USC Viterbi
School of Engineering

Multi-Agent Path Finding

N. Ayanian, T. Cai, L. Cohen, W. Hoenig,
Sven Koenig, S. Kumar, H. Ma, T. Uras, H. Xu,
S. Young, D. Zhang
University of Southern California
C. Tovey
Georgia Institute of Technology
G. Sharon
University of Texas at Austin

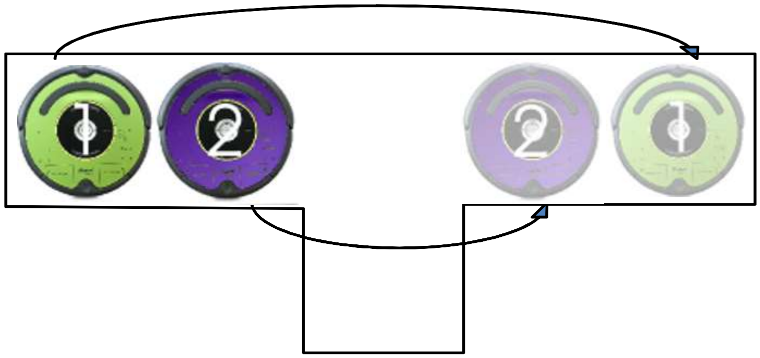
idm-lab.org
skoenig@usc.edu

2

Multi-Agent Path Finding (MAPF)

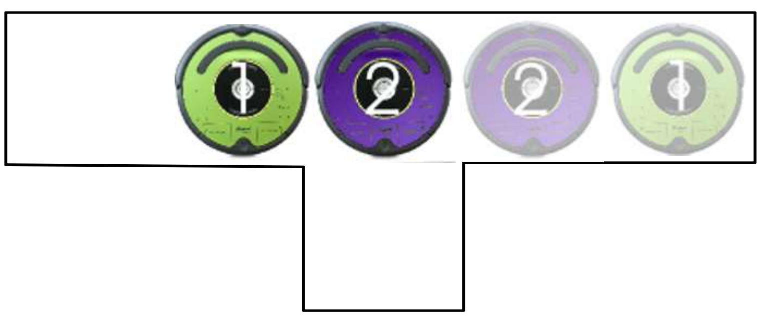
- Multi-agent path finding (MAPF)
  - Given: a number of agents (each with a start and goal location) and a known environment
  - Task: find collision-free paths for the agents from their start to their goal locations that minimize some objective
- Objectives
  - Makespan: latest arrival time of an agent at its goal location
  - Flowtime: sum of the arrival times of all agents at their goal locations

Multi-Agent Path Finding (MAPF)



4-neighbor grid

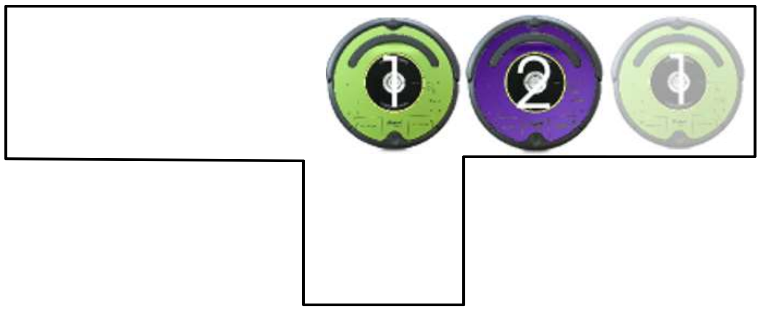
Multi-Agent Path Finding (MAPF)



4-neighbor grid

5

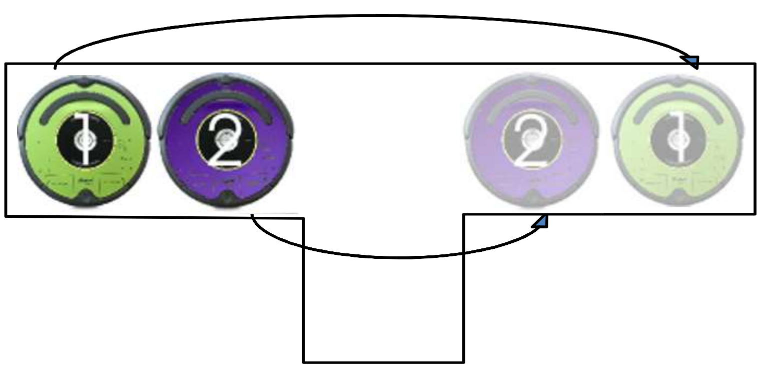
Multi-Agent Path Finding (MAPF)



4-neighbor grid

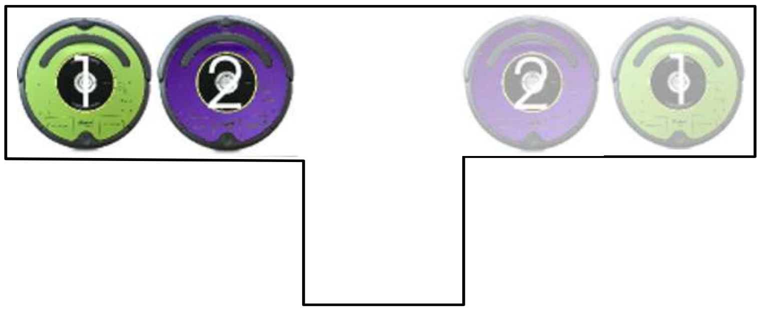
6

Multi-Agent Path Finding (MAPF)



4-neighbor grid

Multi-Agent Path Finding (MAPF)



4-neighbor grid

Multi-Agent Path Finding (MAPF)

4-neighbor grid

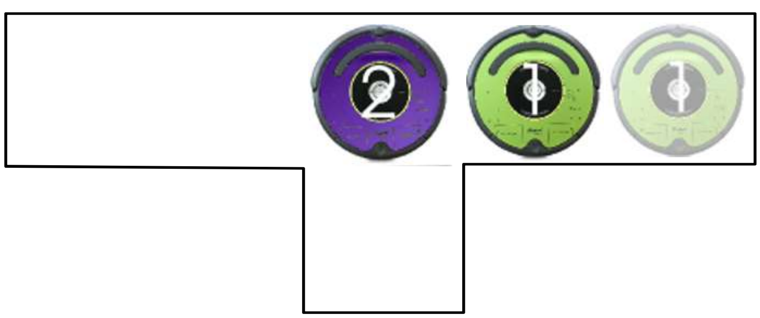
9

Multi-Agent Path Finding (MAPF)

4-neighbor grid

10

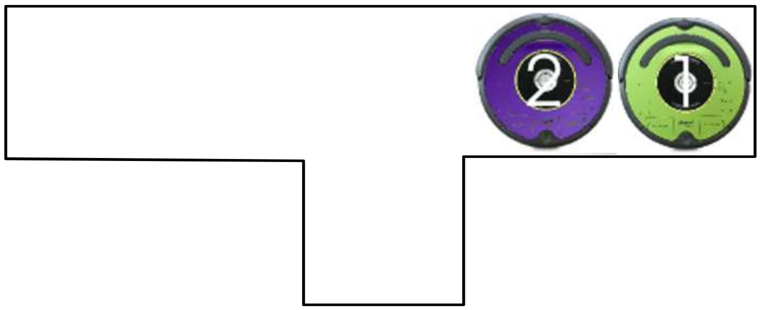
Multi-Agent Path Finding (MAPF)



4-neighbor grid

11

Multi-Agent Path Finding (MAPF)

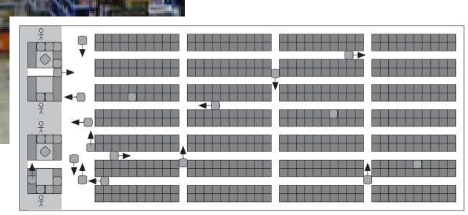


4-neighbor grid

12

Multi-Agent Path Finding (MAPF)

- Application: Amazon fulfillment centers



[work by Kiva Systems/Amazon Robotics, not me]

Multi-Agent Path Finding (MAPF)

- Application: Amazon fulfillment centers

[work by Kiva Systems/Amazon Robotics, not me]

Multi-Agent Path Finding (MAPF)

- Application: autonomous tug robots (joint with NASA Ames)

[Google Earth]

[Morris]

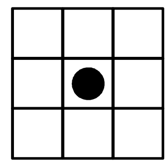
  – Reduce pollution
  – Reduce energy consumption
  – Reduce human danger
  – Reduce human workload
  – Reduce airport size

15

Multi-Agent Path Finding (MAPF)

Robot

Agent



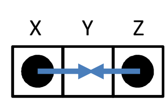
- Simplifying assumptions
  - Point robots
  - No kinematic constraints
  - Discretized environment
    - we use grids here but most techniques work on planar graphs in general

4-neighbor grid

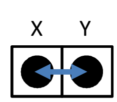
16

Multi-Agent Path Finding (MAPF)

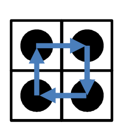
- Each agent moves N, E, S or W into an adjacent unblocked cell
- Not allowed (“vertex collision”)
  - Agent 1 moves from X to Y
  - Agent 2 moves from Z to Y



- Not allowed (“edge collision”)
  - Agent 1 moves from X to Y
  - Agent 2 moves from Y to X



- Allowed



4-neighbor grid

17

Multi-Agent Path Finding (MAPF)

- Optimal MAPF algorithms
  - Theorem [Yu and LaValle]: MAPF is NP-hard to solve optimally for makespan or flowtime minimization

[www.random-ideas.net]

- Bounded-suboptimal MAPF algorithms
  - Theorem: MAPF is NP-hard to approximate within any factor less than 4/3 for makespan minimization on graphs in general

18

Multi-Agent Path Finding (MAPF)

- Reduction from (≤3, =3)-SAT: It is NP-complete to determine whether a given (≤3, =3)-SAT instance is satisfiable

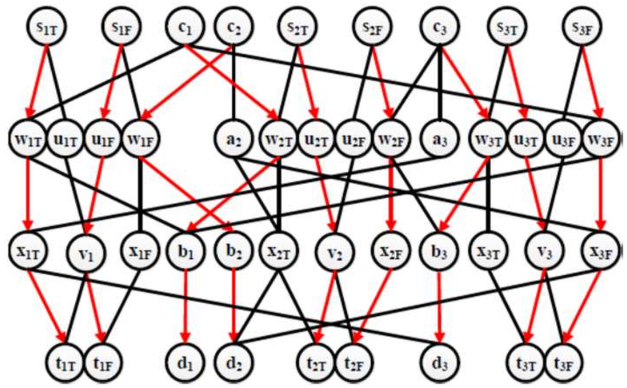
- Each clause contains at most 3 literals
- Each variable appears in exactly 3 clauses
- Each variable appears uncomplemented at least once
- Each variable appears complemented at least once

- Example: $(X_1 \vee X_2 \vee \overline{X_3}) \wedge (\overline{X_1} \vee X_2 \vee \overline{X_3}) \wedge (X_1 \vee \overline{X_2} \vee X_3)$

19

Multi-Agent Path Finding (MAPF)

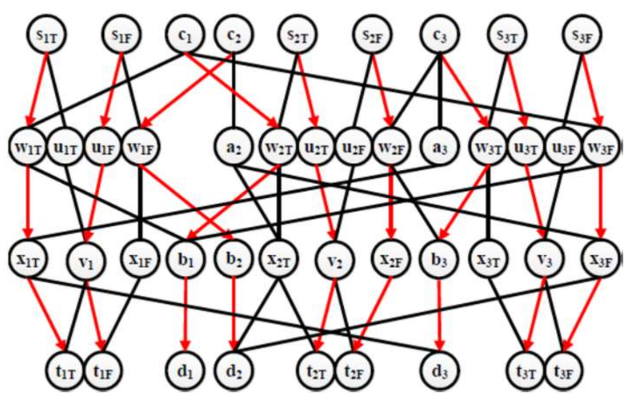
- Example: $(X_1 \vee X_2 \vee \overline{X_3}) \wedge (\overline{X_1} \vee X_2 \vee \overline{X_3}) \wedge (X_1 \vee \overline{X_2} \vee X_3)$



20

Multi-Agent Path Finding (MAPF)

- Example: $(X_1 \vee X_2 \vee \overline{X_3}) \wedge (\overline{X_1} \vee X_2 \vee \overline{X_3}) \wedge (X_1 \vee \overline{X_2} \vee X_3)$



X1 ≡ false X2 ≡ true X3 ≡ true

21

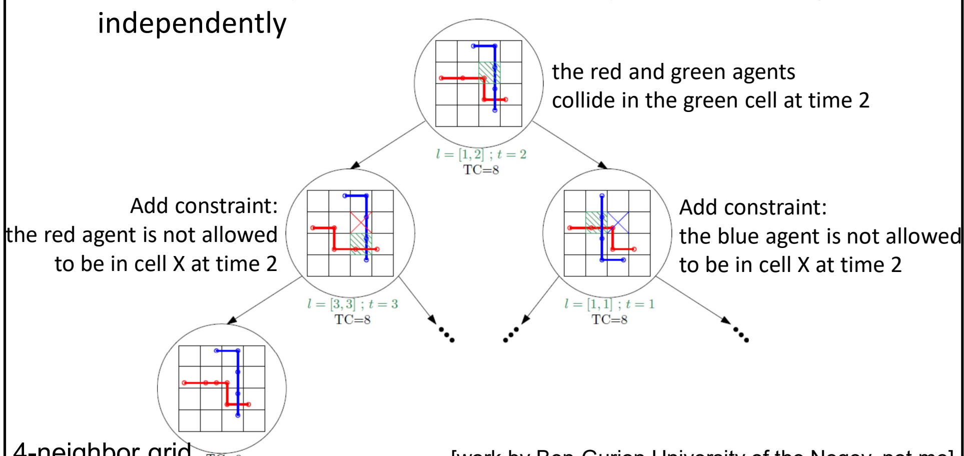
Multi-Agent Path Finding (MAPF)

- Makespan is 3 if and only if (≤3, =3)-SAT instance is satisfiable
- Makespan is 4 if and only if (≤3, =3)-SAT instance is unsatisfiable
- Any MAPF approximation algorithm with ratio 4/3 – ε thus computes a MAPF plan with makespan 3 whenever the (≤3, =3)-SAT instance is satisfiable and therefore solves it

22

Conflict-Based Search with Highways

- Conflict-based search [Sharon, Stern, Felner and Sturtevant]: Bounded-suboptimal MAPF solver that plans for each agent independently



[work by Ben-Gurion University of the Negev, not me]

Conflict-Based Search with Highways

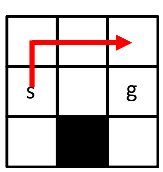
- Experience graphs [Phillips, Cohen, Chitta and Likhachev]: Bounded-suboptimal single-agent path planner so that the resulting path uses edges in a given subgraph (the experience graph) as much as possible

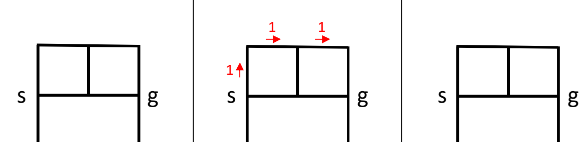
[work by CMU, not me]

Conflict-Based Search with Highways

	optimal	suboptimality bound 4	
	regular (no highways)	highways #1	highways #2 (experience graphs)

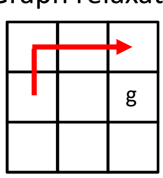
- Graph for an A* search

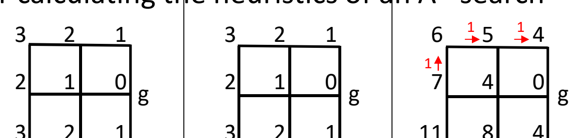




all costs are 1 | all costs are 4 except for the ones shown | all costs are 1

- Graph relaxation for calculating the heuristics of an A* search





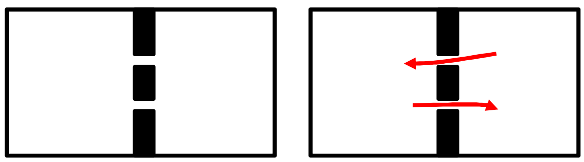
all costs are 1 | all costs are 1 | all costs are 4 except for the ones shown

4-neighbor grid

[work by CMU, not me]

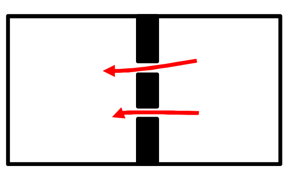
Conflict-Based Search with Highways

- Conflict-based search with highways (ECBS+HWY): Bounded suboptimal MAPF solver
  - Conflict-based search
  - Experience graphs create lanes (called highways) for the agents to avoid head-to-head collisions, which decreases the computation time of conflict-based search



Conflict-Based Search with Highways

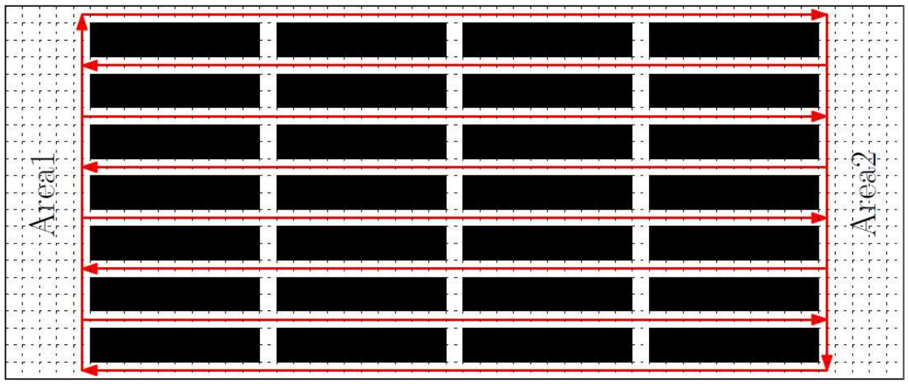
- Conflict-based search with highways (ECBS+HWY)
  - Highways provide consistency and thus predictability of agent movement, which might be important for human co-workers
  - Highways do not make MAPF instances unsolvable because they are only used as advice rather than hard constraints



27

Conflict-Based Search with Highways

- Conflict-based search with highways (ECBS+HWY)

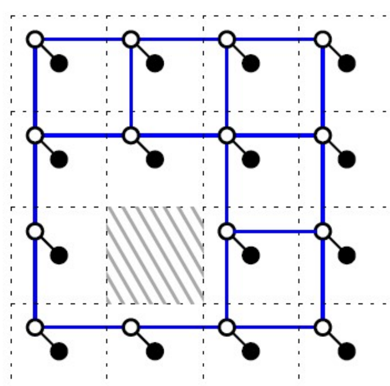


4-neighbor grid

28

Conflict-Based Search with Highways

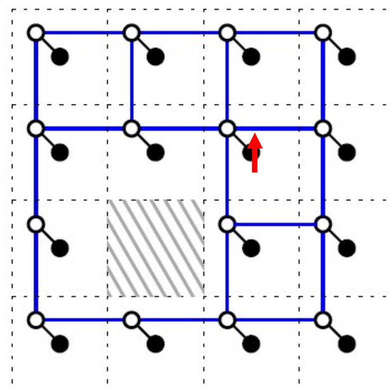
- Learning highways with graphical models
- Plan a shortest path for each agent independently
- Direction vector of a cell: Average of entry and exit directions of each path for the given cell
- Features
  - Collision?
  - Direction of direction vector (N, E, S, W)
  - Magnitude of direction vector > 0.5?



4-neighbor grid

Conflict-Based Search with Highways

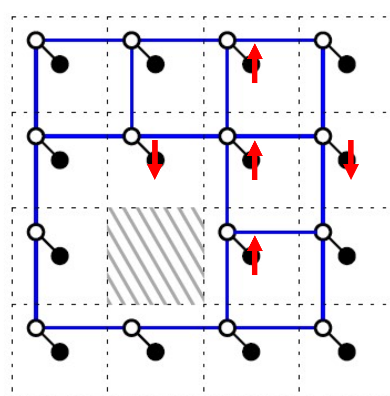
- Learning highways with graphical models
- Plan a shortest path for each agent independently
- Direction vector of a cell: Average of entry and exit directions of each path for the given cell
- Features
  - Collision?
  - Direction of direction vector (N, E, S, W)
  - Magnitude of direction vector > 0.5?



4-neighbor grid

Conflict-Based Search with Highways

- Learning highways with graphical models
- Plan a shortest path for each agent independently
- Direction vector of a cell: Average of entry and exit directions of each path for the given cell
- Features
  - Collision?
  - Direction of direction vector (N, E, S, W)
  - Magnitude of direction vector > 0.5?



4-neighbor grid

31

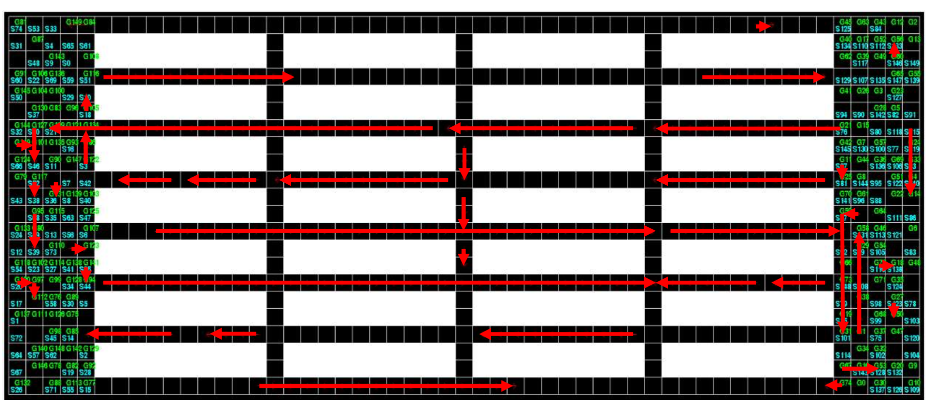
Conflict-Based Search with Highways

- Graphical models basically encode probabilistic knowledge
  - If agents collide in a cell, make it more likely that there is a highway in that cell
  - If most agents move northward in a cell, make it more likely that a highway in that cell, if any, is a northward one
  - If a northward highway is in a cell, make it more likely that highways in its northern and southern neighbors, if any, are also northward ones (to form a longer lane)
  - If a northward highway is in a cell, make it more likely that highways in its western and eastern neighbors, if any, are southward ones (to form adjacent lanes in opposite directions)

4-neighbor grid

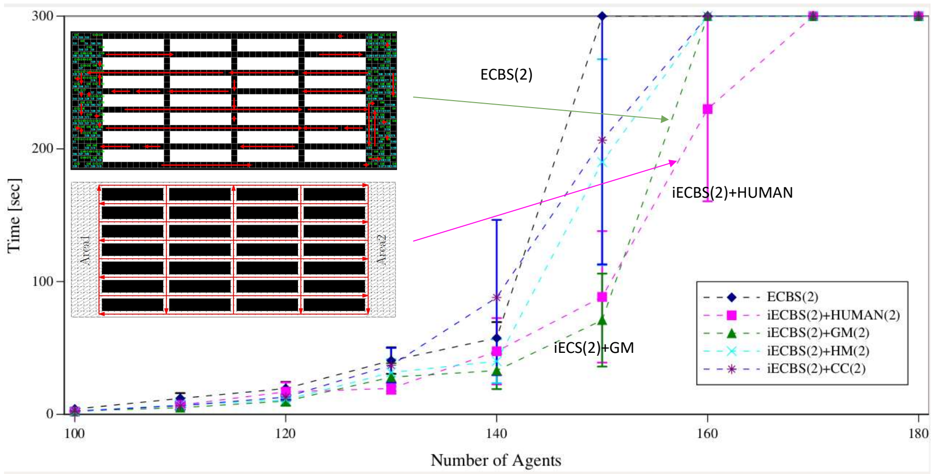
32

Conflict-Based Search with Highways



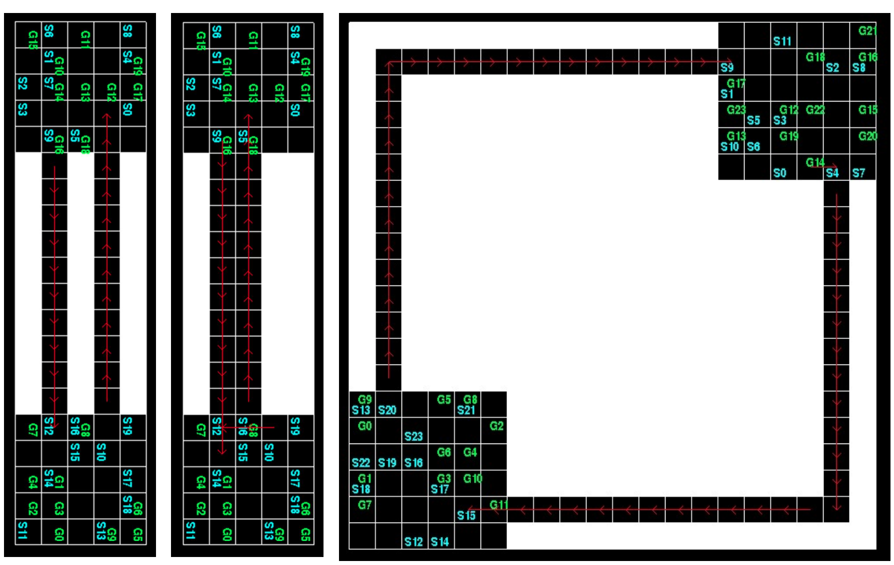
4-neighbor grid

Conflict-Based Search with Highways



4-neighbor grid

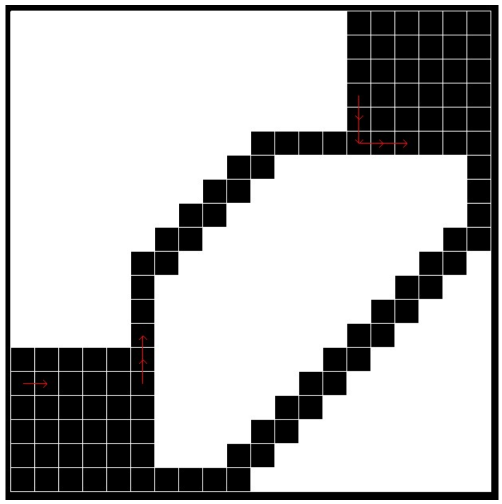
Conflict-Based Search with Highways



4-neighbor grid

35

Conflict-Based Search with Highways



4-neighbor grid

36

Conflict-Based Search with Highways

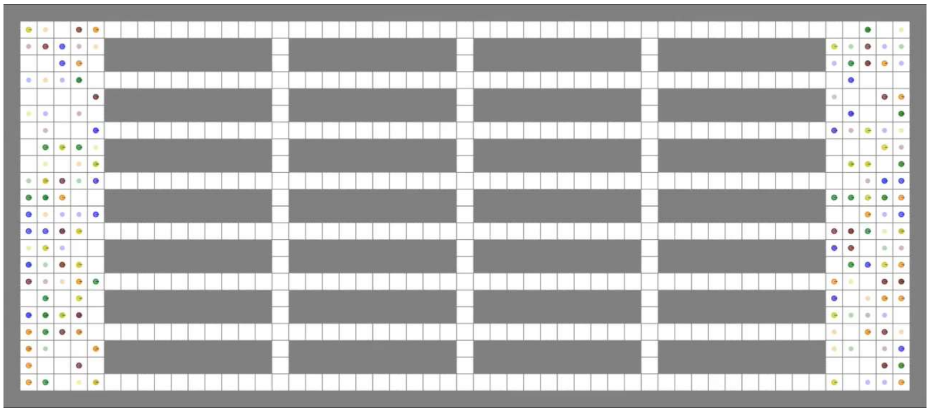
- Rapid random restarts help to solve more multi-agent path finding problems within a given runtime limit.
- Here: We randomize the ordering in which the agents plan their paths in the high-level root node.

runs	time limit	38 “easy”	12 “hard”	50 total
1	300 sec	100.00%	0.00%	76.00%
3	100 sec	97.65%	96.87%	97.60%
5	60 sec	98.57%	98.81%	98.70%

37

Conflict-Based Search with Highways

- Conflict-based search with highways (ECBS+HWY)



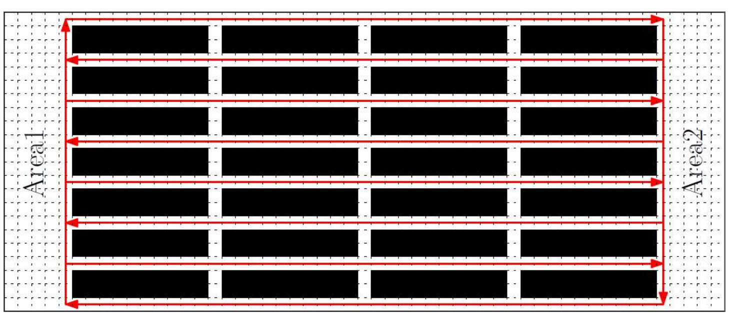
4-neighbor grid

8x

38

Conflict-Based Search with Highways

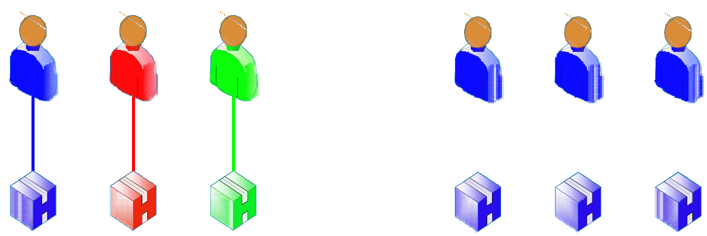
- 130 agents (half moving to the right, half moving to the left)
- Minimize flowtime with suboptimality bound 2



- Conflict-based search: 48.5 seconds
- Conflict-based search with highways: 29.1 seconds

39

Extensions



non-anonymous MAPF	**anonymous MAPF**
NP-hard	polynomial-time solvable for makespan minimization
solved with A* approaches e.g. conflict-based search or M*	solved with flow approaches e.g. max-flow algorithm

40

Anonymous MAPF

- (Non-anonymous) MAPF
  - Given: a number of agents (each with a start and goal location) and a known environment
  - Task: find collision-free paths for the agents from their start to their goal locations that minimize makespan or flowtime
- Anonymous MAPF
  - Given: a number of agents (each with a start location), an equal number of goal locations, and a known environment
  - Task: assign a different goal location to each agent and then find collision-free paths for the agents from their start to their goal locations that minimize makespan or flowtime

Anonymous MAPF

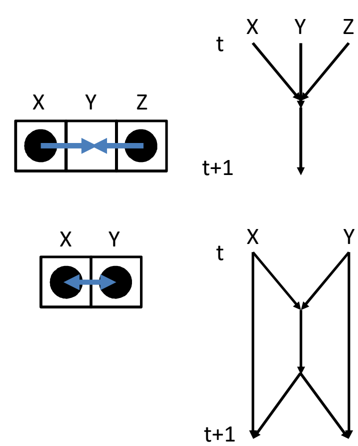
- Theorem [Yu and Lavalle]: An anonymous MAPF instance admits a MAPF plan with makespan at most T if and only if the time-expanded network with T periods admits a max flow of the number of agents.

[work by the University of Illinois at Urbana-Champaign, not me]

Anonymous MAPF

all edges have capacity one

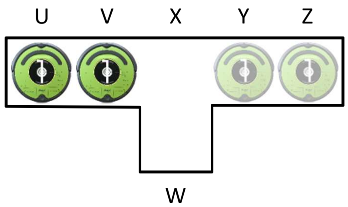
- Each agent moves N, E, S or W into an adjacent unblocked cell
- Not allowed ("vertex collision")
  - Agent 1 moves from X to Y
  - Agent 2 moves from Z to Y
- Not allowed ("edge collision")
  - Agent 1 moves from X to Y
  - Agent 2 moves from Y to X

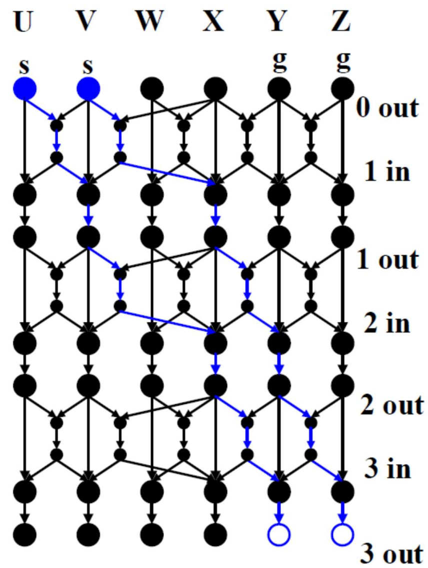


[work by the University of Illinois at Urbana-Champaign, not me]

43

Anonymous MAPF

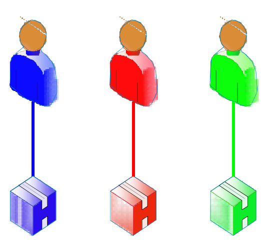




[work by the University of Illinois at Urbana-Champaign, not me]

44

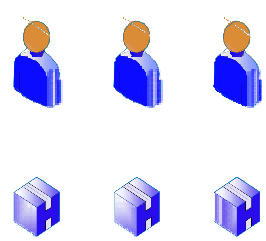
Target Assignment and Path Finding (TAPF)



non-anonymous MAPF

NP-hard

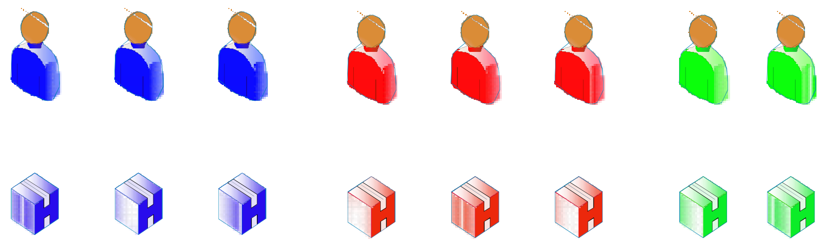
solved with A* approaches
e.g. conflict-based search or M*



anonymous MAPF

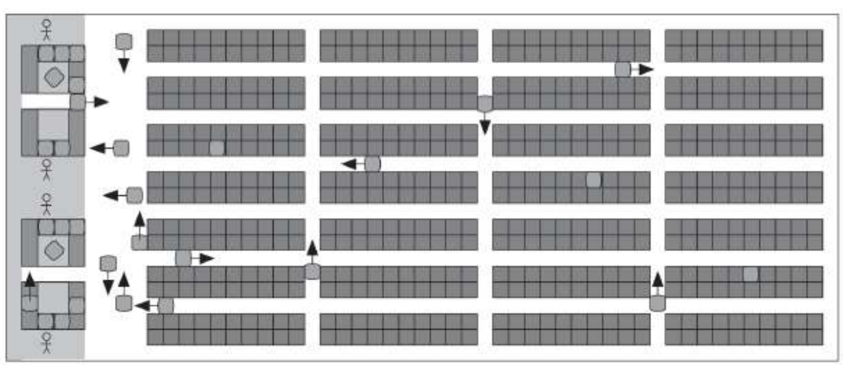
polynomial-time solvable for makespan minimization
solved with flow approaches
e.g. max-flow algorithm

Target Assignment and Path Finding (TAPF)



mix of non-anonymous and anonymous MAPF
Target Assignment and Path Finding (TAPF)
with k groups (here: 3), also called types

Target Assignment and Path Finding (TAPF)



[Wurman, D'Andrea and Mountz]

Group 0: Agents that move from the packing stations to the storage locations
Group 1: Agents that move from the storage locations to Packing Station 1
Group 2: Agents that move from the storage locations to Packing Station 2
Group 3: Agents that move from the storage locations to Packing Station 3

47

Target Assignment and Path Finding (TAPF)

- Theorem: TAPF (with k>1 groups) is NP-hard to solve optimally for makespan or flowtime minimization
- Theorem: TAPF (with k>1 groups) is NP-hard to approximate within any factor less than 4/3 for makespan minimization on graphs in general

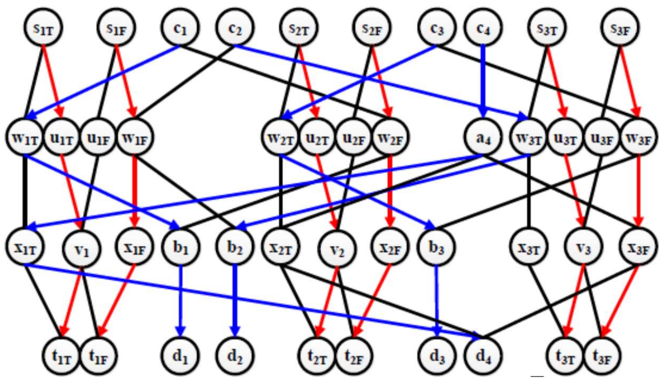
48

Target Assignment and Path Finding (TAPF)

- Reduction from $2/\overline{2}/3$-SAT: It is NP-complete to determine whether a given $2/\overline{2}/3$-SAT instance is satisfiable
- Each variable appears in exactly 3 clauses
- Each variable appears uncomplemented in a clause of size two
- Each variable appears complemented in a clause of size two
- Each variable appears in a clause of size three
- Example: $(X_1 \vee \overline{X_2}) \wedge (\overline{X_1} \vee X_3) \wedge (X_2 \vee \overline{X_3}) \wedge (X_1 \vee X_2 \vee \overline{X_3})$

Target Assignment and Path Finding (TAPF)

- Example: $(X_1 \vee \overline{X}_2) \wedge (\overline{X}_1 \vee X_3) \wedge (X_2 \vee \overline{X}_3) \wedge (X_1 \vee X_2 \vee \overline{X}_3)$



Target Assignment and Path Finding (TAPF)

- CBM combines the max-flow algorithm and conflict-based search to minimize makespan for TAPF instances
  - CBM uses the max-flow algorithm to assign goal locations and plan paths for all agents in a group (to solve the corresponding anonymous MAPF instance)
    CBM actually uses a min-cost max-flow algorithm since is important to choose paths that result in few collisions with agents from other groups
  - CBM treats each group as a meta-agent and uses conflict-based search to plan sets of paths for all meta-agents (to solve the corresponding non-anonymous MAPF problem)

- Theorem: CBM is complete and optimal for minimizing makespan for TAPF instances

51

Target Assignment and Path Finding (TAPF)

- Experimental results

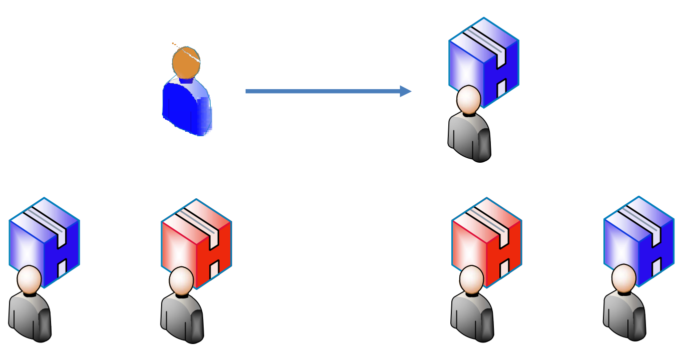
	CBM		Mixed Integer Program	
Agents	Time	Success	Time	Success
10	0.34	100%	18.24	100%
20	0.78	100%	62.85	94%
30	1.71	100%	108.75	66%
40	2.95	100%	152.98	14%
50	5.32	100%	161.95	4%

30x30 4-neighbor grids with 10% randomly blocked cells and a 5-minute time limit

52

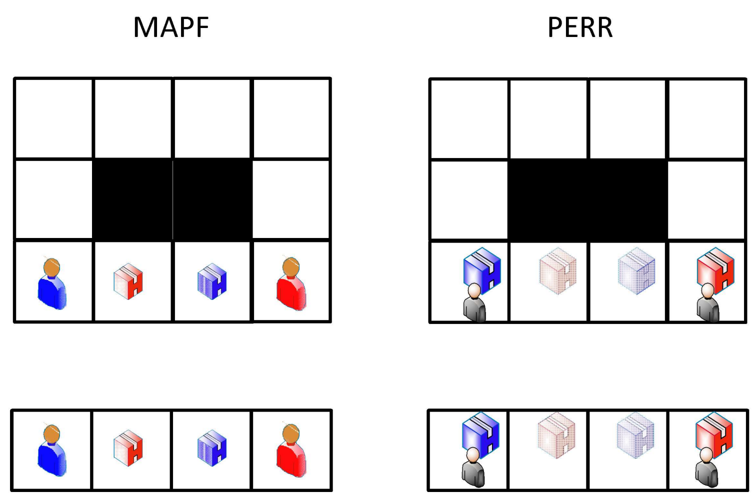
Package Exchange Robot Routing (PERR)

- The Package Exchange Robot Routing problem (PERR)
  - Each agent carries exactly one package
  - Each package needs to be delivered to a given goal location
  - Two agents in adjacent locations can exchange packages



53

Package Exchange Robot Routing (PERR)



54

Package Exchange Robot Routing (PERR)

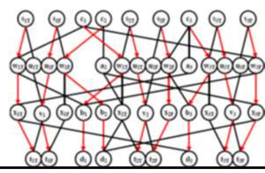
- Theorem: All PERR instances (with k≥1 groups) are solvable (as long as all goal locations are different and all agents are in the same connected components as their goal locations)
- Theorem: Plans with polynomial makespans and flowtimes can be found in polynomial time.

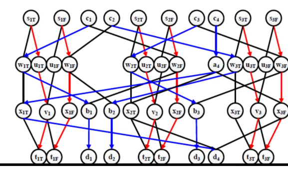
Package Exchange Robot Routing (PERR)

- Theorem: PERR (with k>1 groups) is NP-hard to solve optimally for makespan or flowtime minimization

- Theorem: PERR (with k>1 groups) is NP-hard to approximate within any factor less than 4/3 for makespan minimization on graphs in general

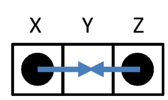
- Reductions from ≤3,=3-SAT or 2/$\overline{2}$/3 SAT as before (because transfers do not help for our constructions)



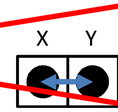


Package Exchange Robot Routing (PERR)

- Each agent moves N, E, S or W into an adjacent unblocked cell
- Not allowed (“vertex collision”)
  – Agent 1 moves from X to Y
  – Agent 2 moves from Z to Y



- ~~Not allowed (“edge collision”)~~
  – ~~Agent 1 moves from X to Y~~
  – ~~Agent 2 moves from Y to X~~



- PERR instances can be solved with versions of conflict-based search and multi-commodity flow algorithms

57

Execution of MAPF Plans

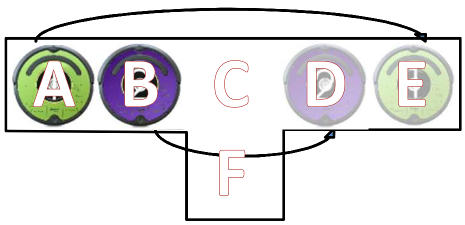
- Planning uses models that are not completely accurate
  - Robots are not completely synchronized
  - Robots do not move exactly at the nominal speed
  - Robots have unmodeled kinematic constraints
  - …
- Plan execution will therefore likely deviate from the plan
- Replanning whenever plan execution deviates from the plan is intractable since it is NP-hard to find good plans

58

Execution of MAPF Plans

- MAPF-POST makes use of a simple temporal network to post-process the output of a multi-agent path finding solver in polynomial time to allow for plan execution on robots
  - Takes into account edge lengths
  - Takes into account velocity limits (for both robots and edges)
  - Guarantees a safety distance among robots
  - Avoids replanning in many cases

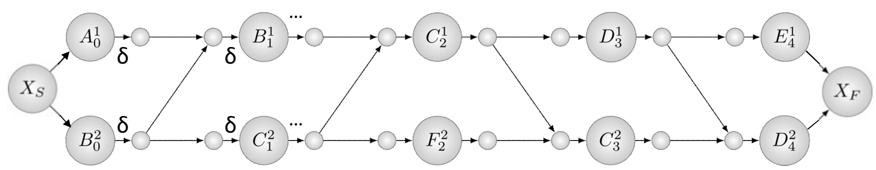
Execution of MAPF Plans



Agent 1 A → B → C → D → E
Agent 2 B → C → F → C → D

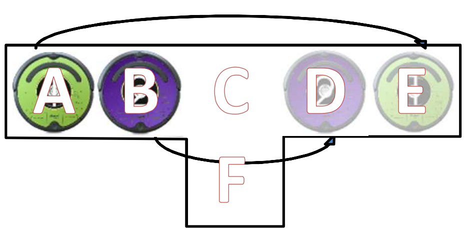
Precedence Graph

vertex = event that an agent arrives at a location



4-neighbor grid

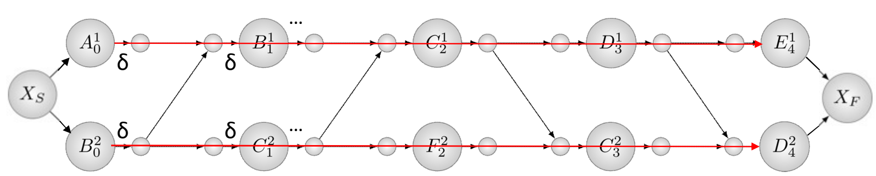
Execution of MAPF Plans



Agent 1 A → B → C → D → E
Agent 2 B → C → F → C → D

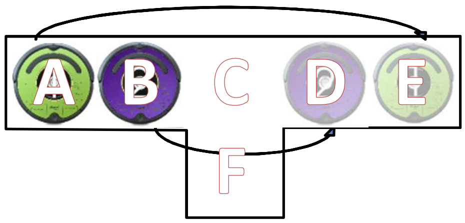
Precedence Graph

Type 1 edge = order in which the same agent arrives at locations



4-neighbor grid

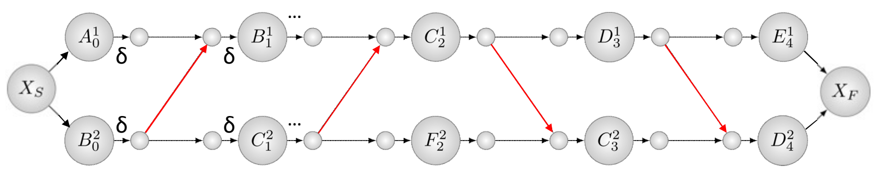
Execution of MAPF Plans



Agent 1 A → B → C → D → E
Agent 2 B → C → F → C → D

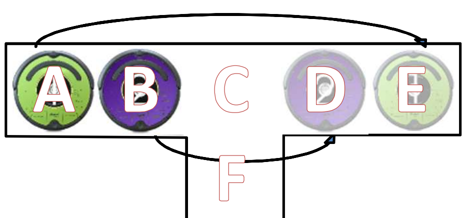
Precedence Graph

Type 2 edge = order in which two different agents arrive at the same location



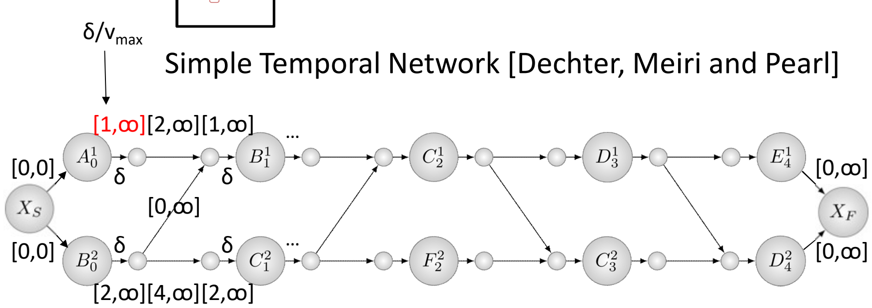
4-neighbor grid

Execution of MAPF Plans



Agent 1 A → B → C → D → E
Agent 2 B → C → F → C → D

Simple Temporal Network [Dechter, Meiri and Pearl]



4-neighbor grid

Execution of MAPF Plans

- Minimize makespan and flowtime
  - Schedule each arrival in a location as early as allowed by the constraints

Minimize $\sum_{j=1}^{K} t(v^j)$
such that $t(X_S) = 0$
and, for all $e = (v, v') \in \mathcal{E}'$,
$t(v') - t(v) \geq LB(e)$
$t(v') - t(v) \leq UB(e)$

polynomial time

Execution of MAPF Plans

- Maximize safety distance
  - Assume that each agent moves with a constant velocity of at most v_{min} along every Type 1 edge
  - Then, the safety distance is $2\delta v_{min}/v_{max}$

Maximize v^*_{min}
such that $t(X_S) = 0$
and, for all $e = (v, v') \in \mathcal{E}'$,
$t(v') - t(v) \geq LB(e)$
$t(v') - t(v) \leq UB(e)$
$t(v') - t(v) \leq l(e)(v^*_{\min})^{-1}$ if e is a Type 1 edge

polynomial time

Execution of MAPF Plans

- Main loop
  - Run Conflict-Based Search with Highways to find a MAPF plan (slow)

Execution of MAPF Plans

- Main loop
  - Run Conflict-Based Search with Highways to find a MAPF plan (slow)
  - Construct a simple temporal network for the MAPF plan

67

Execution of MAPF Plans

- Main loop
  - Run Conflict-Based Search with Highways to find a MAPF plan (slow)
  - Construct a simple temporal network for the MAPF plan
  - Determine the earliest arrival times in the nodes

68

Execution of MAPF Plans

- Main loop
  - Run Conflict-Based Search with Highways to find a MAPF plan (slow)
  - Construct a simple temporal network for the MAPF plan
  - Determine the earliest arrival times in the nodes

  - Calculate speeds for the robots from the earliest arrival times

Execution of MAPF Plans

- Main loop
  - Run Conflict-Based Search with Highways to find a MAPF plan (slow)
  - Construct a simple temporal network for the MAPF plan
  - Determine the earliest arrival times in the nodes
  - Calculate speeds for the robots from the earliest arrival times
  - Move robots along their paths in the MAPF plan with these speeds

Execution of MAPF Plans

- Main loop
  - Run Conflict-Based Search with Highways to find a MAPF plan (slow)
  - Construct a simple temporal network for the MAPF plan
  - Determine the earliest arrival times in the nodes
  - Calculate speeds for the robots from the earliest arrival times
  - Move robots along their paths in the MAPF plan with these speeds
  - If plan execution deviates from the plan, then

Execution of MAPF Plans

- Main loop
  - Run Conflict-Based Search with Highways to find a MAPF plan (slow)
  - Construct a simple temporal network for the MAPF plan
  - Determine the earliest arrival times in the nodes
  - Calculate speeds for the robots from the earliest arrival times
  - Move robots along their paths in the MAPF plan with these speeds
  - If plan execution deviates from the plan, then

Execution of MAPF Plans

- Main loop
  - Run Conflict-Based Search with Highways to find a MAPF plan (slow)
  - Construct a simple temporal network for the MAPF plan
  - Determine the earliest arrival times in the nodes
  - Calculate speeds for the robots from the earliest arrival times
  - Move robots along their paths in the MAPF plan with these speeds
  - If plan execution deviates from the plan, then

73

Execution of MAPF Plans

- Main loop
  - Run Conflict-Based Search with Highways to find a MAPF plan
  - Construct a simple temporal network for the MAPF plan
  - Determine the earliest arrival times in the nodes
  - If they do not exist, then
  - Calculate speeds for the robots from the earliest arrival times
  - Move robots along their paths in the MAPF plan with these speeds
  - If plan execution deviates from the plan, then

74

Execution of MAPF Plans

- MAPF solver: ECBS+HWY
- MAPF-POST: C++, boost graph library, Gurobi LP solver
- PC: i7-4600U 2.1 GHz, 12 GB RAM
- Terrain: 4x3 gridworld with 1m^2 cells and δ = 0.4m
- Architecture: ROS with decentralized execution
  - Robot controller with state [x,y,Θ] (attempts to meet deadline)
  - PID controller (corrects for heading error and drift)
- Robot simulator: V-REP
- Robots: iRobot Create2 robots

- Test environment: VICON MX Motion Capture System

Execution of MAPF Plans

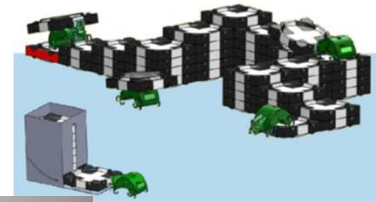
4-neighbor grid 8x

Planning for Delays

- Poster Presentation
  - Ma, Kumar, Koenig, MAPF with Delay Probabilities
  - Session "PS1: Planning," Monday 2:00-3:30, Plaza A
  - How to address delays with planning and execution monitoring rather than execution monitoring alone

Feasibility Study: TERMES Robots

- Consider the TERMES robots

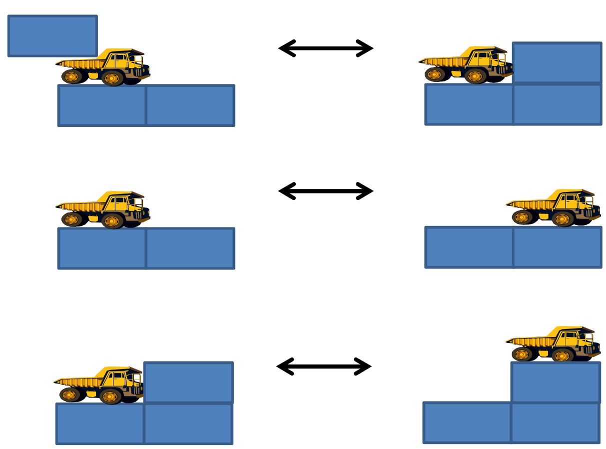


4-neighbor grid

[work by Harvard University, not me]

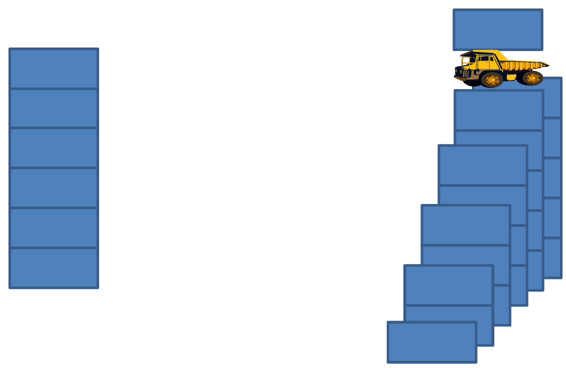
Feasibility Study: TERMES Robots

- Capabilities of the TERMES robots



Feasibility Study: TERMES Robots

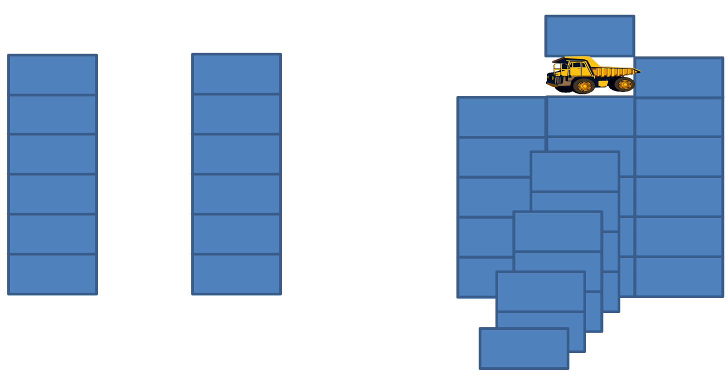
- Difficulty



4-neighbor grid

Feasibility Study: TERMES Robots

- Difficulty



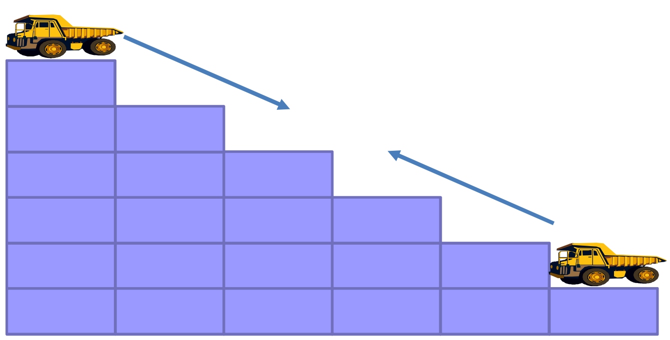
4-neighbor grid

Feasibility Study: TERMES Robots

- Difficulty
  - Behavior-based robotics does badly
  - General-purpose planning does badly
- We need a special-purpose planner for the construction task

Feasibility Study: TERMES Robots

- Two robots cannot pass each other on a ramp. Thus, one needs to solve a multi-robot path-planning problem



Single Robot Case

- Tree-based dynamic programming

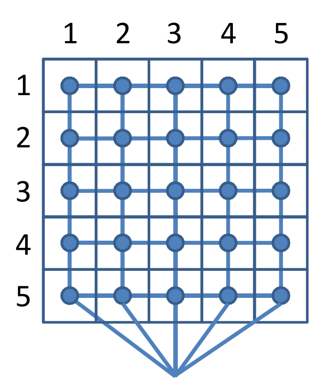
1	1	1	1	1
1				1
1		3		1
1				1
1	1	1	1	1

Tower by Tower (TBT) Method

4-neighbor grid

Single Robot Case

- Tree-based dynamic programming



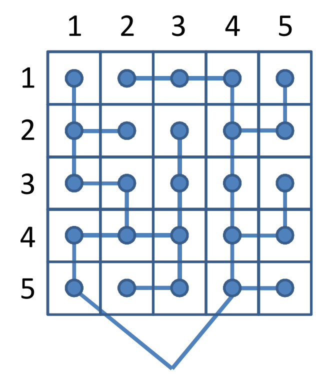
block reservoir (to get new blocks)

4-neighbor grid

85

Single Robot Case

- Tree-based dynamic programming

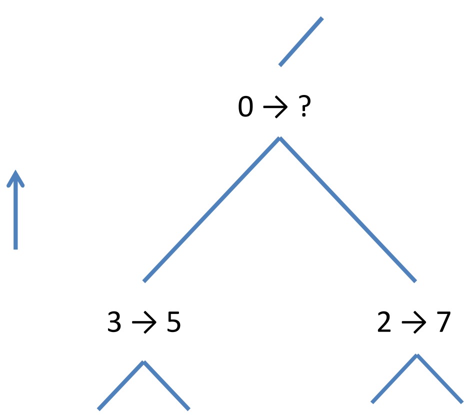


block reservoir (to get new blocks)

4-neighbor grid

Single Robot Case

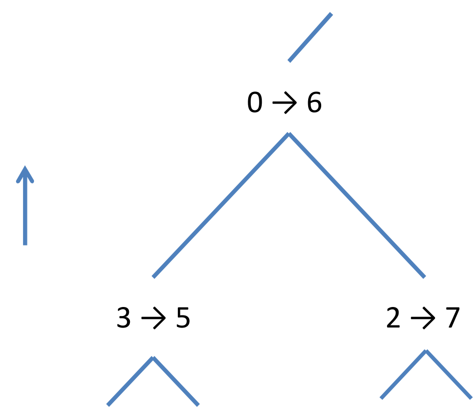
- Tree-based dynamic programming



86

Single Robot Case

- Tree-based dynamic programming



Single Robot Case

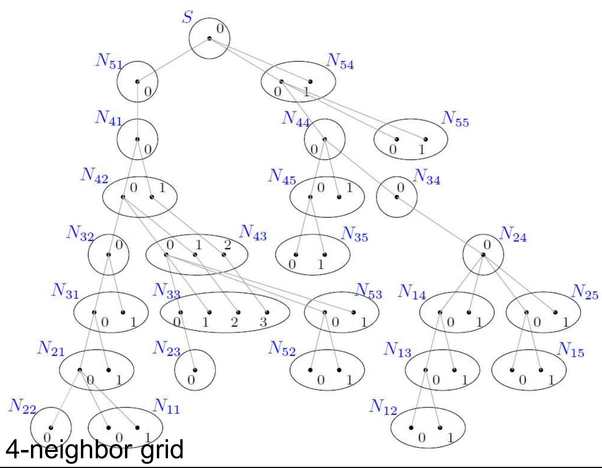
- Tree-based dynamic programming

1	1	1	1	1
1				1
1		3		1
1				1
1	1	1	1	1

4-neighbor grid

Single Robot Case

- Tree-based dynamic programming



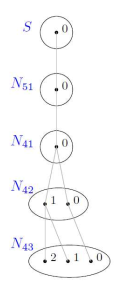
1	1	1	1	1
1				1
1		3		1
	1	2		1
	1	1	1	1

4-neighbor grid

89

Single Robot Case

- Tree-based dynamic programming



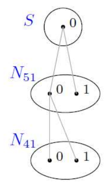
1	1	1	1	1
1				1
1		3		1
				1
	1	1	1	1

4-neighbor grid

90

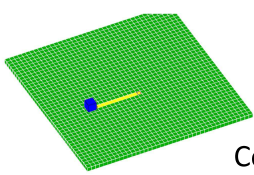
Single Robot Case

- Tree-based dynamic programming

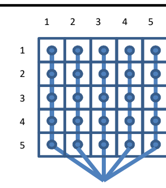


1	1	1	1	1
1				1
1		3		1
1				1
1	1	1	1	1

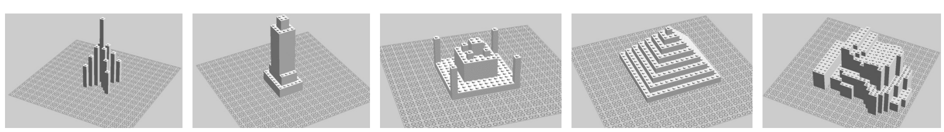
4-neighbor grid



Single Robot Case



Computation time: < 5 seconds



Building Model	Matrix	Max H	TBT	RBR	MST	RMST
Eiffel Tower	7×7	15	845	845	781	509
Empire State	6×8	15	3152	932	450	476
Taj Mahal	12×12	6	896	384	352	350
Giza Pyramid	15×15	8	2752	680	680	680
Disney Hall	22×16	10	11091	2245	1493	1499

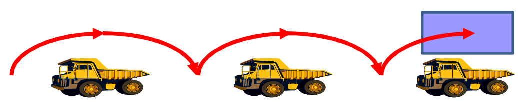
Number of block operations

TBT = Tower by Tower Method
RBR = Row by Row Method
MST = (Minimum) Spanning Tree Method
RMST = Reweighted (Minimum) Spanning Tree Method

4-neighbor grid

Multi-Robot Case

- Ongoing work
  - Spanning trees allow for parallelism since different robots might be able to work on different subtrees
  - Multiple robots can implement strategies that single robots cannot implement, for example, bucket brigades



93

Publications: Multi-Agent Path Finding

- H. Ma, S. Kumar and S. Koenig. Multi-Agent Path Finding with Delay Probabilities. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI),* 2017
- H. Ma, C. Tovey, G. Sharon, S. Kumar and S. Koenig. Multi-Agent Path Finding with Payload Transfers and the Package-Exchange Robot-Routing Problem. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI),* 3166-3173, 2016
- L. Cohen, T. Uras, S. Kumar, H. Xu, N. Ayanian and S. Koenig. Improved Solvers for Bounded-Suboptimal Multi-Agent Path Finding. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI),* 3067-3074, 2016
- H. Ma and S. Koenig. Optimal Target Assignment and Path Finding for Teams of Agents. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS),* 1144-1152, 2016
- W. Hoenig, S. Kumar, L. Cohen, H. Ma, H. Xu, N. Ayanian and S. Koenig. Multi-Agent Path Finding with Kinematic Constraints. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS),* 477-485, 2016
- W. Hoenig, S. Kumar, H. Ma, S. Koenig and N. Ayanian. Formation Change for Robot Groups in Occluded Environments. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS),* 4836-4842, 2016
- H. Ma, S. Koenig, N. Ayanian, L. Cohen, W. Hoenig, S. Kumar, T. Uras, H. Xu, C. Tovey and G. Sharon. Overview: Generalizations of Multi-Agent Path Finding to Real-World Scenarios. In Proceedings of IJCAI-16 Workshop on Multi-Agent Path Finding, 2016
- G. Sharon, R. Stern, A. Felner and N. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence* 219:40-66, 2015.
- L. Cohen, T. Uras and S. Koenig. Feasibility Study: Using Highways for Bounded-Suboptimal Multi-Agent Path Finding. In *Proceedings of the Symposium on Combinatorial Search (SOCS),* 2-8, 2015
- M. Cirillo, T. Uras and S. Koenig. A Lattice-Based Approach to Multi-Robot Motion Planning for Non-Holonomic Vehicles. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS),* 232-239, 2014
- M. Cirillo, F. Pecora, H. Andreasson, T. Uras and S. Koenig. Integrated Motion Planning and Coordination for Industrial Vehicles. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS),* 463-471, 2014

94

Publications: Multi-Agent Path Finding

- J. Yu and S. LaValle. Planning optimal paths for multiple robots on graphs. In Proceedings of the *IEEE International Conference on Robotics and Automation (ICRA),* 3612-3617, 2013
- J. Yu and S. LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI),* 1444-1449, 2013
- M. Phillips, B. Cohen, S. Chitta and M. Likhachev. In *Proceedings of the Robotics: Science and Systems Conference (RSS),* 2012.
- P. Wurman, R. D'Andrea and M. Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine* 29(1):9-20, 2008.

see http://idm-lab.org/project-p.html for more information

95

Publications: TERMES Robots

- T. Cai, D. Zhang, S. Kumar, S. Koenig and N. Ayanian. Local Search on Trees and a Framework for Automated Construction Using Multiple Identical Robots [Short Paper]. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS),* pages 1301-1302, 2016
- S. Kumar, S. Jung and S. Koenig. A Tree-Based Algorithm for Construction Robots. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS),* 2014
- K. Petersen, R. Nagpal and J. Werfel. Termes: An autonomous robotics system for three-dimensional collective construction. In *Proceedings of the Robotics: Science and Systems Conference (RSS),* 2011.

96

Conclusions

- The research on multi-agent path finding is joint work with H. Andreasson, N. Ayanian, M. Cirillo, L. Cohen, W. Hoenig, S. Kumar, H. Ma, F. Pecora, G. Sharon, C. Tovey, T. Uras and H. Xu
- The research on planning for the TERMES robots is joint work with T. Cai, S. Jung, S. Kumar and D. Zhang
- Thank you for listening!

- Funded in part by ARO, NASA, NSF and ONR
  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.

- For more information,
  see idm-lab.org and click on "projects"
  or send me an email: skoenig@usc.edu

97