Load Balancing in Distributed Multi-Agent Path Finder (DMAPF)

Poom Pianpak[1], Jiaoyang Li[2], Tran Cao Son[1]

1. New Mexico State University
2. Carnegie Mellon University

EMAS 2023

May 29th, 2023

Multi-Agent Path Finding (MAPF)

Given

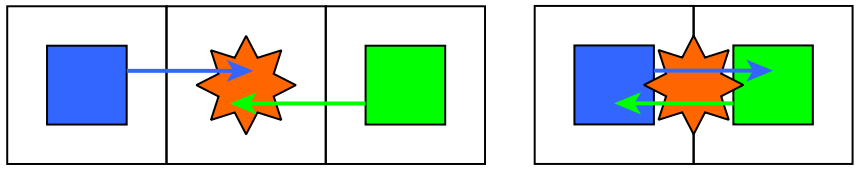
- A map (graph): vertices & edges
- Agents
  - Start locations
  - Goal locations

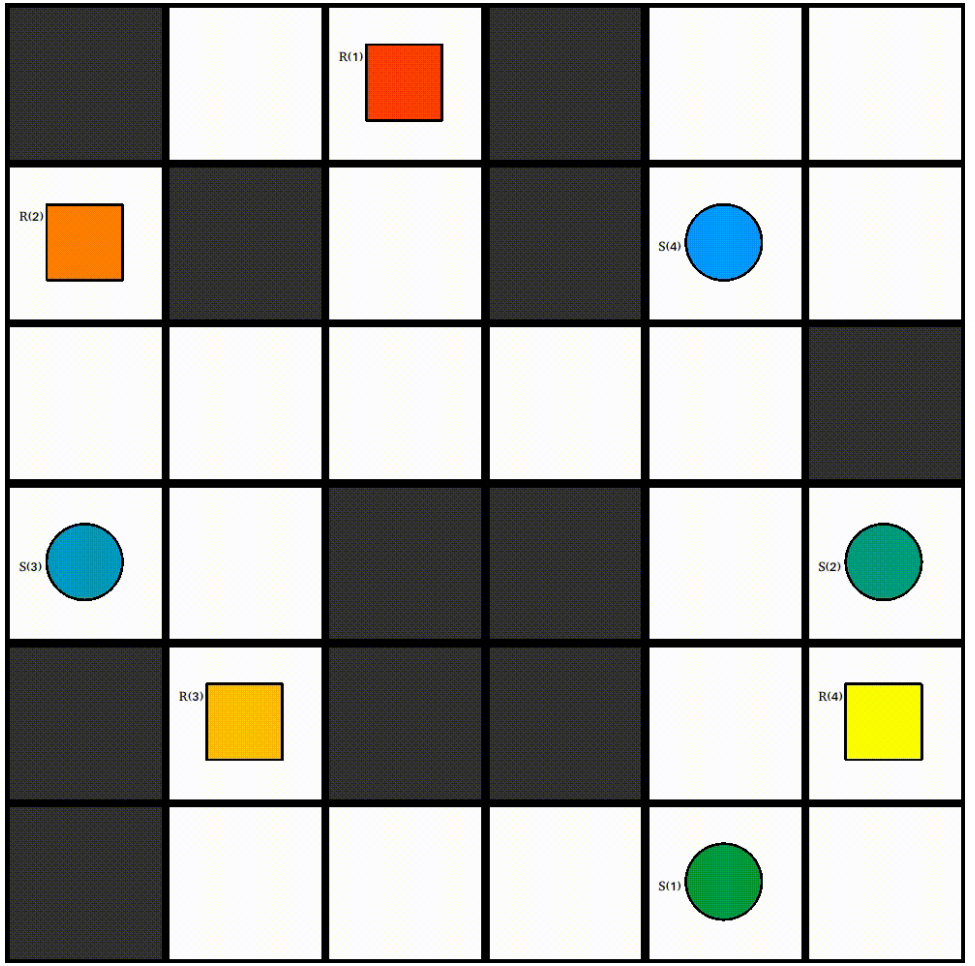
Find

- A sequence of actions that brings agents from to without conflicts.

Move

Wait





Applications of MAPF

Since 1996

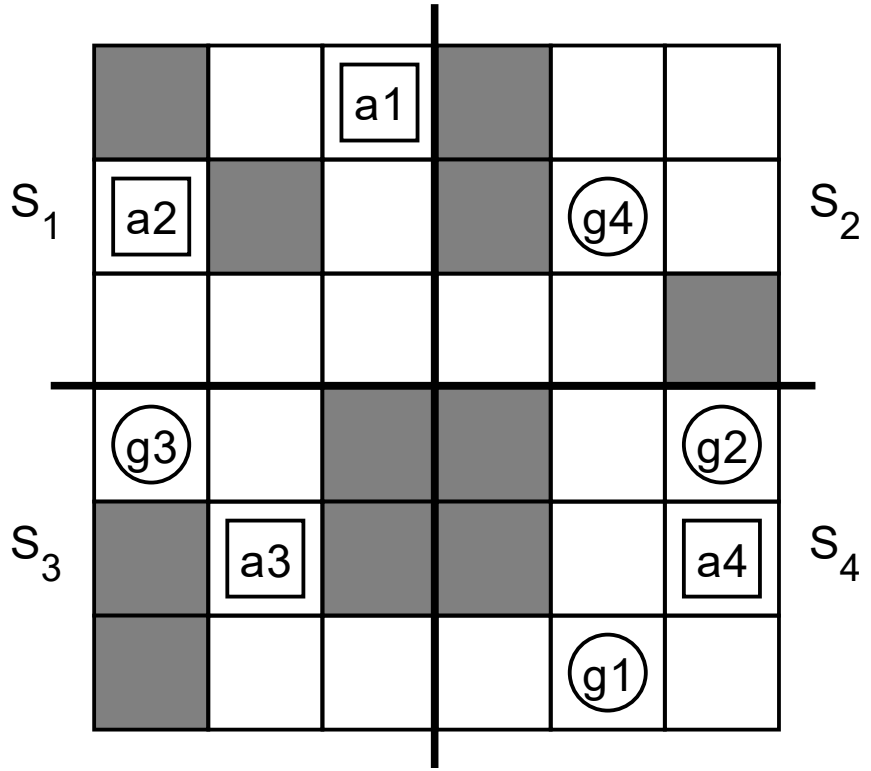
Since 2000s

The industries keep growing!

Distributed Multi-Agent Path Finder (DMAPF)

Idea

1. Partition a given map into subproblems S_1, …, S_n.
2. Assign each set of subproblems to a solving process.
3. Solving processes work together to solve the problem.



For simplicity, we are going to assume that each of the subproblems S_1, …, S_4 is assigned to a distinct solving process.

How Solving Processes Work Together

A sequence of areas from start to goal

First

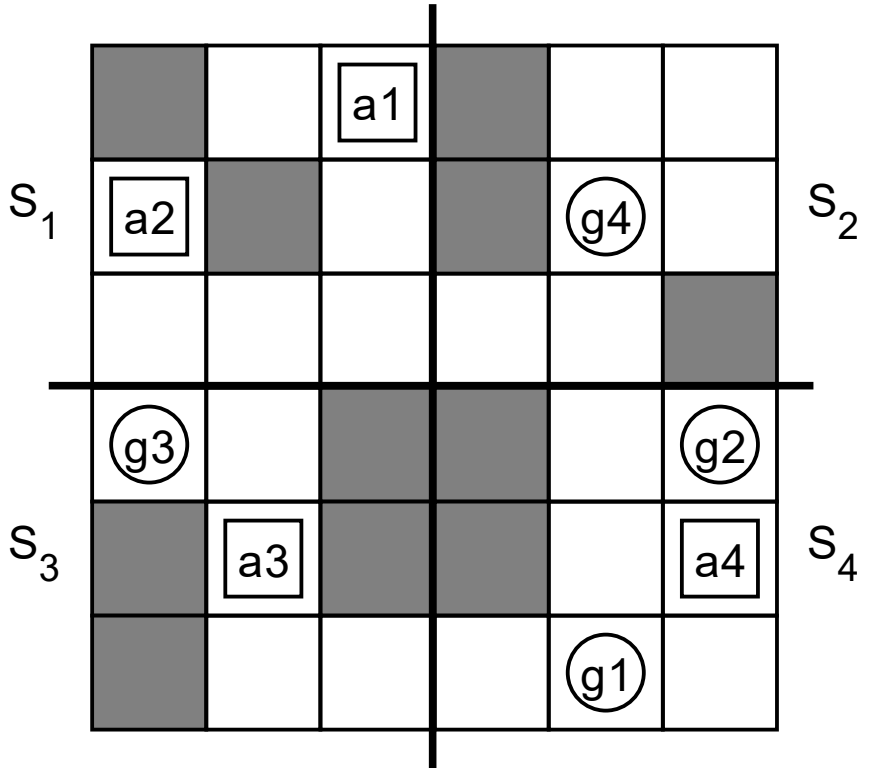
Make an abstract plan for each agent.

e.g.,

- a1: <S_1, S_2, S_4>
- a2: <S_1, S_3, S_4>
- a3: <S_3>
- a4: <S_4, S_2>

Answer Set Programming (ASP)

```
#external q(i).
1 {r(A2,i) : l(A1,A2)} 1 :- r(A1,i-1).
:- q(i), not r(A,i), g(A).
#show r/2.
```

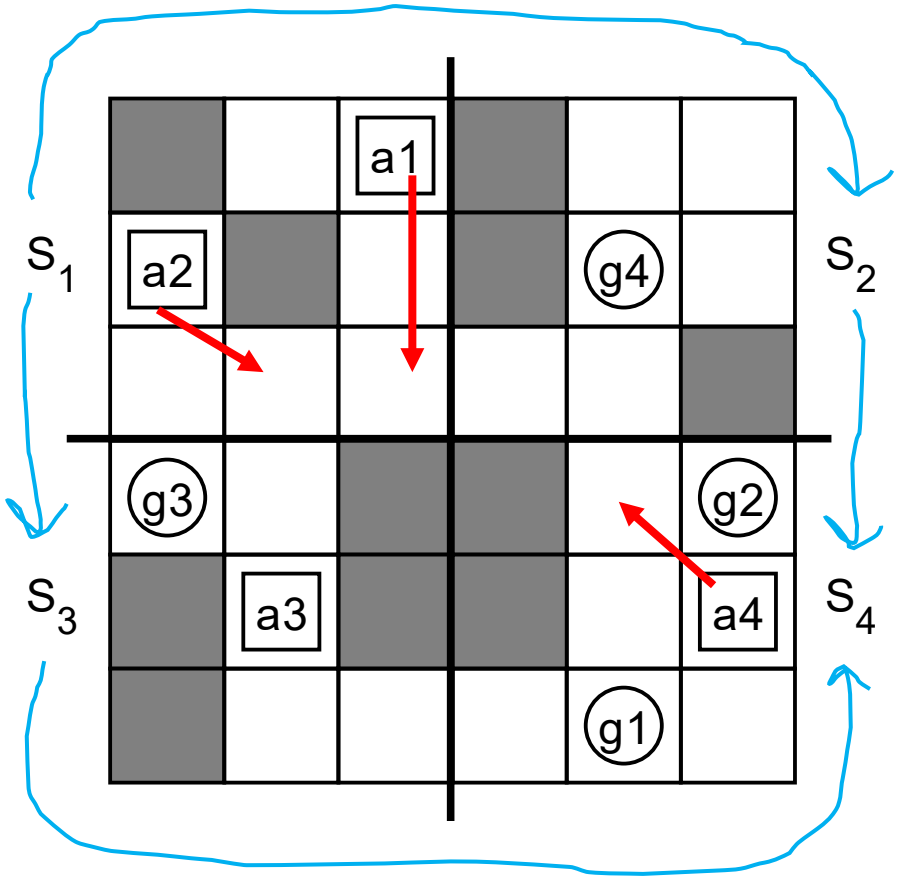


How Solving Processes Work Together

Then

Follow the 3-step protocol:

1. Negotiation
   - Decides which agent to migrate, and to which border location.
   - Prioritize agents with higher number of remaining steps in the abstract plans.

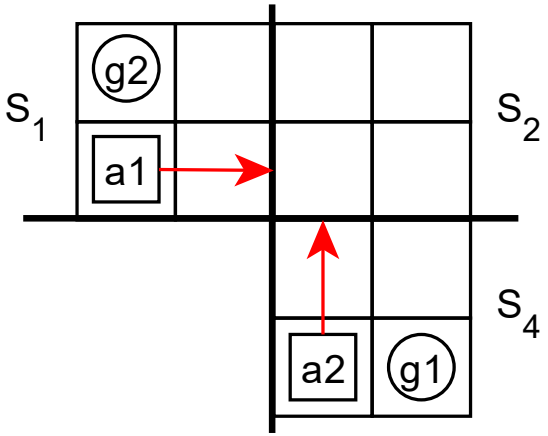


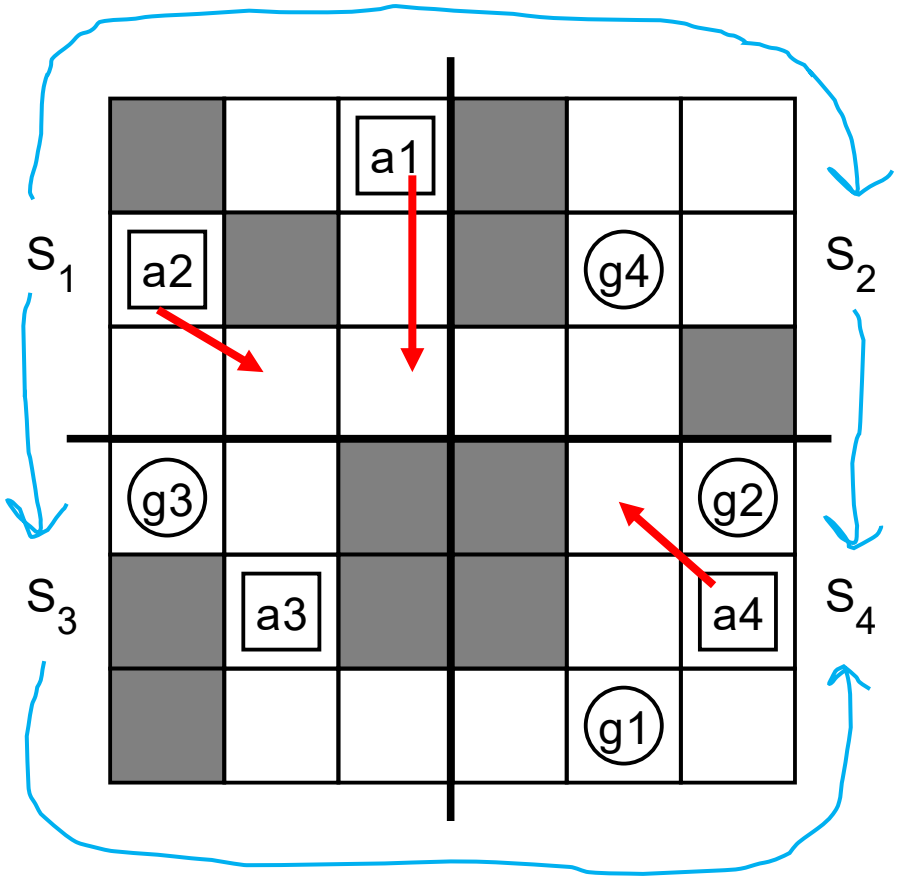
How Solving Processes Work Together

Then

Follow the 3-step protocol:

1. Negotiation
2. Rejection
   - Prevents possible conflicts from the negotiation.
   - Regulate the area.





How Solving Processes Work Together

Then

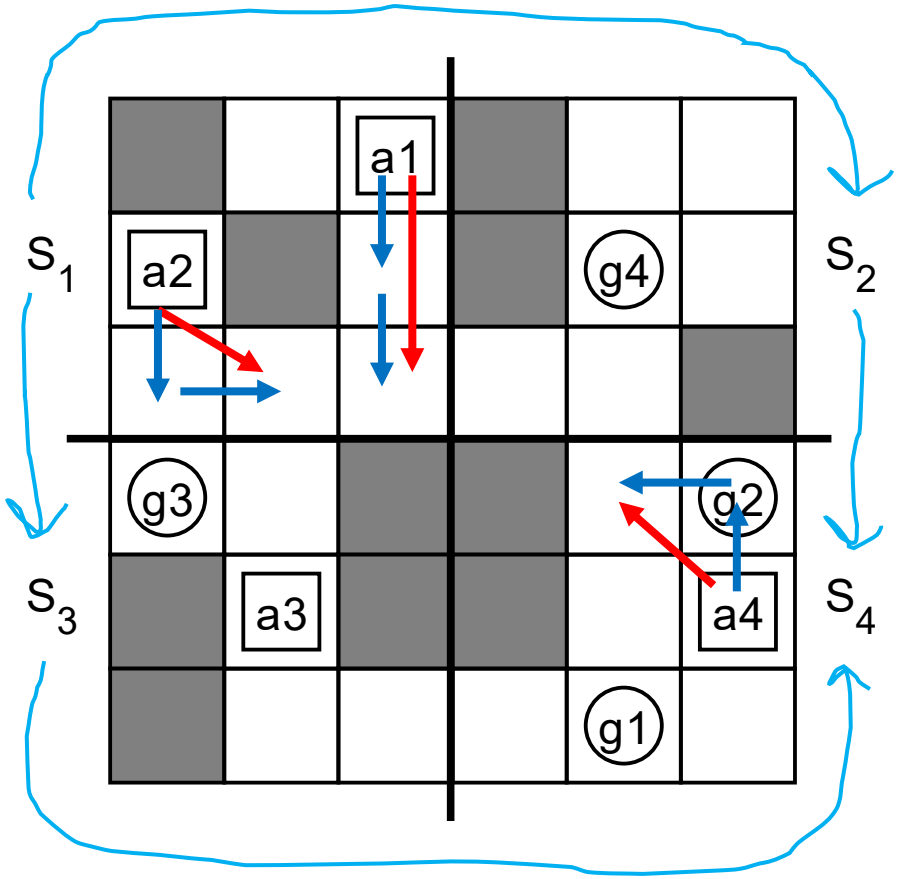
Follow the 3-step protocol:

1. Negotiation
2. Rejection

- Solving MAPF + Relaxation.

↓

Canceling some migration



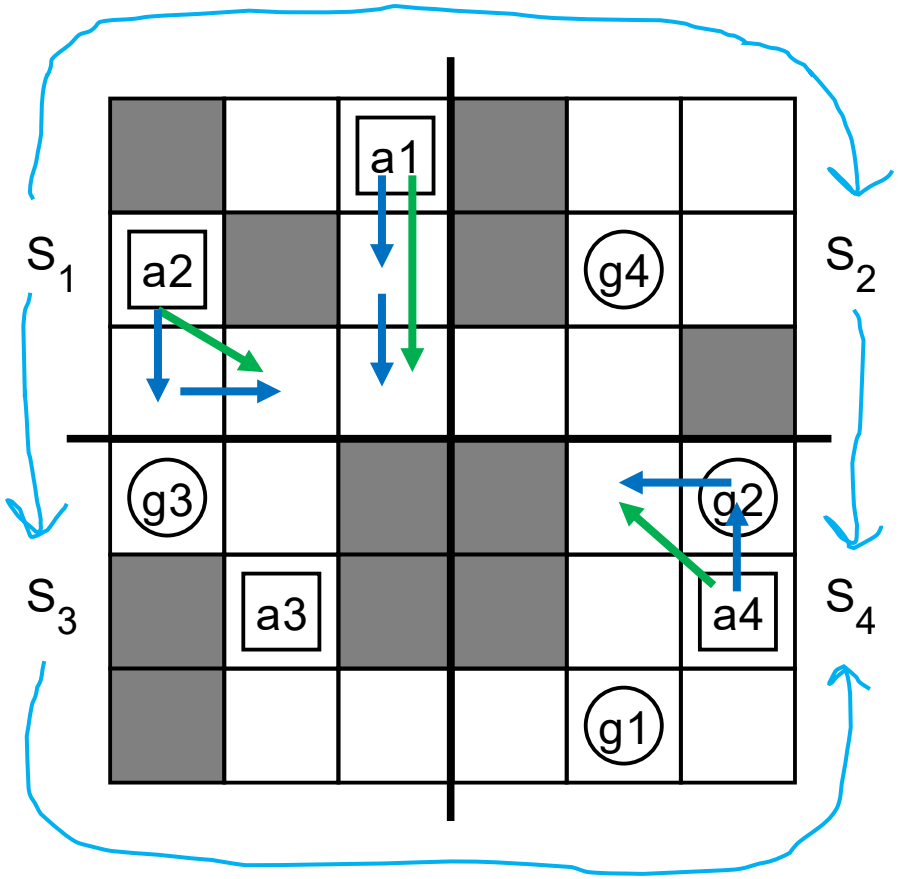
How Solving Processes Work Together

Then

Follow the 3-step protocol:

1. Negotiation
2. Rejection
   - Solving MAPF + Relaxation.
3. Confirmation
   - Confirms agents that can actually migrate.

Loop with new start locations.

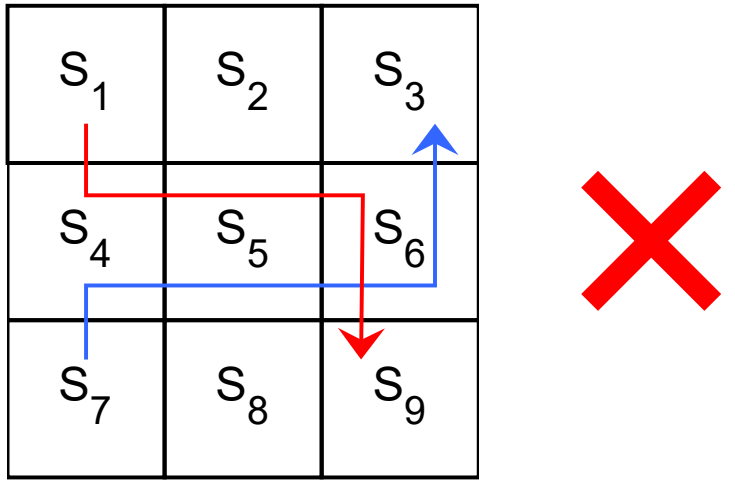


Reducing Congestion in Abstract Level

Observation

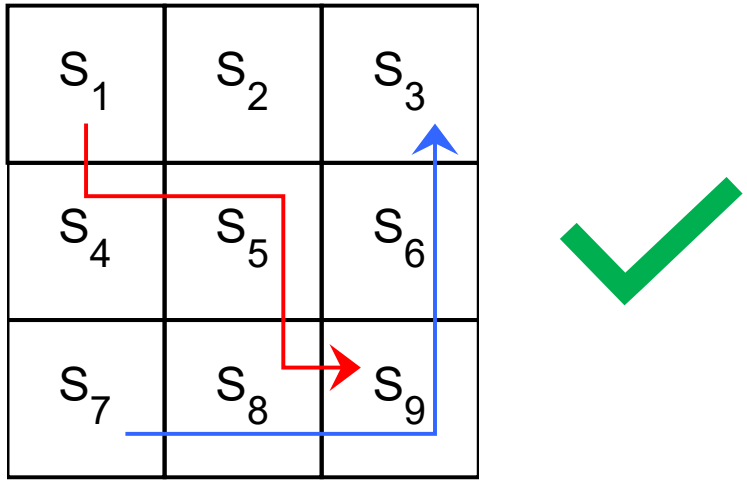
- Using blind search tends to cause congestion in some area.

#agents/#vertices



Idea

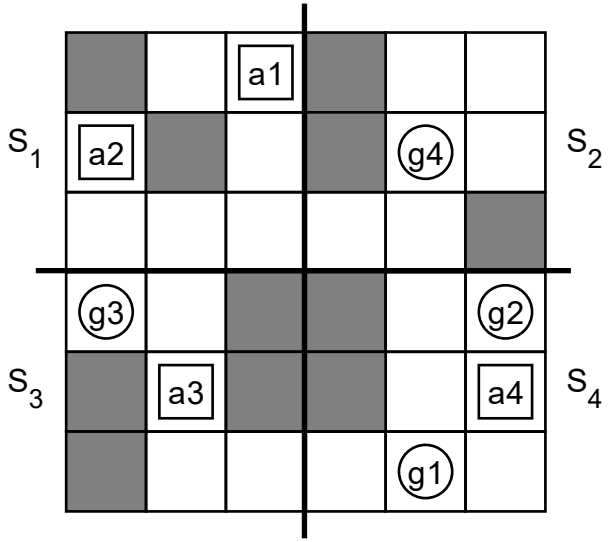
- Search while taking congestion into account.



Uniform-Cost Search on Congestion Matrix

Abstract plans:

- N/A



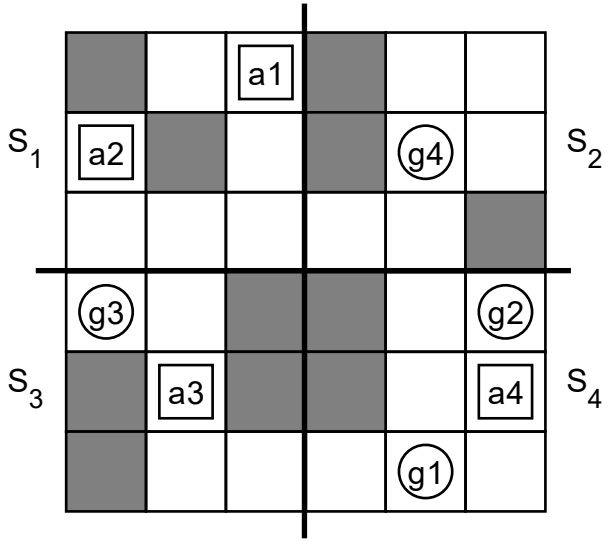
t = 0

S_1	0/7
S_2	0/6
S_3	0/5
S_4	0/7

Uniform-Cost Search on Congestion Matrix

Abstract plans:

- a1: <S_1, S_2, S_4>



t = 0

S_1	0/7
S_2	0/6
S_3	0/5
S_4	0/7

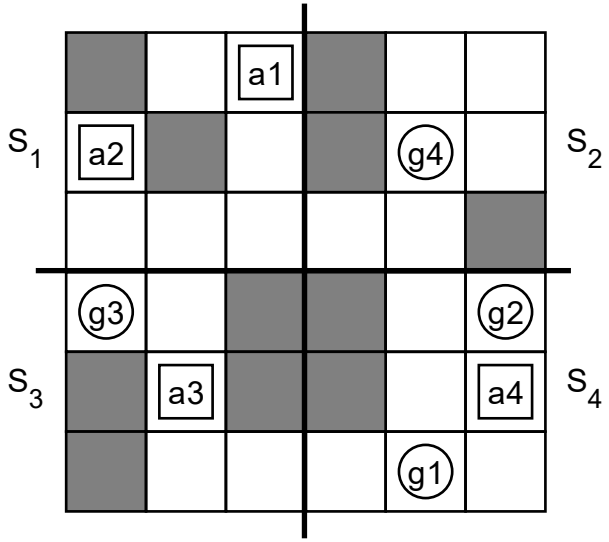
t = 0 1 2

S_1	1/7	0/7	0/7
S_2	0/6	1/6	0/6
S_3	0/5	0/5	0/5
S_4	0/7	0/7	1/7

Uniform-Cost Search on Congestion Matrix

Abstract plans:

- a1: <S_1, S_2, S_4>
- a2: <S_1, S_3, S_4>



t = 0

S_1	0/7
S_2	0/6
S_3	0/5
S_4	0/7

t = 0 1 2

S_1	1/7	0/7	0/7
S_2	0/6	1/6	0/6
S_3	0/5	0/5	0/5
S_4	0/7	0/7	1/7

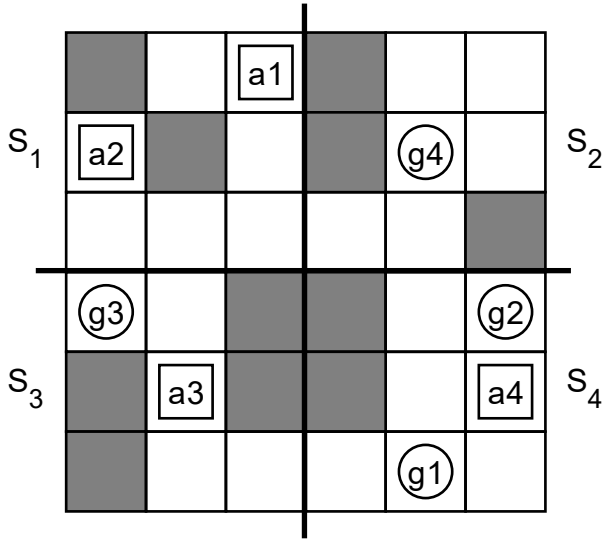
t = 0 1 2

S_1	2/7	0/7	0/7
S_2	0/6	1/6	0/6
S_3	0/5	1/5	0/5
S_4	0/7	0/7	2/7

Uniform-Cost Search on Congestion Matrix

Abstract plans:

- a1: <S_1, S_2, S_4>
- a2: <S_1, S_3, S_4>
- a3: <S_3>



	t = 0
S_1	0/7
S_2	0/6
S_3	0/5
S_4	0/7

	t = 0	1	2
S_1	1/7	0/7	0/7
S_2	0/6	1/6	0/6
S_3	0/5	0/5	0/5
S_4	0/7	0/7	1/7

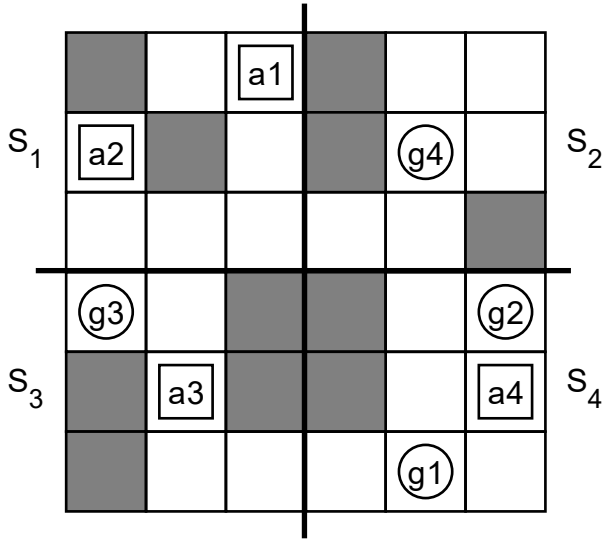
	t = 0	1	2
S_1	2/7	0/7	0/7
S_2	0/6	1/6	0/6
S_3	0/5	1/5	0/5
S_4	0/7	0/7	2/7

	t = 0	1	2
S_1	2/7	0/7	0/7
S_2	0/6	1/6	0/6
S_3	1/5	2/5	1/5
S_4	0/7	0/7	2/7

Uniform-Cost Search on Congestion Matrix

Abstract plans:

- a1: <S_1, S_2, S_4>
- a2: <S_1, S_3, S_4>
- a3: <S_3>
- a4: <S_4, S_2>



	t = 0
S_1	0/7
S_2	0/6
S_3	0/5
S_4	0/7

	t = 0	1	2
S_1	1/7	0/7	0/7
S_2	0/6	1/6	0/6
S_3	0/5	0/5	0/5
S_4	0/7	0/7	1/7

	t = 0	1	2
S_1	2/7	0/7	0/7
S_2	0/6	1/6	0/6
S_3	0/5	1/5	0/5
S_4	0/7	0/7	2/7

	t = 0	1	2
S_1	2/7	0/7	0/7
S_2	0/6	1/6	0/6
S_3	1/5	2/5	1/5
S_4	0/7	0/7	2/7

	t = 0	1	2
S_1	2/7	0/7	0/7
S_2	0/6	2/6	1/6
S_3	1/5	2/5	1/5
S_4	1/7	0/7	2/7

Timeout Estimation Mechanism

Keep the time the problem is solved per agent; call this t_a.

- If the problem is solved within $n \cdot t_a \cdot \varepsilon$
  - If some agent reach its assigned location;
    - update $t_a \leftarrow t_s \,/\, n$
  - Otherwise;
    - update $t_a \leftarrow f \cdot t_a$
- Otherwise
  - If some migrating agent has a location assigned;
    - remove its assigned location.
  - Otherwise;
    - terminate.

n – the number of agents
ε – timeout tolerance factor
f – timeout penalty factor
t_s – solving time

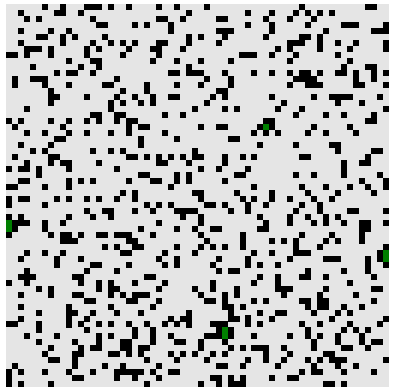
Exp 1. The Numbers of Solving Processes

lak303d
194x194,
|V|=14,784

n	Runtime (s)							
	$p = 4$	$p = 8$	$p = 12$	$p = 16$	$p = 20$	$p = 24$	$p = 28$	$p = 32$
200	32.1	25.1	21.8	20.0	19.4	**18.9**	20.5	19.6
400	97.0	75.1	63.1	66.2	**52.8**	56.2	53.8	63.5
600	214.2	158.7	129.3	127.5	**110.7**	113.3	116.0	120.0

Moving AI's Benchmark:
https://movingai.com/benchmarks/mapf/index.html

Exp 2. The Size of Subproblems



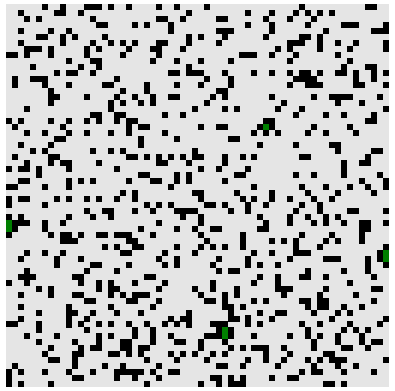
random-64-64-20
64x64, |V|=3,270

n = 1000

Makespan = The last time step where every agent reach its goal.
Sum-of-Cost (SoC) = The total arrival time.

v	Runtime (s)	Makespan	SoC ($\times 1k$)	Success Rate
30	**39.9**	**864**	**558.4**	0.4
40	46.6	906	598.5	**1.0**
50	88.2	1041	627.9	0.8
60	80.0	1107	676.6	**1.0**
70	86.8	1070	633.6	**1.0**

Exp 3. Timeout Sensitivity



random-64-64-20
64x64, |V|=3,270

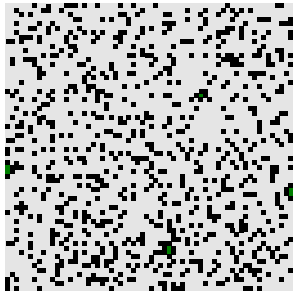
n = 1000

timeout tolerance factor

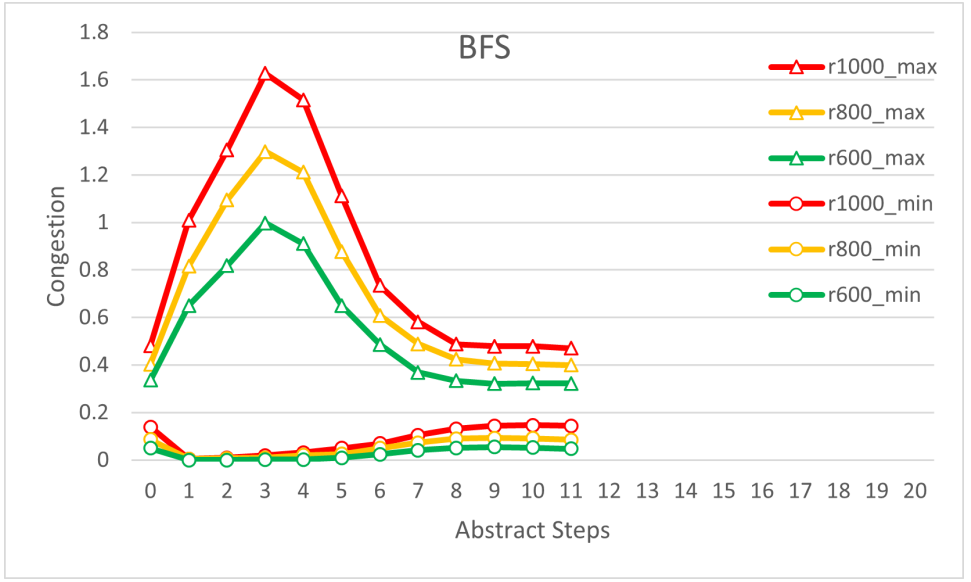
ϵ	Runtime (s)	Makespan	SoC ($\times 1k$)	#Stops
4	51.9	906	611.9	25
6	49.2	906	600.1	13
8	48.6	**903**	607.3	7
10	46.6	906	**598.5**	3
12	**46.3**	906	603.2	**1**
14	49.1	908	606.8	2

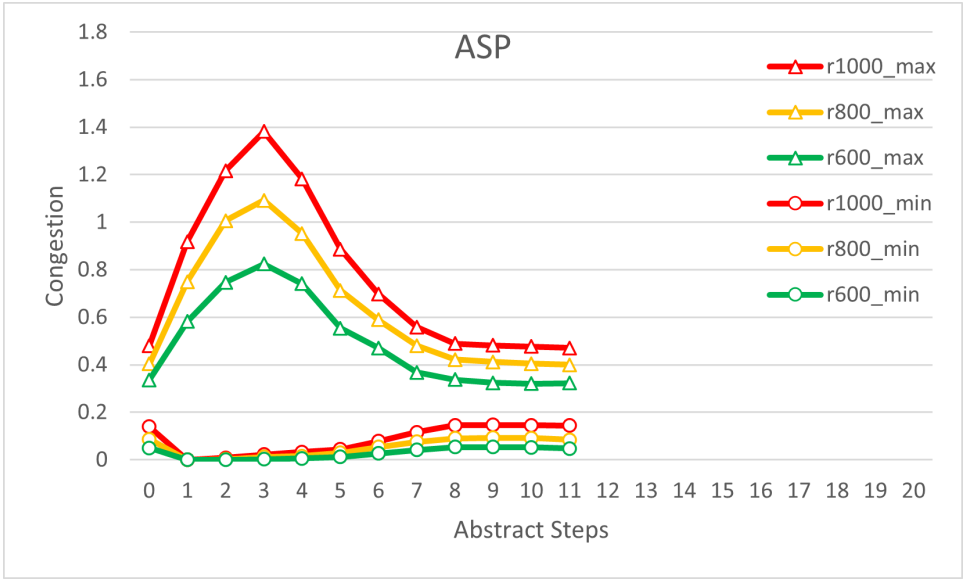
Exp 4. Congestion

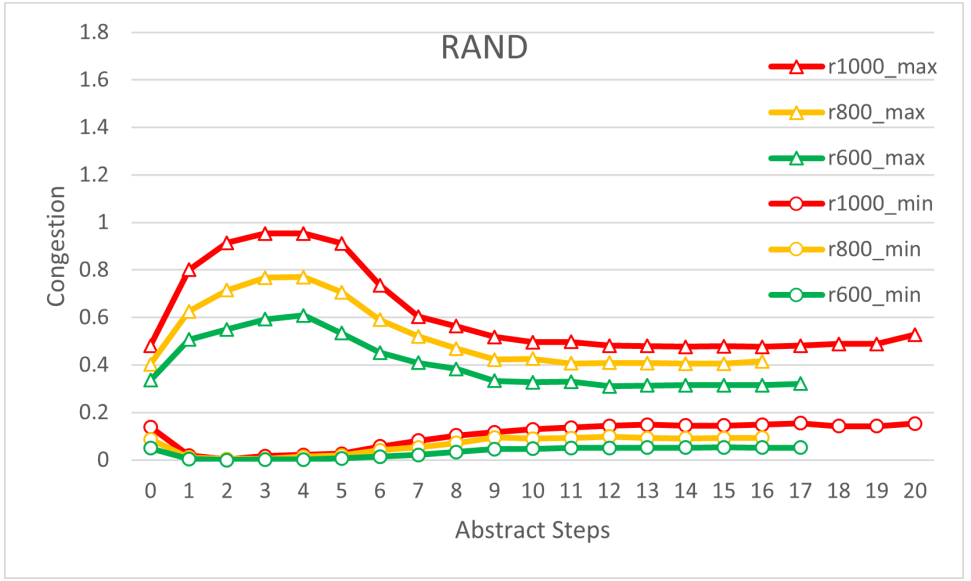
Blind searches…



random-64-64-20
64x64, |V|=3,270

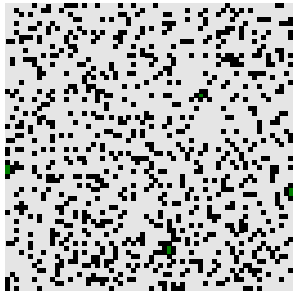




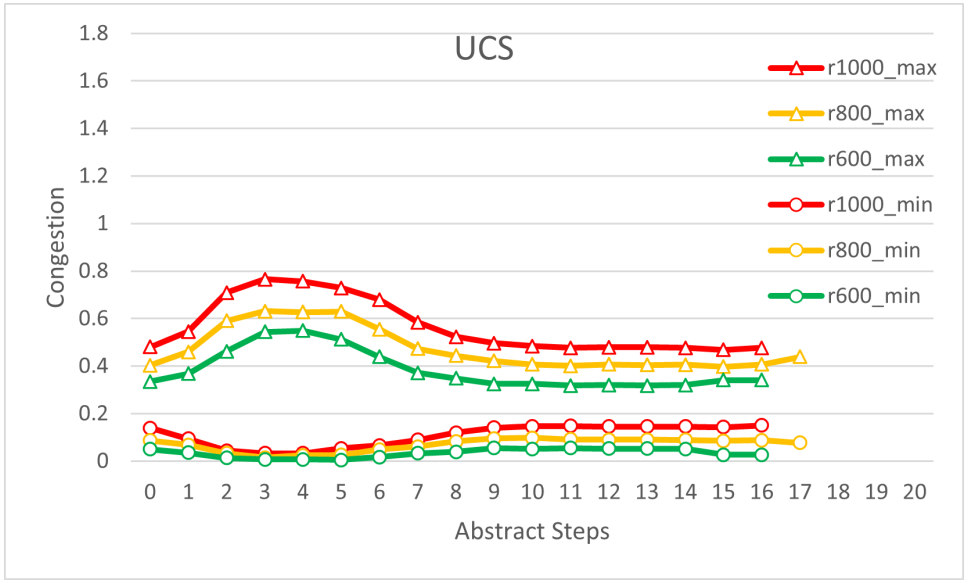


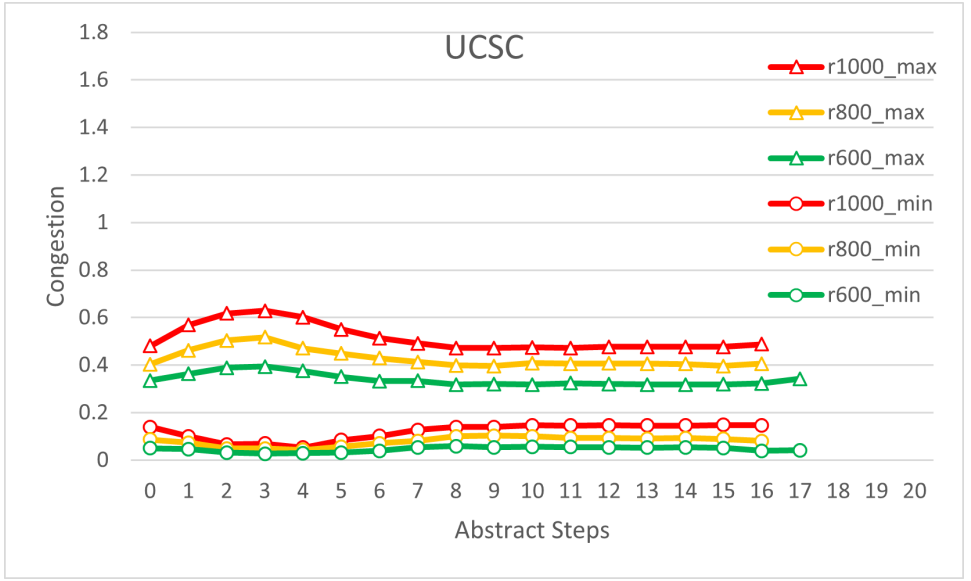
Exp 4. Congestion

Not-so-blind searches…

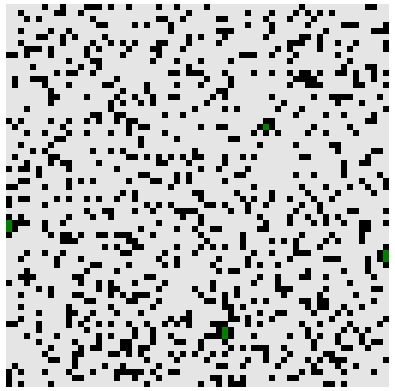


random-64-64-20
64x64, |V|=3,270





Exp 4. Congestion

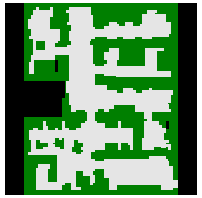
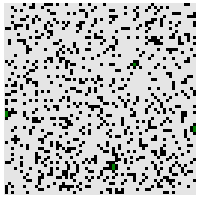
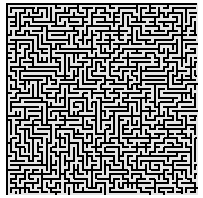


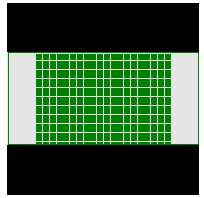
random-64-64-20
64x64, |V|=3,270

n	Method	Runtime (s)		Makespan	SoC ($\times 1k$)	Success Rate
		Abs.	Total			
600	ASP	0.3	48.4	914	247.3	**1.0**
	BFS	**0.0**	110.2	1185	265.6	0.8
	RAND	**0.0**	19.8	673	236.8	**1.0**
	UCS	**0.0**	28.2	779	258.4	**1.0**
	UCSC	**0.0**	**16.1**	**564**	**207.2**	**1.0**
800	ASP	0.4	106.6	1241	511.7	0.6
	BFS	**0.0**	162.1	1367	525.3	0.1
	RAND	**0.0**	74.2	893	424.8	0.8
	UCS	**0.0**	56.1	934	451.0	0.4
	UCSC	**0.0**	**26.1**	**757**	**379.1**	**1.0**
1000	ASP	0.5	-	-	-	0.0
	BFS	**0.0**	-	-	-	0.0
	RAND	**0.0**	204.6	1058	703.9	0.1
	UCS	**0.0**	103.8	1109	740.2	0.1
	UCSC	**0.0**	**46.6**	**906**	**598.5**	**1.0**

Exp 5. Comparisons

DMAPF
-A = with ASP; -C = with CBSH2-RTC
-E = with EECBS; -P = with PBS

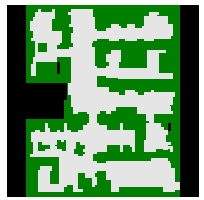
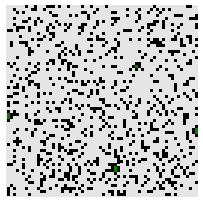
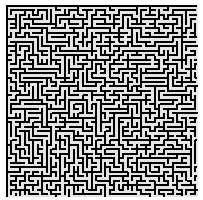




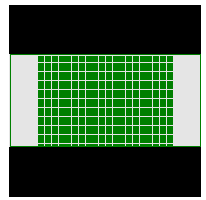


Solver	*den312d* (2445)			*random* (3270)			*maze* (10858)			*lak303d* (14784)			*warehouse* (38756)		
	200	300	400	600	800	1000	100	200	300	200	400	600	600	800	1000
	— Runtime (seconds) —														
DMAPF-A	6.9	18.4	39.3	16.1	26.0	**46.6**	20.7	**40.7**	-	15.6	37.9	75.9	32.5	42.4	52.3
DMAPF-C	-	-	-	-	-	-	8.5	-	-	35.7	-	-	14.2	19.8	35.2
DMAPF-E	170.6	-	-	-	-	-	9.8	-	-	10.7	-	-	14.8	20.4	62.9
DMAPF-P	3.1	-	-	-	-	-	8.1	-	-	7.5	44.2	-	13.1	17.0	**20.8**
EECBS	**0.4**	**1.4**	**6.4**	**2.5**	**21.4**	141.7	**2.6**	135.8	**279.6**	**1.2**	**6.5**	**42.6**	**5.8**	**13.1**	22.0
PBS	15.1	217.5	-	-	-	-	50.5	-	-	17.2	266.9	-	9.4	26.1	57.2
	— Success Rate —														
DMAPF-A	**1.0**	**1.0**	0.7	**1.0**	**1.0**	**1.0**	**1.0**	0.8	0.0	**1.0**	**1.0**	0.6	**1.0**	**1.0**	**1.0**
DMAPF-C	0.0	0.0	0.0	0.0	0.0	0.0	0.8	0.0	0.0	**1.0**	0.0	0.0	**1.0**	**1.0**	0.7
DMAPF-E	0.4	0.0	0.0	0.0	0.0	0.0	0.9	0.0	0.0	**1.0**	0.0	0.0	**1.0**	0.9	0.6
DMAPF-P	0.6	0.0	0.0	0.0	0.0	0.0	0.9	0.0	0.0	**1.0**	0.4	0.0	0.6	0.5	0.3
EECBS	**1.0**	**1.0**	**1.0**	**1.0**	0.9	**1.0**	0.9	**0.9**	**0.1**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
PBS	**1.0**	0.8	0.0	0.0	0.0	0.0	**1.0**	0.0	0.0	**1.0**	0.3	0.0	**1.0**	**1.0**	**1.0**

Exp 5. Comparisons

DMAPF
-A = with ASP; -C = with CBSH2-RTC
-E = with EECBS; -P = with PBS

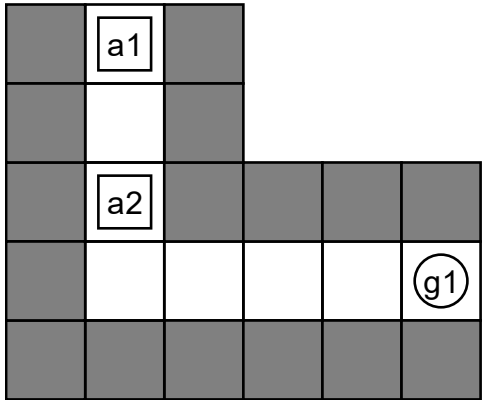


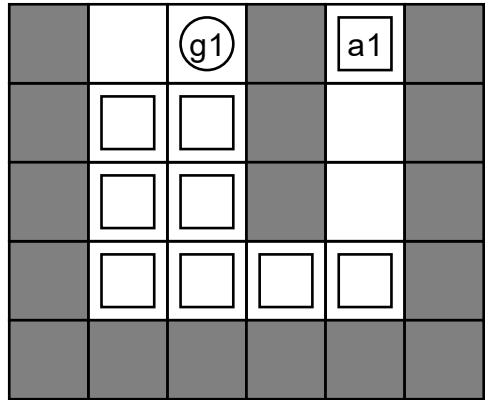
Solver	*den312d* (2445)			*random* (3270)			*maze* (10858)			*lak303d* (14784)			*warehouse* (38756)		
	200	300	400	600	800	1000	100	200	300	200	400	600	600	800	1000
	— Makespan —														
DMAPF-A	475	722	980	564	757	906	3075	3704	-	1014	1794	2774	748	774	795
DMAPF-C	-	-	-	-	-	-	3091	-	-	1017	-	-	753	781	804
DMAPF-E	643	-	-	-	-	-	3072	-	-	1033	-	-	780	828	893
DMAPF-P	477	-	-	-	-	-	3069	-	-	1016	1690	-	761	779	803
EECBS	180	288	**377**	**145**	**218**	**302**	**1474**	**1571**	**1702**	483	511	**583**	**451**	**455**	**457**
PBS	**132**	**158**	-	-	-	-	1475	-	-	**482**	**479**	-	**451**	**455**	**457**
	— Sum-of-Cost (×1000) —														
DMAPF-A	51.8	117.3	248.9	207.2	379.1	598.5	181.4	511.8	-	112.9	362.2	794.9	233.1	331.3	448.3
DMAPF-C	-	-	-	-	-	-	187.2	-	-	110.7	-	-	230.7	335.8	443.2
DMAPF-E	56.4	-	-	-	-	-	175.4	-	-	110.4	-	-	236.7	356.6	498.8
DMAPF-P	54.1	-	-	-	-	-	176.2	-	-	111.8	358.7	-	231.8	334.8	449.1
EECBS	13.8	28.0	**46.9**	**34.8**	**60.6**	**101.6**	**56.1**	**119.7**	**191.4**	38.2	78.6	**131.1**	109.6	146.4	181.1
PBS	**11.6**	**19.1**	-	-	-	-	56.4	-	-	**37.9**	**74.1**	-	**109.5**	**146.2**	**180.9**

Remarks

Comparing with EECBS and PBS…

- DMAPF can find a solution faster in problems with a lot of agents.
- DMAPF has a trouble when the map is not decomposed nicely.





- The solution quality from DMAPF is about 2-6 times worse.