Using Highways for Bounded-Suboptimal Multi-Agent Path Finding

L. Cohen, T. Uras, S. Koenig

Nitzan Madar

Seminar in Robotics - 236824

May 20, 2020

TECHNION
Israel Institute of Technology

Table Of Contents

1 Inroduction

2 Incorporating Highways

3 Collision-Based Search with Highways (CBS+HWY)

4 **Enhanced** Collision-Based Search with Highways (ECBS+HWY)

5 Conclusions

6 Future Work

7 Summary

8 References

Inroduction

Optimal Vs. Bounded-Suboptimal Approach

- Multi-agent path finding (MAPF) problem
- **Optimal Approach:**
  - Conflict-Based Search (CBS) [Sharon et al, 2015]
  - M* [Wagner and Choset, 2015]
- Solving the MAPF problem optimally is "NP-hard"
- In real world, optimal approach can be used only for low numbers of agents
- **Bounded-Suboptimal Approach:** (w-suboptimal):
  - CBS family [Barer et al, 2014]
    - Weighted-CBS
    - Enhanced-CBS (ECBS) - the fastest bounded-suboptimal algorithm
  - M* family [Wagner and Choset, 2015]
    - inflated-M*, inflated-rM*
    - inflated-ODrM*, inflated-EPEM*, and EPERM*

Optimal Vs. Bounded-Suboptimal Approach

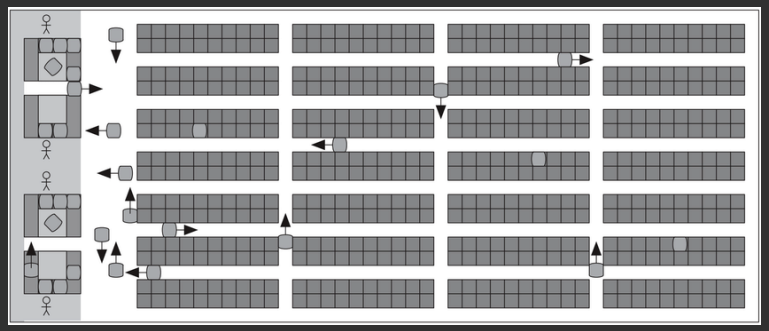
- Multi-agent path finding (MAPF) problem
- **Optimal Approach:**
  - Conflict-Based Search (CBS) [Sharon et al, 2015]
  - M* [Wagner and Choset, 2015]
- Solving the MAPF problem optimally is "NP-hard"
- In real world, optimal approach can be used only for low numbers of agents
- **Bounded-Suboptimal Approach:** (w-suboptimal):
  - CBS family [Barer et al, 2014]
    - Weighted-CBS
    - Enhanced-CBS (ECBS) - the fastest bounded-suboptimal algorithm
  - M* family [Wagner and Choset, 2015]
    - inflated-M*, inflated-rM*
    - inflated-ODrM*, inflated-EPEM*, and EPERM*

Optimal Vs. Bounded-Suboptimal Approach

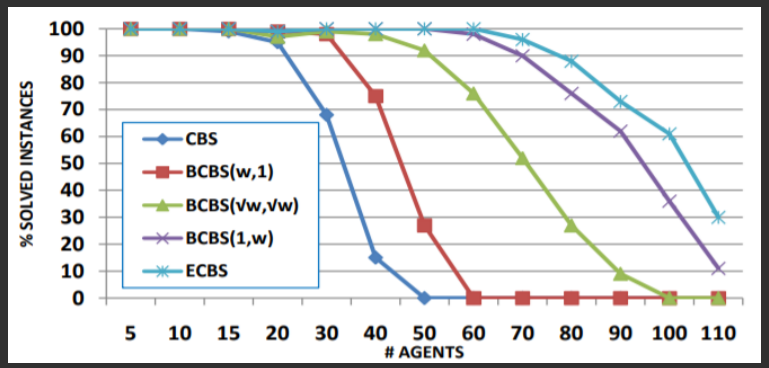
- Multi-agent path finding (MAPF) problem
- **Optimal Approach:**
  - Conflict-Based Search (CBS) [Sharon et al, 2015]
  - M* [Wagner and Choset, 2015]
- Solving the MAPF problem optimally is "NP-hard"
- In real world, optimal approach can be used only for low numbers of agents
- **Bounded-Suboptimal Approach:** (w-suboptimal):
  - CBS family [Barer et al, 2014]
    - Weighted-CBS
    - Enhanced-CBS (ECBS) - the fastest bounded-suboptimal algorithm
  - M* family [Wagner and Choset, 2015]
    - inflated-M*, inflated-rM*
    - inflated-ODrM*, inflated-EPEM*, and EPERM*

Optimal Vs. Bounded-Suboptimal Approach

- In many application domains the runtime of bounded-suboptimal algorithms degrades severely (high agents density for example)



Typical Kiva domain [Wurman, DAndrea & Mountz, 2008]



ECBS runtime example: : $32 \times 32 - 20\%$ obstacles. $w = 1.1$ [Barer et al., 2014]

Brief: Enhanced CBS (ECBS(w)) [Barer et al, 2014]

- CBS-based bounded-suboptimal MAPF solver
- Faster than all other bounded-suboptimal MAPF algorithms (2016)
- w-suboptimal $\rightarrow$ the solution cost $\leq w \cdot Cost_{Optimal}$,
  w is the same from the focal search (will be explained later)
- Like CBS, use high-level (search in constraints tree) and low-level (single agent path finding)
- The high-level and low-level searches are focal search [Pearl and Kim, 1982] instead of best-first searches

Brief: Enhanced CBS (ECBS(w)) [Barer et al, 2014]

- CBS-based bounded-suboptimal MAPF solver
- Faster than all other bounded-suboptimal MAPF algorithms (2016)
- w-suboptimal $\rightarrow$ the solution cost $\leq w \cdot Cost_{Optimal}$,
  w is the same from the focal search (will be explained later)
- Like CBS, use high-level (search in constraints tree) and low-level (single agent path finding)
- The high-level and low-level searches are focal search [Pearl and Kim, 1982] instead of best-first searches

Brief: Enhanced CBS (ECBS(w)) [Barer et al, 2014]

- CBS-based bounded-suboptimal MAPF solver
- Faster than all other bounded-suboptimal MAPF algorithms (2016)
- w-suboptimal $\rightarrow$ the solution cost $\leq w \cdot Cost_{Optimal}$,
  w is the same from the focal search (will be explained later)
- Like CBS, use high-level (search in constraints tree) and low-level (single agent path finding)
- The high-level and low-level searches are focal search [Pearl and Kim, 1982] instead of best-first searches

Brief: Enhanced CBS (ECBS(w)) [Barer et al, 2014]

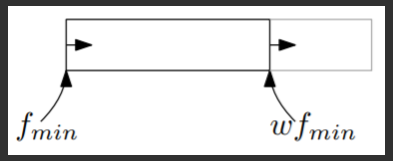
- CBS-based bounded-suboptimal MAPF solver
- Faster than all other bounded-suboptimal MAPF algorithms (2016)
- w-suboptimal $\rightarrow$ the solution cost $\leq w \cdot Cost_{Optimal}$,
  w is the same from the focal search (will be explained later)
- Like CBS, use high-level (search in constraints tree) and low-level (single agent path finding)
- The high-level and low-level searches are focal search [Pearl and Kim, 1982] instead of best-first searches

Brief: Enhanced CBS (ECBS(w)) [Barer et al, 2014]

- CBS-based bounded-suboptimal MAPF solver
- Faster than all other bounded-suboptimal MAPF algorithms (2016)
- w-suboptimal $\rightarrow$ the solution cost $\leq w \cdot Cost_{Optimal}$,
  w is the same from the focal search (will be explained later)
- Like CBS, use high-level (search in constraints tree) and low-level (single agent path finding)
- The high-level and low-level searches are focal search [Pearl and Kim, 1982] instead of best-first searches

Brief: ECBS(w) - FOCAL search

- $FOCAL$ list contains all low-level nodes $n \in OPEN$ in the $OPEN$ list of the low-level search such that $f(n) \leq w \cdot f_{min}$



FOCAL and OPEN lists (figure from *"Anytime Focal Search with Application"*, Cohen et al. 2018)

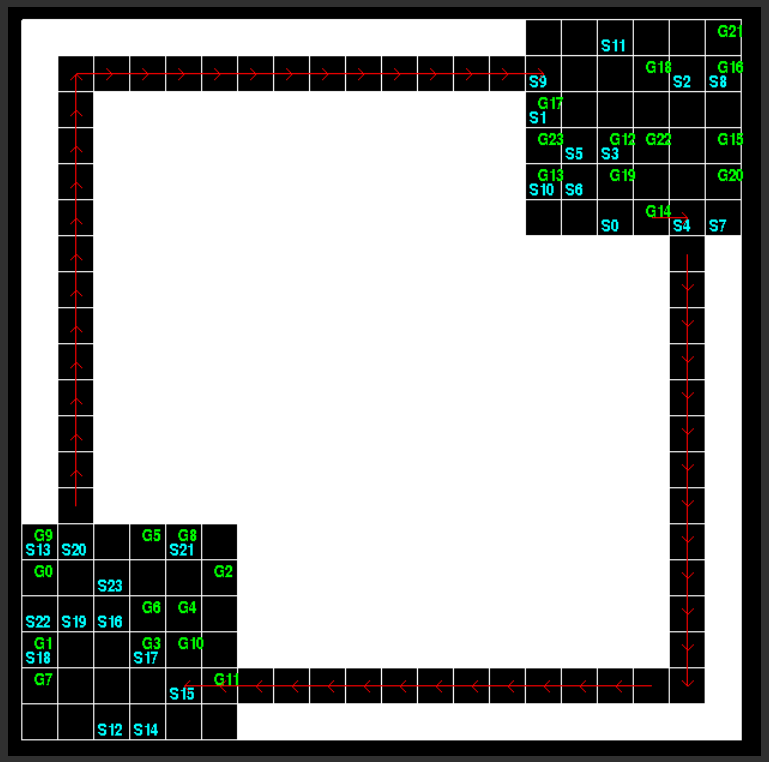
- A focal search expands a node in the $FOCAL$ list that is determined by a user-provided tie-breaking criterion, ECBS use measures related to the number of collisions as secondary heuristic values.

Low-level (for Agent i in High-level node N)		High-level	
Sort $n \in$ OPEN$^i(N)$	**Sort $n \in$ FOCAL$^i(N)$**	**Sort $N \in$ OPEN**	**Sort $N \in$ FOCAL**
1. $g^i(n) + h^i_{sp}(n)$ 2. $g^i(n)$	1. $h^i_{c(N)}(n)$ 2. $g^i(n) + h^i_{sp}(n)$ 3. $g^i(n)$	1. $\sum_i \min_{n \in \text{OPEN}^i} (g^i(n) + h^i_{sp}(n))$	1. $h_c(N)$ 2. $g(N)$

Sort criteria in high and low levels (figure from *"Improved Solvers for Bounded-Suboptimal Multi-Agent Path Finding"*, Cohen et al. 2016)

Using Highways

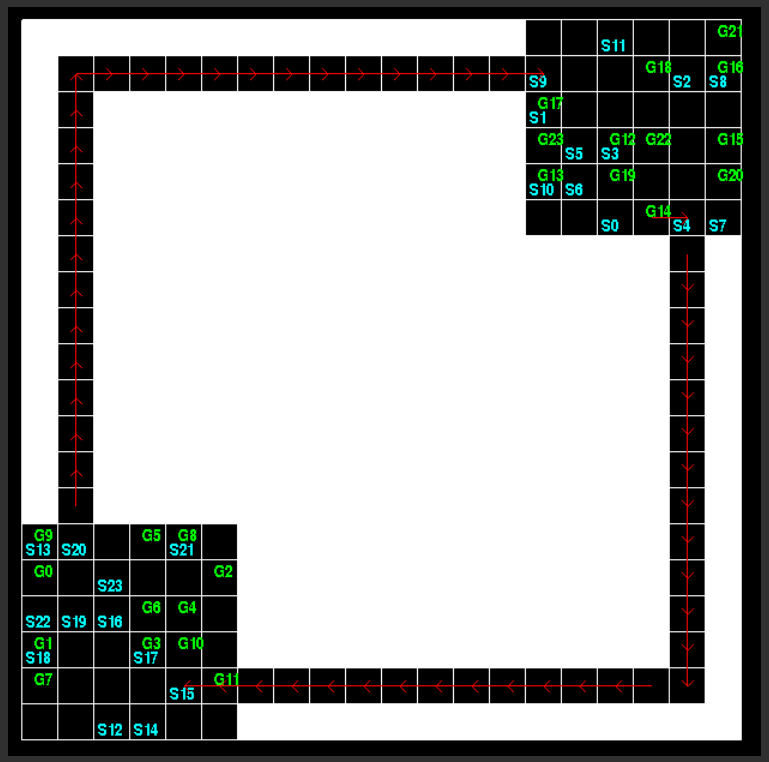
- Problem structure exploitation by finding paths for the agents that include edges from user-provided sets of edges (called highways)
- Highways are a convenient way for users to influence the heuristic values (by marking "better" moves)
- Since the runtime increase with the number of collisions, have the potential to speed it up



Highways example [Cohen et al., 2016]

Using Highways

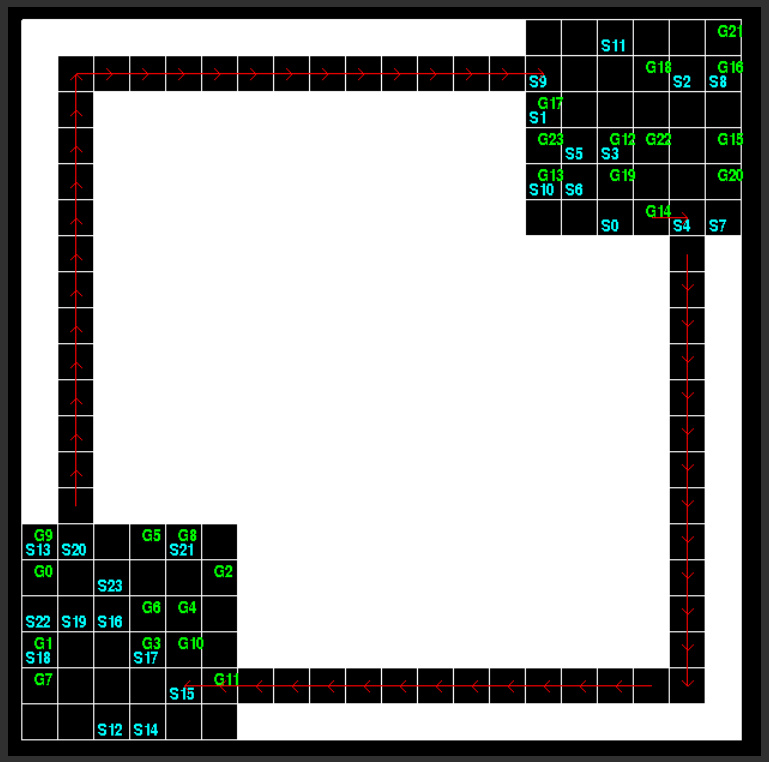
- Problem structure exploitation by finding paths for the agents that include edges from user-provided sets of edges (called highways)
- Highways are a convenient way for users to influence the heuristic values (by marking "better" moves)
- Since the runtime increase with the number of collisions, have the potential to speed it up



Highways example [Cohen et al., 2016]

Using Highways

- Problem structure exploitation by finding paths for the agents that include edges from user-provided sets of edges (called highways)
- Highways are a convenient way for users to influence the heuristic values (by marking "better" moves)
- Since the runtime increase with the number of collisions, have the potential to speed it up



Highways example [Cohen et al., 2016]

Incorporating Highways

Heuristics: Shortest Path Vs. Highways

Shortest Path Heuristic:

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

- $\pi = \{s, \ldots, g^j\}$ = feasible path that takes agent j from s to its goal vertex
- The low-level searches of all CBS variants typically use the shortest-path heuristic values $h^j(s)$

Highways Heuristic:

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

- Formally, a highway is a subgraph $G_{hwy} = (V_{hwy}, E_{hwy})$ of the given graph $G = (V, E)$, preferable edges.
- $w > 1$, non-E_{hwy} edges more expensive. Increasing w allows for longer paths, which provides agents to avoid collisions by moving around other agents.

Heuristics: Shortest Path Vs. Highways

Shortest Path Heuristic:

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

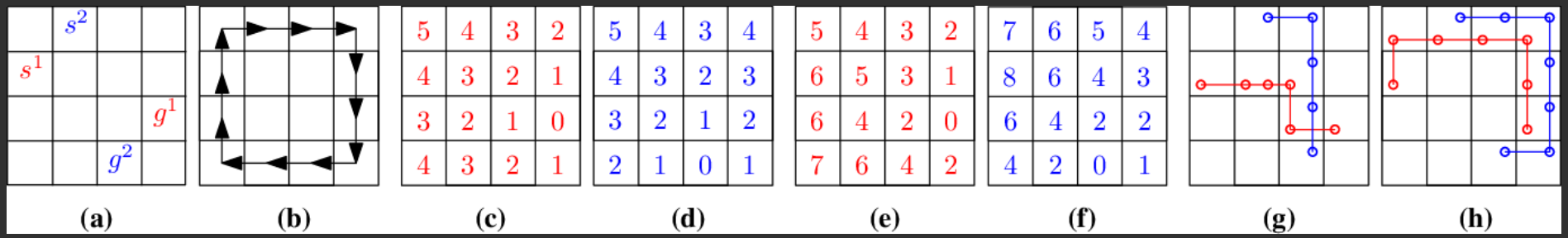
- $\pi = \{s, \ldots, g^j\}$ = feasible path that takes agent j from s to its goal vertex
- The low-level searches of all CBS variants typically use the shortest-path heuristic values $h^j(s)$

Highways Heuristic:

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

- Formally, a highway is a subgraph $G_{hwy} = (V_{hwy}, E_{hwy})$ of the given graph $G = (V, E)$, preferable edges.
- $w > 1$, non-E_{hwy} edges more expensive. Increasing w allows for longer paths, which provides agents to avoid collisions by moving around other agents.

Highways Example



Differences between CBS and CBS+HWY(2.0) in a pathological domain where CBS is extremely inefficient. Each dot represent a time step, two dots in the same cell thus represent a wait action.

- **(a):** shows the start and goal vertices for the two agents.
- **(b):** shows the highway.
- **(c) and (d):** shows the shortest-path heuristic values for agents 1 and 2, respectively.
- **(e) and (f):** shows 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).

Collision-Based Search with Highways (CBS+HWY)

CBS+HWY(w) - Theorem

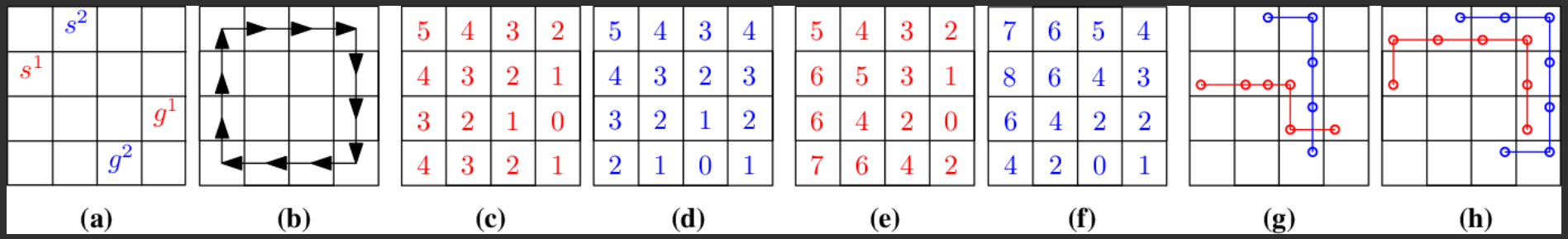
Lemma: $h^j(s) \leq h^j_{hwy} \leq w \cdot h^j(s)$

- HWY heuristic is w-admissible
  - Heuristic values $h(s)$ are w-admissible $\iff 0 \leq h(s) \leq w \cdot opt(s)$ for each state s

Theorem: CBS+HWY(w) is w-suboptimal

- Highway parameter affect the solution cost
- Different value $\rightarrow$ Different solution

Expanded and generated nodes and solution costs for CBS, CBS+HWY(w),and ECBS(w)



Differences between CBS and CBS+HWY(2.0) in a pathological domain where CBS is extremely inefficient.

	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

Expanded and generated nodes and solution costs for CBS, CBS+HWY(w),and ECBS(w). HL=high level, LL=Low level

Results

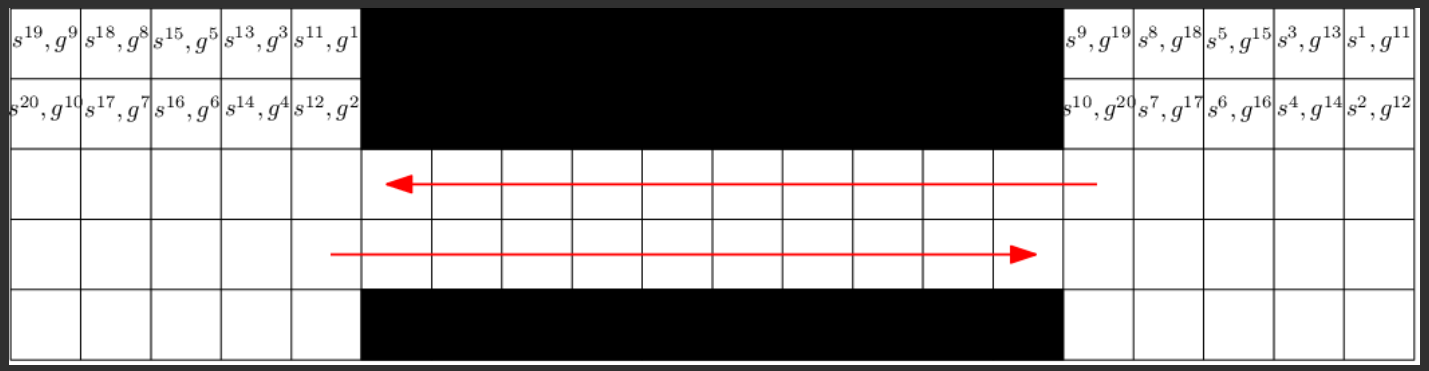
- $w = 2$ is high enough to make both agents travel along the outer ring (the highway) and avoid the inevitable collision of the optimal paths of the two agents when there are no constraints, but with higher solution cost

- ECBS(2.0) expands fewer nodes than CBS+HWY(2.0) and finds a solution of lower cost. However this example is too simple

	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

Expanded and generated nodes and solution costs for CBS, CBS+HWY(w),and ECBS(w). HL=high level, LL=Low level

Corridors



Corridor example on which compare CBS(w), CBS+HWY(w) and ECBS(w)

	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

Expanded and generated nodes, runtimes (in milliseconds) and solution costs for CBS+HWY(w) and ECBS(w)

Corridors - Results

- CBS+HWY(1.1), CBS+HWY(1.2), ECBS(1.1), and ECBS(1.2) fail to find any solutions within the five-minute
- ECBS:
  - 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, which in turn results in a higher number of additional collisions
- HWY:
  - The highways directs agents moving the corridor in the highways $\rightarrow$ avoids many collisions.
  - This property could explain why CBS+HWY(w) finds a solution of lower cost than ECBS(w) for both w
  - $w = 1.5$ not "strong" enough $\rightarrow$ higher running time in ECBS+HWY

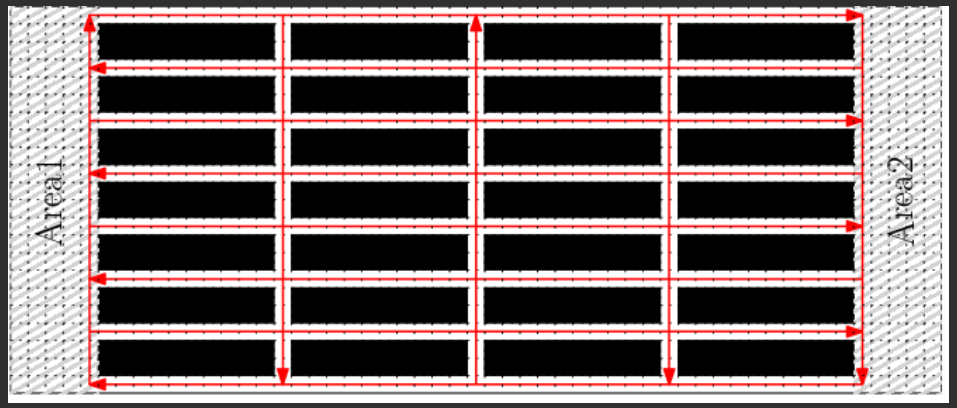
Enhanced Collision-Based Search with Highways (ECBS+HWY)

Theorems

ECBS(w_1)+HWY(w_2) is ($w_1 \cdot w_2$)-suboptimal

- w_1 - parameter of ECBS focal search
- w_2 - parameter of highways heuristic (cost of using edges that not on the highways)
- The solution $\leq (w_1 \cdot w_2) Optimal$

Kiva-Like Domain Tests



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

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		

Runtimes (in milliseconds) and solution costs for ECBS(1.5) and ECBS(w_1)+HWY(2.0)

Kiva-Like Domain Tests - Results

- ECBS(1.1), ECBS(1.2), ECBS(2.0) and ECBS(2.0)+HWY(2.0) fail to find any solutions within the five-minute runtime limit
- HWY constant fixed to $w_2 = 2$, and ran for each $w_1 \in \{1.1, 1.2, 1.5, 2.0\}$. Best Value is $w_1 = 1.2$ ("optimization")
- Success rate of ECBS+HWY is better for any parameter choosing in this example
- Highways choosing: if we reduce the highway to the outer ring only the success rate reduce dramatically

Conclusions

Conclusions and Insights

- CBS+HWY(w) is w-suboptimal and ECBS(w_1)+HWY(w_2) is ($w_1 \cdot w_2$)-suboptimal algorithms for MAPF
- Choosing w in CBS+HWY(w) or w_1, w_2 in ECBS(w_1)+HWY(w_2) is important:
  - Increasing w_1: allows longer paths, which provides agents with the flexibility to avoid collisions by moving around other agents
  - Increasing w_2: guides the agents through the highways
- Choosing the highways according to the map affect the runtime and solution cost dramatically
- This approach can be effective to real worlds problems (like Kiva's domains)

Conclusions and Insights

- CBS+HWY(w) is w-suboptimal and ECBS(w_1)+HWY(w_2) is ($w_1 \cdot w_2$)-suboptimal algorithms for MAPF
- Choosing w in CBS+HWY(w) or w_1, w_2 in ECBS(w_1)+HWY(w_2) is important:
  - Increasing w_1: allows longer paths, which provides agents with the flexibility to avoid collisions by moving around other agents
  - Increasing w_2: guides the agents through the highways
- Choosing the highways according to the map affect the runtime and solution cost dramatically
- This approach can be effective to real worlds problems (like Kiva's domains)

Conclusions and Insights

- CBS+HWY(w) is w-suboptimal and ECBS(w_1)+HWY(w_2) is ($w_1 \cdot w_2$)-suboptimal algorithms for MAPF
- Choosing w in CBS+HWY(w) or w_1, w_2 in ECBS(w_1)+HWY(w_2) is important:
  - Increasing w_1: allows longer paths, which provides agents with the flexibility to avoid collisions by moving around other agents
  - Increasing w_2: guides the agents through the highways
- Choosing the highways according to the map affect the runtime and solution cost dramatically
- This approach can be effective to real worlds problems (like Kiva's domains)

Conclusions and Insights

- CBS+HWY(w) is w-suboptimal and ECBS(w_1)+HWY(w_2) is ($w_1 \cdot w_2$)-suboptimal algorithms for MAPF
- Choosing w in CBS+HWY(w) or w_1, w_2 in ECBS(w_1)+HWY(w_2) is important:
  - Increasing w_1: allows longer paths, which provides agents with the flexibility to avoid collisions by moving around other agents
  - Increasing w_2: guides the agents through the highways
- Choosing the highways according to the map affect the runtime and solution cost dramatically
- This approach can be effective to real worlds problems (like Kiva's domains)

Conclusions and Insights

- CBS+HWY(w) is w-suboptimal and ECBS(w_1)+HWY(w_2) is ($w_1 \cdot w_2$)-suboptimal algorithms for MAPF
- Choosing w in CBS+HWY(w) or w_1, w_2 in ECBS(w_1)+HWY(w_2) is important:
  - Increasing w_1: allows longer paths, which provides agents with the flexibility to avoid collisions by moving around other agents
  - Increasing w_2: guides the agents through the highways
- Choosing the highways according to the map affect the runtime and solution cost dramatically
- This approach can be effective to real worlds problems (like Kiva's domains)

Future Work

Future Work and Ideas

- Develop approaches that determine highways automatically
- Investigate whether inflating the edge costs of the given graph in addition to inflating the heuristic values provides additional benefits
- Figure out whether penalizing movement costs against highway edges helps to improve the performance
- Combine highways and warehouse rearrangement

Future Work and Ideas

- Develop approaches that determine highways automatically
- Investigate whether inflating the edge costs of the given graph in addition to inflating the heuristic values provides additional benefits
- Figure out whether penalizing movement costs against highway edges helps to improve the performance
- Combine highways and warehouse rearrangement

Future Work and Ideas

- Develop approaches that determine highways automatically
- Investigate whether inflating the edge costs of the given graph in addition to inflating the heuristic values provides additional benefits
- Figure out whether penalizing movement costs against highway edges helps to improve the performance
- Combine highways and warehouse rearrangement

Future Work and Ideas

- Develop approaches that determine highways automatically
- Investigate whether inflating the edge costs of the given graph in addition to inflating the heuristic values provides additional benefits
- Figure out whether penalizing movement costs against highway edges helps to improve the performance
- Combine highways and warehouse rearrangement

Future Work - iECBS

- Main ideas in "Improved Solvers for Bounded-Suboptimal Multi-Agent Path Finding" by L. Cohen, T. Uras, T. K. S. Kumar, H. Xu, N. Ayanian, S. Koenig:
  - Automatically generating highways -
    - Graphical model-based approach.
    - Heat map-based approach.
  - iECBS(w) - w-suboptimal algorithm.

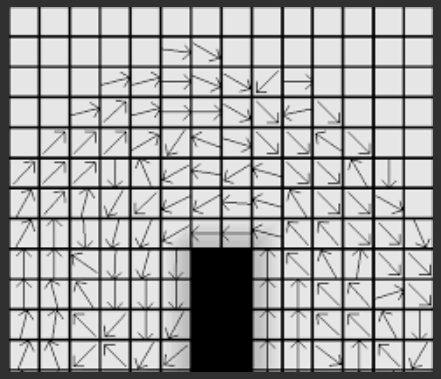
	Low-level (for Agent i in High-level node N)		High-level		SUB-OPT
	Sort $n \in \text{OPEN}^i(N)$	Sort $n \in \text{FOCAL}^i(N)$	Sort $N \in \text{OPEN}$	Sort $N \in \text{FOCAL}$	
ECBS(w_1)	1. $g^i(n) + h^i_{sp}(n)$ 2. $g^i(n)$	1. $h^i_{c(N)}(n)$ 2. $g^i(n) + h^i_{sp}(n)$ 3. $g^i(n)$	1. $\sum_i \min_{n \in \text{OPEN}^i} (g^i(n) + h^i_{sp}(n))$	1. $h_c(N)$ 2. $g(N)$	w_1
ECBS(w_1)+HWY(w_2)	1. $g^i(n) + h^i_{hwy}(n)$ 2. $g^i(n)$	1. $h^i_{c(N)}(n)$ 2. $g^i(n) + h^i_{hwy}(n)$ 3. $g^i(n)$	1. $\sum_i \min_{n \in \text{OPEN}^i} (g^i(n) + h^i_{hwy}(n))$	1. $h_c(N)$ 2. $g(N)$	$w_1 \cdot w_2$
iECBS(w_1)	1. $g^i(n) + h^i_{sp}(n)$ 2. $g^i(n)$	1. $h^i_{c(N)}(n)$ 2. $g^i(n) + h^i_{hwy}(n)$ 3. $g^i(n)$	1. $\sum_i \min_{n \in \text{OPEN}^i} (g^i(n) + h^i_{sp}(n))$	1. $h_c(N)$ 2. $g(N)$	w_1

Summarizes the differences in the high-level and low-level searches for various MAPF solvers. (sp = shortest path; hwy = highways; c = collisions)

Highways in Nature - Stigmergy

- Stigmergy is a mechanism of indirect coordination, through the environment, between agents or actions.
- Uses in nature for various tasks (collecting food for example).

Ant paths built from pheromone traces



Direction map [2]

Stigmergy simulation[1]

[1] vimeo.com/200946575

[2] M. Renee Jansen, Nathan R. Sturtevant: Direction Maps for Cooperative Pathfinding. AIIDE 2008

Summary

Summary

- Optimal Vs. bounded-suboptimal approach
- ECBS(w_1) algorithm - w_1-suboptimal
- Highways:
  - Heuristic - $h^j_{hwy}(s) = \min_\pi \sum^{\infty}_{(s_i,s_{i+1})\in\pi} \begin{cases} 1, & \text{if } (s_i, s_{i+1}) \in E_{hwy} \\ w, & \text{otherwise} \end{cases}$
  - How to use it
  - CBS+HWY(w_1): w_1-suboptimal
  - ECBS(w_1)+HWY(w_2): $(w_1 \cdot w_2)$-suboptimal
- Experimental results and conclusions
- Future work:
  - Automatically generating highways
  - iECBS

Summary

- Optimal Vs. bounded-suboptimal approach
- ECBS(w_1) algorithm - w_1-suboptimal
- Highways:
  - Heuristic - $h^j_{hwy}(s) = \min_\pi \sum^{\infty}_{(s_i,s_{i+1})\in\pi} \begin{cases} 1, & \text{if } (s_i, s_{i+1}) \in E_{hwy} \\ w, & \text{otherwise} \end{cases}$
  - How to use it
  - CBS+HWY(w_1): w_1-suboptimal
  - ECBS(w_1)+HWY(w_2): $(w_1 \cdot w_2)$-suboptimal
- Experimental results and conclusions
- Future work:
  - Automatically generating highways
  - iECBS

Summary

- Optimal Vs. bounded-suboptimal approach
- ECBS(w_1) algorithm - w_1-suboptimal
- Highways:
  - Heuristic - $h^j_{hwy}(s) = \min_\pi \sum^{\infty}_{(s_i,s_{i+1})\in\pi} \begin{cases} 1, & \text{if } (s_i, s_{i+1}) \in E_{hwy} \\ w, & \text{otherwise} \end{cases}$
  - How to use it
  - CBS+HWY(w_1): w_1-suboptimal
  - ECBS(w_1)+HWY(w_2): $(w_1 \cdot w_2)$-suboptimal
- Experimental results and conclusions
- Future work:
  - Automatically generating highways
  - iECBS

Summary

- Optimal Vs. bounded-suboptimal approach
- ECBS(w_1) algorithm - w_1-suboptimal
- Highways:
  - Heuristic - $h^j_{hwy}(s) = \min_\pi \sum^{\infty}_{(s_i,s_{i+1})\in\pi} \begin{cases} 1, & \text{if } (s_i, s_{i+1}) \in E_{hwy} \\ w, & \text{otherwise} \end{cases}$
  - How to use it
  - CBS+HWY(w_1): w_1-suboptimal
  - ECBS(w_1)+HWY(w_2): $(w_1 \cdot w_2)$-suboptimal
- Experimental results and conclusions
- Future work:
  - Automatically generating highways
  - iECBS

Summary

- Optimal Vs. bounded-suboptimal approach
- ECBS(w_1) algorithm - w_1-suboptimal
- Highways:
  - Heuristic - $h^j_{hwy}(s) = \min_\pi \sum^{\infty}_{(s_i,s_{i+1})\in\pi} \begin{cases} 1, & \text{if } (s_i, s_{i+1}) \in E_{hwy} \\ w, & \text{otherwise} \end{cases}$
  - How to use it
  - CBS+HWY(w_1): w_1-suboptimal
  - ECBS(w_1)+HWY(w_2): $(w_1 \cdot w_2)$-suboptimal
- Experimental results and conclusions
- Future work:
  - Automatically generating highways
  - iECBS

References

- Cohen, L. *et al.* *Anytime Focal Search with Application.* (IJCAI, 2018).
- Cohen, L. *et al.* *Improved Solvers for Bounded-Suboptimal Multi-Agent Path Finding.* (IJCAI, 2016).
- Cohen, L., Uras, T. & Koenig, S. *Feasibility study: Using highways for bounded-suboptimal multi-agent path finding.* (SOCS, 2015).
- M. Barer G. Sharon, R. S. & Felner, A. *Suboptimal Variants of the Conflict-Based Search Algorithm for the Multi-Agent Pathfinding Problem.* (ECAI, 2014).
- M. Renee Jansen, N. R. S. *Direction Maps for Cooperative Pathfinding.* (AIIDE, 2008).

Thank You!