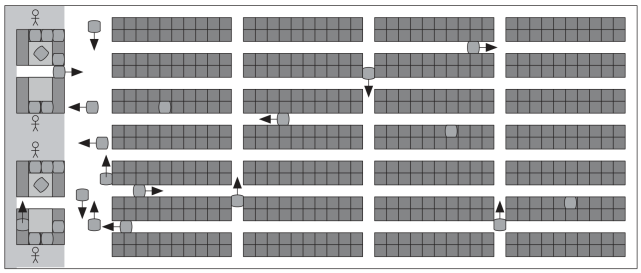

Figure 1: Typical Kiva domain (Wurman, D'Andrea, and Mountz 2008).

tion in road networks, such that movement along cells in the grid is restricted to only one direction. Additional rules are provided to ensure that no feasible path becomes infeasible. Another approach was introduced by (Jansen and Sturtevant 2008) that uses direction maps, which represent joint information about how agents have been moving on the grid, and leads to implicit cooperation during movement. However, non of these approaches can guarantee bounded suboptimality.

Definitions

We define the MAPF problem formally as follows: We are given a graph $G = (V, E)$ and a set of K agents $1, \ldots, K$. Each agent j has a unique start vertex $s^j \in V$ and a unique goal vertex $g^j \in V$. At each time step, each agent can either move to an adjacent vertex or wait at its current vertex, both with cost one. A *solution* to a MAPF instance is a set of *feasible paths*, one path $\{s_0^j, \ldots, s_{T_j}^j, s_{T_j+1}^j, \ldots\}$ for each agent $j \in \{1, \ldots, K\}$, such that no two paths are in *collision*. A path for agent j is *feasible* if and only if 1) it starts at agent j's start vertex, that is, $s_0^j = s^j$; 2) it ends at agent j's goal vertex and remains there, that is, there exists a lowest T_j such that $s_{T_j}^j = g^j$ and, for each $t > T_j$, $s_t^j = g^j$; and 3) every action is a legal move or wait action, that is, for all $t \in \{0, 1, \ldots, T_j - 1\}$, $(s_t^j, s_{t+1}^j) \in E$ or $s_t^j = s_{t+1}^j$. A *collision* between the paths of agents j and k is either a *vertex collision* (j, k, s, t), that is, $s = s_t^j = s_t^k$, or an *edge collision* (j, k, s_1, s_2, t), that is, $s_1 = s_t^j = s_{t+1}^k$ and $s_2 = s_{t+1}^j = s_t^k$. A constraint is either a vertex constraint or an edge constraint. A *vertex constraint* (j, s, t) prohibits agent j from occupying s at time step t. An *edge constraint* (j, s_1, s_2, t) prohibits agent j from moving from s_1 to s_2 at time step t. The cost of agent j's path is the number of time steps T_j until it reaches its goal vertex. Our objective is to minimize the sum $\sum_{j=1}^{K} T_j$ of the path costs of all agents, which is a common objective in the literature (Yu and LaValle 2013; Sharon et al. 2015).

CBS

CBS (Sharon et al. 2015) is an optimal MAPF algorithm. It performs high-level and low-level searches. Each node in the high-level search tree (high-level node) contains a set of constraints and a set of feasible paths (one for each agent) that respect the set of constraints. The high-level root node has no constraints. The high-level search of CBS is a best-first search that uses the costs of the high-level nodes as f-values. The cost of a high-level node is the sum of the path costs of its paths. When CBS expands a high-level node N, it checks whether the node is a goal node. A high-level node is a goal node if and only if none of its paths are in collision. If N is a goal node, then CBS terminates successfully and outputs the paths of the goal node as solution. Otherwise, at least two paths are in collision, and CBS generates two high-level children of N, called N_1 and N_2. Both N_1 and N_2 inherit the constraints of N. If the collision is a vertex collision (j, k, s, t), then CBS adds the constraint (j, s, t) to N_1 and the constraint (k, s, t) to N_2. If the collision is an edge collision (j, k, s_1, s_2, t), then CBS adds the constraint (j, s_1, s_2, t) to N_1 and the constraint (k, s_2, s_1, t) to N_2. When CBS generates a high-level node N, it performs a low-level search for each agent independently. The low-level search for agent j is a (best-first) A* search that ignores all other agents and finds a minimum-cost path from agent j's start vertex to its goal vertex that is both feasible and respects the constraints of N that involve agent j.

We use the following notation for all CBS variants: OPT is the cost of an optimal solution of the MAPF instance. OPT_N^j is the cost of an optimal path for agent j that respects the constraints of high-level node N. COST_N^j is the cost of the path found by the low-level search. Let $\text{OPT}_N = \sum_{j=1}^{K} \text{OPT}_N^j$ and $\text{COST}_N = \sum_{j=1}^{K} \text{COST}_N^j$. LB_N^j is the minimum f-value in the OPEN list of the low-level search for agent j after it terminates when high-level node N is generated. Let $\text{LB}_N = \sum_{j=1}^{K} \text{LB}_N^j$ and $\text{LB} = \min_{N \in \text{OPEN}} \text{LB}_N$ for the OPEN list of the high-level search.

ECBS

ECBS(w) (Barer et al. 2014) is a bounded-suboptimal MAPF algorithm based on CBS that is faster than all other bounded-suboptimal MAPF algorithms compared in (Barer et al. 2014). The high-level and low-level searches of ECBS(w) are focal searches (Pearl and Kim 1982) instead of best-first searches. A best-first search maintains an OPEN list. A focal search also maintains a FOCAL list, that contains a subset of the OPEN list. A best-first search expands the node with the lowest f-value in the OPEN list. A focal search, on the other hand, expands a node in the FOCAL list that is determined by a user-provided tie-breaking criterion. The low-level searches of ECBS(w) for agent j are focal searches with a FOCAL list that contains all low-level nodes $n \in$ OPEN in the OPEN list of the low-level search such that $f(n) \leq w f_{min}$, where $w > 1$ is a user-provided parameter and f_{min} is the lowest f-value of any node in the OPEN list. We refer to a focal search with such a FOCAL list as *regular focal search*. The high-level search of ECBS(w) is a focal search with a FOCAL list that contains all high-level nodes $N \in$ OPEN in the OPEN list of the high-level search such that $\text{COST}_N \leq w\text{LB}$. The low-level searches and the high-level search use their flexibility in deciding which nodes to expand to reduce the number of collisions by using appropriate tie-breaking criteria. ECBS(w) is w-suboptimal if its low-level searches are allowed to re-expand nodes to lower

the f-values of the nodes.

Incorporating Highways

The low-level searches of all CBS variants when high-level node N is generated typically use the *shortest-path heuristic values* $h^j(s)$, that is, the perfect heuristic values for agent j when all constraints of N are ignored, defined formally as

$$h^j(s) = \min_{\pi} \sum_{(s_i,s_{i+1})\in\pi} 1,$$

where $\pi = \{s, \ldots, g^j\}$ is a feasible path that takes agent j from s to its goal vertex. The shortest-path heuristic values can be computed quickly and are more informed than the Manhattan distance heuristic values, which is why CBS variants use them to guide their low-level searches. The low-level searches of Weighted-CBS and M* for agent j are weighted A* searches with $wh^j(s)$ as heuristic values, that is, the heuristic values $h^j(s)$ are inflated uniformly by w. The low-level searches of ECBS(w) for agent j are regular focal searches with a FOCAL list that contains all nodes $n \in$ OPEN in the OPEN list of the low-level search such that $f(n) = g(n) + h^j(n) \leq wf_{min}$, where f_{min} is the lowest f-value of any node in the OPEN list.

Increasing w allows for longer paths, which provides agents with the flexibility to avoid collisions by moving around other agents in domains with few agents. The high-level searches of CBS variants then have to resolve fewer collisions and can terminate earlier, potentially reducing the runtimes of the CBS variants. However, this also makes the agents spread out more, which can in turn result in a higher number of additional collisions in domains with many agents and increase the runtimes of the CBS variants. Thus, larger values of w are not necessarily beneficial.

We therefore develop bounded-suboptimal MAPF algorithms that exploit the problem structure of a given MAPF instance by finding paths for the agents that include edges from user-provided sets of edges (called highways). Highways are used in the context of transportation as a means to provide guidance for vehicles to avoid collisions with vehicles that travel in the opposite direction. Our algorithms inflate heuristic values non-uniformly in a way that encourages path finding to return paths that include the edges of the highways, which encourages a global behavior of the agents that avoids collisions. Highways are a convenient way for users to influence the heuristic values and, since the runtimes of CBS variants increase with the number of collisions, have the potential to speed them up. We use the ideas behind experience graphs (Phillips et al. 2012), which were developed to speed up motion planning, to derive suitable heuristic values.

Formally, a *highway* is a subgraph $G_{hwy} = (V_{hwy}, E_{hwy})$ of the given graph $G = (V, E)$. The low-level searches use the *highway heuristic values* $h^j_{hwy}(s)$ for the user-provided parameter $w > 1$ (that determines the level of encouragement for path finding to return paths that include the edges of the highway), defined formally as

$$h^j_{hwy}(s) = \min_{\pi} \sum_{(s_i,s_{i+1})\in\pi} \begin{cases} 1 & \text{if } (s_i, s_{i+1}) \in E_{hwy} \\ w & \text{otherwise,} \end{cases}$$

where $\pi = \{s, \ldots, g^j\}$ is a feasible path that takes agent j from s to its goal vertex. We can easily adapt the highway heuristic values to graphs with non-uniform edge costs.

Proposition 1. *For all agents j and states s, $h^j(s) \leq h^j_{hwy}(s) \leq wh^j(s)$.*

Heuristic values $h(s)$ are *w-admissible* if and only if $0 \leq h(s) \leq wopt(s)$ for each state s, where $opt(s)$ is the cost of an optimal solution for start vertex s. Thus, the heuristic values $h^j_{hwy}(s)$ are w-admissible according to Proposition 1 since the shortest-path heuristic values are admissible.

Figure 2(a) shows a MAPF instance as an example. The shortest-path heuristic values for agents 1 and 2 are shown in (c) and (d), respectively. The highway heuristic values for agents 1 and 2 are shown in (e) and (f), respectively, for $w = 2$ and the highway shown in (b). For instance, the shortest-path heuristic value for agent 1 and the cell in row 2 and column 3 is 2 (corresponding to moving east and south, each with cost one) while the highway heuristic value is 3 (corresponding to moving east with cost two and then following the highway south with cost one).

Collision-Based Search with Highways

Our first bounded-suboptimal MAPF algorithm, called CBS+HWY(w), is a version of CBS whose low-level searches use the highway heuristic values with parameter w instead of the shortest-path heuristic values.

Theorem 2. *CBS+HWY(w) is w-suboptimal.*

Proof. Let M be any high-level node in the OPEN list of the high-level search of CBS+HWY(w) that contains an optimal solution of the MAPF instance (that is, $\text{OPT}_M = \text{OPT}$). Such an M exists according to Lemma 2 of (Sharon et al. 2015). The low-level searches of CBS+HWY(w) use w-admissible heuristic values according to Lemma 1. The low-level search for agent j when high-level node M is generated thus is guaranteed to find a path with cost at most $w\text{OPT}^j_M$, that is, $\text{COST}^j_M \leq w\text{OPT}^j_M$. The proof is identical to the one that proves the suboptimality guarantee of Weighted-A* (Pohl 1970). Consequently, $\text{COST}_M \leq w\text{OPT}_M = w\text{OPT}$. The high-level search of CBS+HWY(w) is a best-first search and can thus never expand a high-level goal node with a cost of more than wOPT. Thus, CBS+HWY(w) is w-suboptimal. □

	CBS	**CBS+HWY(w)**				**ECBS(w)**			
w	**-**	**1.1**	**1.2**	**1.5**	**2.0**	**1.1**	**1.2**	**1.5**	**2.0**
HL-Exp	7	7	7	3	1	7	3	1	1
HL-Gen	13	13	13	5	1	13	5	1	1
LL-Exp	70	70	70	30	12	70	30	9	9
LL-Gen	277	277	277	115	31	277	115	33	33
SolCost	9	9	9	9	12	9	9	9	9

Table 1: Expanded and generated nodes and solution costs for CBS, CBS+HWY(w), and ECBS(w) for the example in Figure 2.

Figures 2(g) and (h) show the paths found by CBS and CBS+HWY(2.0), respectively, for the example in Figure 2(a). The value of $w = 2.0$ is high enough to

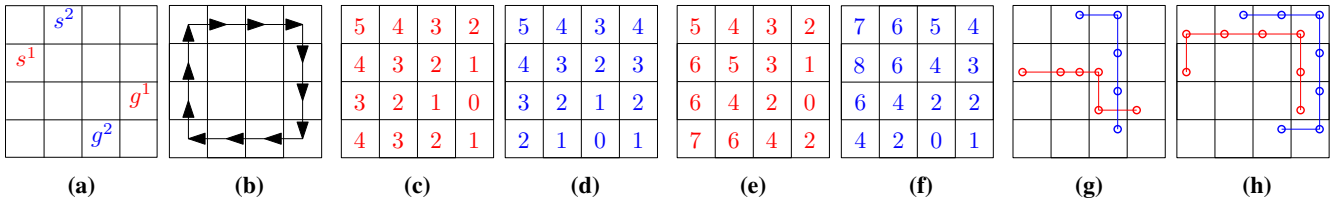

Figure 2: Differences between CBS and CBS+HWY(2.0) in a pathological domain where CBS is extremely inefficient. (a) shows the start and goal vertices for the two agents. (b) shows the highway. (c) and (d) show the shortest-path heuristic values for agents 1 and 2, respectively. (e) and (f) show the highway heuristic values for agents 1 and 2, respectively. (g) shows the paths found by CBS. (h) shows the paths found by CBS+HWY(2.0). Each dot represent a time step. Two dots in the same cell thus represent a wait action.

make both agents travel along the outer ring to their respective goal vertices. We ran CBS and, for each $w \in \{1.1, 1.2, 1.5, 2.0\}$, CBS+HWY($w$) and ECBS($w$) on the example in Figure 2. We ran all experiments on a computer with a 3.2GHz Intel Core i7 CPU and 8GBytes of RAM. Table 1 shows the number of expanded and generated nodes by the high-level and low-level searches (HL-Exp, LL-Exp, HL-Gen, and LL-Gen, respectively) and the solution costs (SolCost). The paths found by the low-level searches of CBS+HWY(2.0) avoid the inevitable collision of the optimal paths of the two agents when there are no constraints, which explains why CBS+HWY(2.0) expands fewer nodes than CBS although it has a higher solution cost. However, comparing a bounded-suboptimal MAPF algorithm, like CBS+HWY(2.0), with an optimal one, like CBS, is like comparing apples and oranges. Comparing two bounded-suboptimal MAPF algorithms with the same suboptimality bound, like CBS+HWY(2.0) and ECBS(2.0), is fairer. Unfortunately, ECBS(2.0) expands fewer nodes than CBS+HWY(2.0) and finds a solution of lower cost. However, the example is too simple. The number of agents is low, and a single wait action by one of the two agents prevents the collision of their paths.

	CBS+HWY(w)		**ECBS(w)**	
w	**1.5**	**2.0**	**1.5**	**2.0**
HL-Exp	8071	27	33	67
HL-Gen	16,119	47	37	88
LL-Exp	992,260	3,754	10,979	63,486
LL-Gen	2,623,514	10,212	19,567	87,229
Runtime	51,293	266	468	3,073
SolCost	440	423	460	521

Table 2: Expanded and generated nodes, runtimes (in milliseconds) and solution costs for CBS+HWY(w) and ECBS(w) for the example in Figure 3.

We therefore also ran, for each $w \in \{1.1, 1.2, 1.5, 2.0\}$, CBS+HWY($w$) and ECBS($w$) on the example in Figure 3, which is a more complex MAPF instance with a problem structure that highways can exploit. Blocked cells in the grid of size 5×20 are shown in black. There is open space of size 5×5 on both sides, connected by a corridor of width two. 10 agents have their start vertices in the left open space and their goal vertices in the right open space, and 10 agents have their start vertices in the right open space and their goal vertices in the left open space. The arrows show the highway. Table 2 shows the number of expanded and generated nodes by the high-level and low-level searches, the runtimes, and the solution costs. CBS+HWY(1.1), CBS+HWY(1.2), ECBS(1.1), and ECBS(1.2) fail to find any solutions within the five-minute runtime limit and are thus omitted from the table. ECBS(2.0) can find longer paths than ECBS(1.5) which provides agents with the flexibility to avoid collisions by moving around other agents, resulting in them spreading out more, which in turn results in a higher number of additional collisions since many agents traverse the narrow corridor in both directions. This property could explain why ECBS(2.0) expands more nodes than ECBS(1.5), has a higher runtime and finds a solution of higher cost. On the other hand, the highway directs agents moving from east to west to stay north in the corridor and agents moving from west to east to stay south in the corridor, which avoids many collisions. This property could explain why CBS+HWY(w) finds a solution of lower cost than ECBS(w) for both $w = 1.5$ and $w = 2.0$. It has a higher runtime for $w = 1.5$ but a lower runtime for $w = 2.0$.

Enhanced Collision-Based Search with Highways

The example in Figure 2 demonstrates the advantage of ECBS(w), whose strength seems to be in reducing vertex collisions. On the other hand, the example in Figure 3 demonstrates the advantage of using highways in the context of CBS+HW(w), whose strength seems to be in reducing edge collisions. Both MAPF algorithms provide suboptimality guarantees. It thus makes sense to try to combine them to develop a bounded-suboptimal MAPF algorithm whose strength is in reducing both vertex and edge collisions. Our second bounded-suboptimal MAPF algorithm, called ECBS(w_1)+HWY(w_2), is therefore a version of ECBS(w_1) whose low-level searches use the highway heuristic values with parameter w_2 instead of the shortest-path heuristic values. The following lemma asserts that the low-level searches of ECBS(w_1)+HWY(w_2) are w_1w_2-suboptimal and that $\text{LB}^j_N \leq w_2\text{OPT}^j_N$ for all high-level nodes N and agents j.

Lemma 3. *A regular focal search with parameter $w_1 > 1$ that uses the f-values $f(n) = g(n) + w_2h(n)$ for admissible heuristic values $h(n)$ and parameter $w_2 > 1$ is w_1w_2-suboptimal. Furthermore, the minimum f-value in its OPEN list is always at most w_2 times the cost of an optimal solution.*

Proof. We consider two focal searches: First, we consider regular focal search Search1 with parameter w_1 that uses the

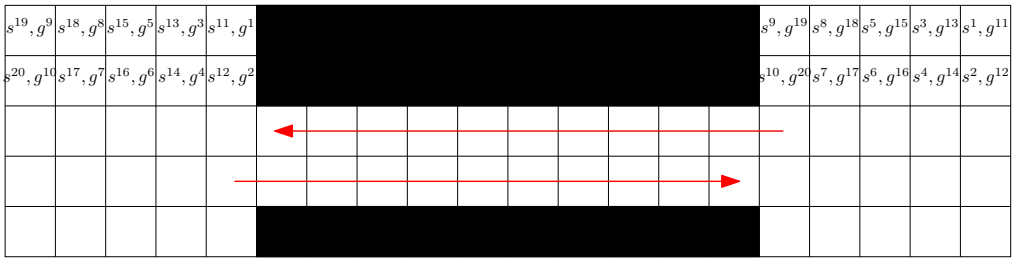

Figure 3: Corridor example on which we compare CBS+HWY(w) and ECBS(w).

f-values $f(n) = g(n) + w_2h(n)$, as used in the lemma. Its FOCAL list thus is

$$\text{FOCAL}^1 = \{n \in \text{OPEN} : g(n) + w_2h(n) \leq w_1(g(n^1_{min}) + w_2h(n^1_{min}))\},$$

where $n^1_{min} = \arg\min_{n \in \text{OPEN}}(g(n) + w_2h(n))$. Second, we consider regular focal search Search2 with parameter w_1w_2 that uses the f-values $f(n) = g(n) + h(n)$. Its FOCAL list thus is

$$\text{FOCAL}^2 = \{n \in \text{OPEN} : g(n) + h(n) \leq w_1w_2(g(n^2_{min}) + h(n^2_{min}))\},$$

where $n^2_{min} = \arg\min_{n \in \text{OPEN}}(g(n) + h(n))$. Both focal searches are allowed to re-expand nodes to lower the f-values of the nodes.

We show by induction that the OPEN lists of both searches can always be the same. Initially, both OPEN lists contain only the start node and thus are the same. Assume that they are the same at some point in time. Then, for each node $n \in \text{FOCAL}^1$, it holds that

$$\begin{aligned} g(n) + h(n) &\leq g(n) + w_2h(n) \\ &\leq w_1(g(n^1_{min}) + w_2h(n^1_{min})) \\ &\leq w_1(g(n^2_{min}) + w_2h(n^2_{min})) \\ &\leq w_1(w_2g(n^2_{min}) + w_2h(n^2_{min})) \\ &\leq w_1w_2(g(n^2_{min}) + h(n^2_{min})). \end{aligned}$$

Therefore, $n \in \text{FOCAL}^2$ and, consequently, $\text{FOCAL}^1 \subseteq \text{FOCAL}^2$. Thus, if Search1 expands n, Search2 can be forced to expand n as well and their OPEN lists are then again the same. Thus, they eventually find the same solution. It is known that Search 2 is w_1w_2-suboptimal. Thus, Search1 is w_1w_2-suboptimal as well. Furthermore, it is known that the minimum f-value in the OPEN list of Search2 is always at most the cost of the optimal solution *opt*. Then, since the OPEN lists of both searches are always the same, it always holds that

$$\begin{aligned} \min_{n \in \text{OPEN}}(g(n) + h(n)) &\leq opt \\ w_2 \min_{n \in \text{OPEN}}(g(n) + h(n)) &\leq w_2opt \\ \min_{n \in \text{OPEN}}(w_2g(n) + w_2h(n)) &\leq w_2opt \\ \min_{n \in \text{OPEN}}(g(n) + w_2h(n)) &\leq w_2opt. \end{aligned}$$

Thus, the minimum f-value in the OPEN list of Search1 is always at most w_2 times the cost of the optimal solution. □

Theorem 4. *ECBS(w_1)+HWY(w_2) is w_1w_2-suboptimal.*

Proof. Let M be any high-level node in the OPEN list of the high-level search of ECBS(w_1)+HWY(w_2) that contains an optimal solution of the MAPF instance (that is, $\text{OPT}_M = \text{OPT}$). Such an M exists analogously to Lemma 2 of (Sharon et al. 2015). Then,

$$\begin{aligned} \text{LB} \leq \text{LB}_M &= \sum_{j=1}^{K} \text{LB}^j_M \\ &\leq \sum_{j=1}^{K} w_2\text{OPT}^j_M \\ &= w_2\text{OPT}_M \\ &= w_2\text{OPT} \end{aligned}$$

since $\text{LB}^j_M \leq w_2\text{OPT}^j_M$ according to Lemma 3. The high-level search of ECBS(w_1)+HWY(w_2) is a focal search with a FOCAL list that contains all nodes $N \in$ OPEN such that $\text{COST}_N \leq w_1\text{LB}$. Thus, it always expands a node whose cost is at most $w_1\text{LB} \leq w_1w_2\text{OPT}$. Consequently, the high-level search can never expand a high-level goal node with a cost of more than $w_1w_2\text{OPT}$, and ECBS(w_1)+HWY(w_2) is w_1w_2-suboptimal. □

Experimental Results

We evaluated ECBS(w) and ECBS(w_1)+HWY(w_2) in a Kiva-like domain, see Figure 4. Blocked cells in the grid of size 22×54 are shown in black, and all corridors between them are of width one. There are open spaces of size 22×5 on both sides, called Area1 and Area2. We randomly generated 10 instances with 150 agents. We simulated robots carrying shelving units from one side of the warehouse to a pick-pack-and-ship station on the other side of the warehouse. 75 agents thus have randomly chosen start vertices from Area1 and randomly chosen goal vertices from Area2, while the other 75 agents have randomly chosen start vertices from Area2 and randomly chosen goal vertices from Area1.

We ran, for each $w \in \{1.1, 1.2, 1.5, 2.0\}$, ECBS($w$) on each of the 10 instances. Table 3 shows the runtimes and solution costs. ECBS(1.5) solves 4 of the 10 instances. ECBS(1.1), ECBS(1.2), and ECBS(2.0) fail to find any solutions within the five-minute runtime limit and are thus omitted from the table, showing that higher values of w are not necessarily beneficial, as argued earlier.

Instance	ECBS(1.5)		ECBS(w_1)+HWY(2.0)					
			$w_1 = 1.1$		$w_1 = 1.2$		$w_1 = 1.5$	
	Runtime	SolCost	Runtime	SolCost	Runtime	SolCost	Runtime	SolCost
1	272,440	10,258			103,600	9,625	223,159	10,588
2	267,807	10,530	191,211	9,660	183,379	9,736	260,522	10,603
3					204,533	10,041		
4					179,214	9,892	268,431	10,577
5	253,564	10,246	209,197	9,619	146,298	9,880	294,717	10,396
6			210,227	9,494			261,957	10,272
7			206,498	9,476	136,049	9,834		
8			291,254	9,449	83,679	9,590	277,931	10,313
9	261,067	10,310			118,998	9,865	239,336	10,639
10					201,038	10,085		

Table 3: Runtimes (in milliseconds) and solution costs for ECBS(1.5) and ECBS(w_1)+HWY(2.0) for the example in Figure 4. Cells are empty if an algorithm did not terminate within a five-minute runtime limit.

We ran, for each $w_1 \in \{1.1, 1.2, 1.5, 2.0\}$, ECBS($w_1$)+HWY(2.0) on each of the 10 instances. The arrows in Figure 4 show the highway. Again, Table 3 shows the runtimes and solution costs. ECBS(2.0)+HWY(2.0) fails to find any solutions within the five-minute runtime limit and is thus omitted from the table, showing again that higher values of w_1 are not necessarily beneficial. ECBS(w_1)+HWY(w_2) often has lower runtimes or solution costs or solves more instances than ECBS(w), which is encouraging despite being anecdotal.

We experimented with different highway layouts and parameters w_1 and w_2 for ECBS(w_1)+HWY(w_2) but no combination dominates all others. However, these parameters are clearly important factors for the performance of ECBS(w_1)+HWY(w_2): First, we ran, for each $w_2 \in \{1.2, 1.5, 2.0, 3.0\}$, ECBS(1.5)+HWY($w_2$) on each of the 10 instances of the example in Figure 4 after reducing the highway to the outer ring (that is, the top-most, right-most, bottom-most and left-most arrows). The level of encouragement for path finding to return paths that include the edges of the highways and thus the solution costs increase with w_2 because the agents then tend to use the highway to circumnavigate the center rather than cut through it. Second, if the highways do not capture the problem structures well and thus do not help to reduce collisions among the paths, then ECBS(w_1)+HWY(w_2) not only does not improve over ECBS(w) but can have higher runtimes or solution costs or solve fewer instances.

Instance	ECBS(1.5)+HWY_ring(w)							
	w=1.2		w=1.5		w=2		w=3	
	RunTime	SolCost	RunTime	SolCost	RunTime	SolCost	RunTime	SolCost
1	253,923	10,653			177,171	11,059	276,075	11,354
2	197,154	11,067	258,463	11,098			240,897	11,707
3			244,781	10,856	175,048	11,161	271,442	11,414
4					241,583	11,631	172,725	11,319
5			265,795	11,239	186,265	11,152	200,102	11,363
6	266,169	10,840	294,468	11,308	247,199	11,133		
7								
8	252,721	10,595	251,333	11,150				
9					202,411	11,447	294,624	11,245
10			269,460	11,115				

Table 4: Runtimes (in milliseconds) and solution costs for ECBS(1.5)+HWY(w_2) for the example in Figure 4, where the highway consists of the outer ring only. Cells are empty if an algorithm did not terminate within a five-minute runtime limit.

We also ran CBS+HWY(w) but it fails to terminate within the five-minute runtime limit on all Kiva-like instances regardless of the highway layout. While the highways provide good guidance to move agents in the corridors, CBS+HWY(w) still has to find collision-free paths for

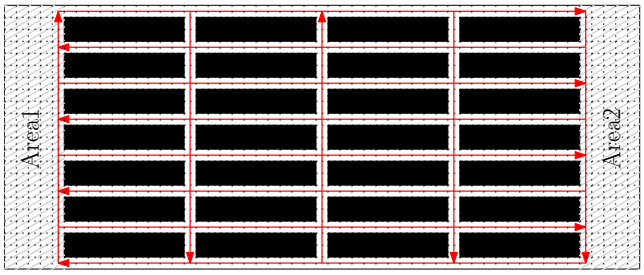

Figure 4: Kiva-like domain on which we compare ECBS(w) and ECBS(w_1)+HWY(2.0).

150 agents inside Area1 and Area2. In those areas, CBS has less flexibility than ECBS(w) to avoid collisions by moving agents around other agents, which could explain why it fails to find solutions within the runtime limit.

Conclusions

We presented a new bounded-suboptimal MAPF approach that takes advantage of additional inputs that represent a highway and a parameter w. It uses the highway to derive new w-admissible heuristic values that encourage path finding to return paths that include the edges of the highway. The level of encouragement increases with w. Our new bounded-suboptimal variants of CBS and ECBS(w), called CBS+HWY(w) and ECBS(w_1)+HWY(w_2), encourage a global behavior of the agents that avoids collisions. On the theoretical side, we developed a simple approach that uses highways for MAPF and provides suboptimality guarantees. On the experimental side, we demonstrated that ECBS(w_1)+HWY(w_2) can decrease the runtimes and solution costs of ECBS(w) in Kiva-like domains with many agents if the highway captures the problem structure well.

In future work, we plan to develop approaches that determine good highways automatically, investigate whether inflating the edge costs of the given graph (by increasing the costs of highway edges to w) in addition to inflating the heuristic values provides additional benefits, figure out whether penalizing movement costs against highway edges (similar to direction maps (Jansen and Sturtevant 2008)) helps to improve the performance of our MAPF approaches while continuing to provide suboptimality guarantees, extend ECBS(w_1)+HWY(w_2) to split the user-provided suboptimality bound w dynamically between w_1 and w_2 (similar to how ECBS(w) splits the suboptimality bound w dynamically between the high-level and low-level searches), and explore highways in the context of other MAPF algorithms, such as M* and inflated M*.

Acknowledgments

We thank Maxim Likhachev and Michael Phillips for helpful discussions. We also thank Ariel Felner, Guni Sharon, and Maxim Barer for their CBS and ECBS(w) source code, which we used in our experiments. Our research was supported by NSF under grant numbers 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 7th Annual Symposium on Combinatorial Search (SOCS).*

Jansen, M., and Sturtevant, N. 2008. Direction maps for cooperative pathfinding. In *Proceedings of the 4th Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, 185–190.

Pearl, J., and Kim, J. H. 1982. Studies in semi-admissible heuristics. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 4:392 –399.

Phillips, M.; Cohen, B. J.; Chitta, S.; and Likhachev, M. 2012. E-graphs: Bootstrapping planning with experience graphs. In *Proceedings of the 8th Robotics: Science and Systems Conference (RSS)*.

Pohl, I. 1970. Heuristic search viewed as path finding in a graph. *Artificial Intelligence* 1(3):193–204.

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

Wagner, G., and Choset, H. 2015. Subdimensional expansion for multirobot path planning. *Artificial Intelligence* 219:1–24.

Wang, K.-H. C., and Botea, A. 2008. Fast and Memory-Efficient Multi-Agent Pathfinding. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS)*, 380–387.

Wilt, C. M., and Ruml, W. 2012. When does weighted A* fail? In *Proceedings of the 5th Annual Symposium on Combinatorial Search (SOCS)*.

Wurman, P. R.; D'Andrea, R.; and Mountz, M. 2008. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine* 29(1):9–20.

Yu, J., and LaValle, S. M. 2013. Structure and intractability of optimal multi-robot path planning on graphs. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*.