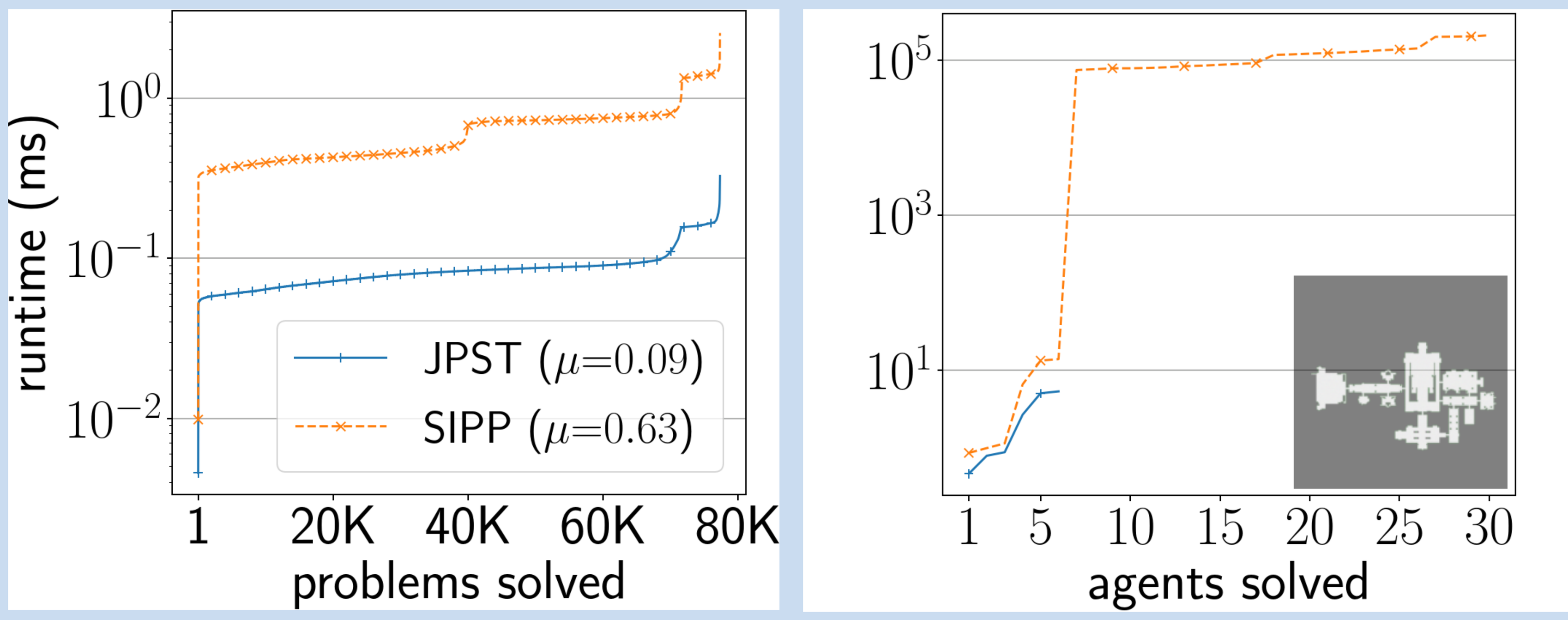
Multi-Agent Path Finding with Temporal Jump Point Search

Shuli Hu, Daniel D. Harabor, Graeme Gange, Peter J. Stuckey, Nathan R. Sturtevant

NORTHEAST NORMAL UNIVERSITY 东北师范大学 | MONASH University | UNIVERSITY OF ALBERTA

Motivation

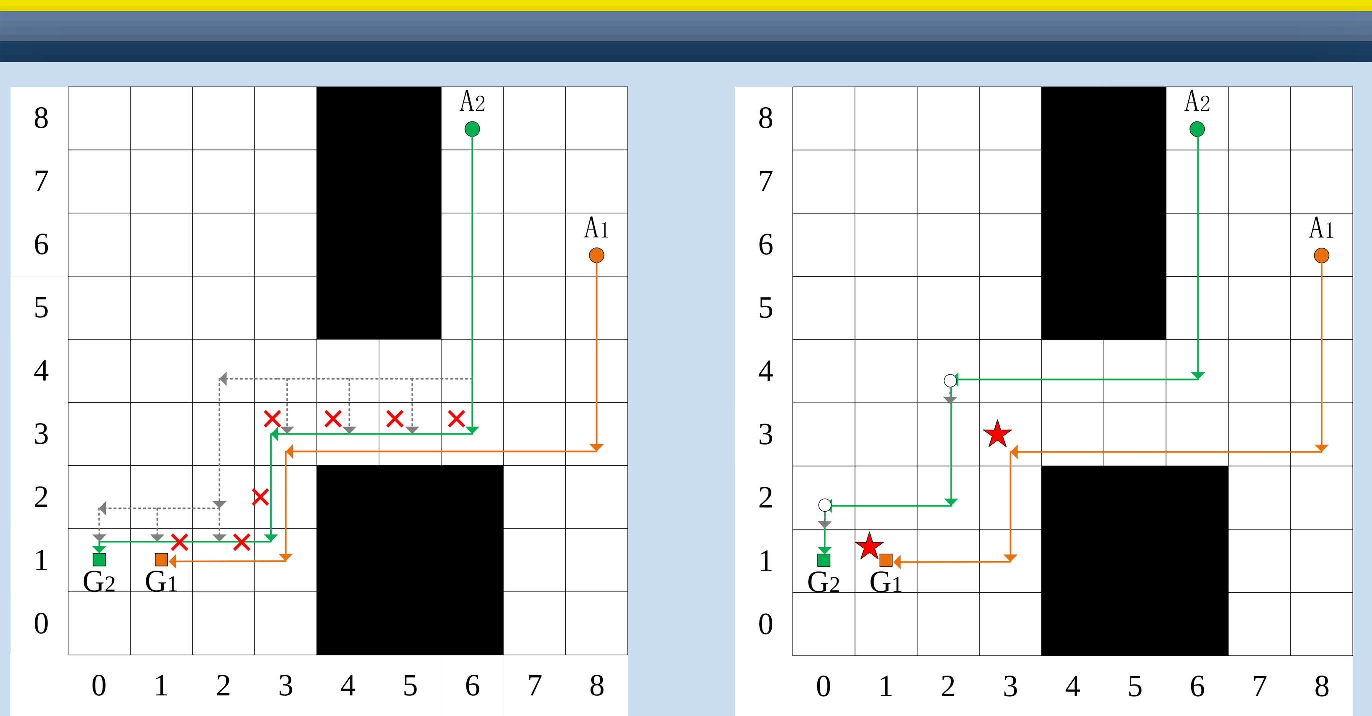
Multi-Agent Path Finding (MAPF) is a combinatorial planning problem that asks us to coordinate a team of moving agents, which can be applied in various industrial scenarios.
Temporal Jump Point Search (JPST) is proposed for optimal grid-based pathfinding with temporal obstacles. JPST finds the canonical paths, and it's shown to be more than one order of magnitude faster than safe interval path planning (SIPP).
However, when JPST is applied in MAPF conflict-based search (CBS) algorithm, it doesn't get the similar performance gains. In Figure 1a, JPST is far superior, but in Figure 1b, CBS with JPST fails to find any solution after the first 6 agents. So, we plan to investigate what's happening.



(a) SAPF (b) MAPF

Figure 1: For Single-Agent Path Finding (SAPF), we create a CBS Conflict Tree and use JPST and SIPP to solve the path planning problem arising at each node. For MAPF, we use JPST and SIPP as the low-level planners in CBS, and add agents until CBS timeout (5 mins). We test on *lt_gallostemplar_n* map.

Canonical Paths in MAPF



(a) (b)

Figure 2: Examples of canonical replanning with JPST

In order to understand why JPST does worse than SIPP, we examine Figure 2a. Here we show the unique VHW-canonical paths for agents A_1 (orange) and A_2 (green). The paths conflict from (6, 3) until (1, 1). CBS branches on the earliest conflict, but the uniqueness of canonical path makes it harder to resolve collisions.

Our ideas

We consider several bypassing strategies to help us fruitfully combine CBS and JPST.

- Explicit Bypassing: To find an equally good non-canonical path, we add the conflicting jump points as the temporal obstacles. We prove that it is enough to create a canonical bypass path, if there exists. In Figure 2b, it's an example of explicit bypassing.
- Replanning End: We don't need to replan for the whole path, and we try three following segments instead. They all start from the previous jump point closest to the conflict v@t.
  (J): to the next jump point not in a straight line
  (M): to the furthest point which is Manhattan optimal from v@t
  (G): to the goal location
- Replanning Method:
  a) A* with CAT; b)SIPP with CAT; c)JPST

Experimental Results

Experiment #1: The max number of agents solved

JPST is much faster than A*, so it solves more problems overall, but it dose worse than SIPP. Note that canonical A* is also worse than A*, showing that canonical paths make collisions harder to resolve.

					JPST									
map	JPST	SIPP	A*	CA*	N	A-J	S-J	J-J	A-M	S-M	J-M	A-G	S-G	J-G
Berlin_1_256	54	54	54	54	54	54	54	54	54	54	54	54	54	54
Boston_0_256	59	56	52	58	44	59	59	69	59	59	77	78	77	**83**
brc202d	39	39	**40**	39	23	**40**	**40**	**40**	**40**	**40**	**40**	**40**	**40**	**40**
den312d	33	25	25	33	25	33	33	**36**	**36**	**36**	**36**	**36**	**36**	**36**
den520d	23	23	23	23	23	23	23	23	23	23	23	23	23	23
empty-16-16	26	35	35	28	18	30	30	28	**38**	**38**	37	**38**	**38**	37
empty-32-32	33	33	33	33	33	33	33	33	38	35	33	**45**	36	33
empty-48-48	56	56	56	56	53	56	56	56	56	56	56	56	56	56
empty-8-8	20	20	25	20	20	24	24	22	**26**	**26**	22	**26**	**26**	22
ht_chantry	27	**30**	27	27	27	**30**	**30**	27	**30**	**30**	27	**30**	**30**	**30**
ht_mansion_n	**49**	**49**	**49**	46	46	**49**	**49**	**49**	**49**	**49**	**49**	**49**	**49**	**49**
lak303d	25	25	25	25	25	25	25	25	25	25	25	**34**	**34**	**34**
lt_gallowstemplar_n	6	**30**	6	6	6	6	6	6	11	6	6	26	11	6
maze-128-128-10	19	21	21	19	19	19	19	19	19	19	19	**26**	**26**	19
maze-128-128-2	**18**	**18**	17	**18**	16	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**
maze-32-32-2	16	16	16	16	14	**17**	**17**	**17**	**17**	**17**	**17**	**17**	**17**	**17**
maze-32-32-4	**25**	22	**25**	24	20	22	22	**25**	**25**	**25**	**25**	**25**	**25**	**25**
orz900d	39	**41**	24	24	23	40	39	40	**41**	**41**	**41**	27	40	**41**
ost003d	23	23	23	23	23	23	23	23	23	23	23	23	23	23
Paris_1_256	81	67	67	67	72	98	98	98	**118**	**118**	**118**	**118**	**118**	**118**
random-32-32-10	42	52	47	42	38	43	43	42	**54**	53	43	**54**	**54**	53
random-32-32-20	46	45	45	40	28	**47**	**47**	**47**	**47**	**47**	**47**	**47**	**47**	**47**
random-64-64-10	18	18	18	18	18	18	18	18	18	18	18	18	18	18
random-64-64-20	62	**66**	62	62	48	**66**	**66**	**66**	**66**	**66**	**66**	**66**	**66**	**66**
room-32-32-4	13	15	13	15	13	15	15	15	16	15	15	16	**19**	**19**
room-64-64-16	28	28	28	28	28	28	28	28	28	28	28	28	28	28
room-64-64-8	19	19	19	19	19	19	19	19	19	19	19	**20**	19	19
w_woundedcoast	29	35	29	37	29	29	29	29	**41**	40	40	**41**	40	40
Total solved agents	928	961	904	900	805	964	963	972	1035	1024	1022	**1079**	1068	1054

Table 1: The max number of agents solved by different algorithms on the MAPF benchmark problems.

The best algorithm overall is JPST-A*-G for bypass replanning from the previous jump point to the goal, which does slightly better than JPST-SIPP-G for bypass replanning from the previous jump point to the goal.

Experiment #2: Success rate

We present the success rate on two maps (one small, one large) using all 25 scenario files. From figure 3, we can observe that JPST variants are clearly better than baseline algorithms.

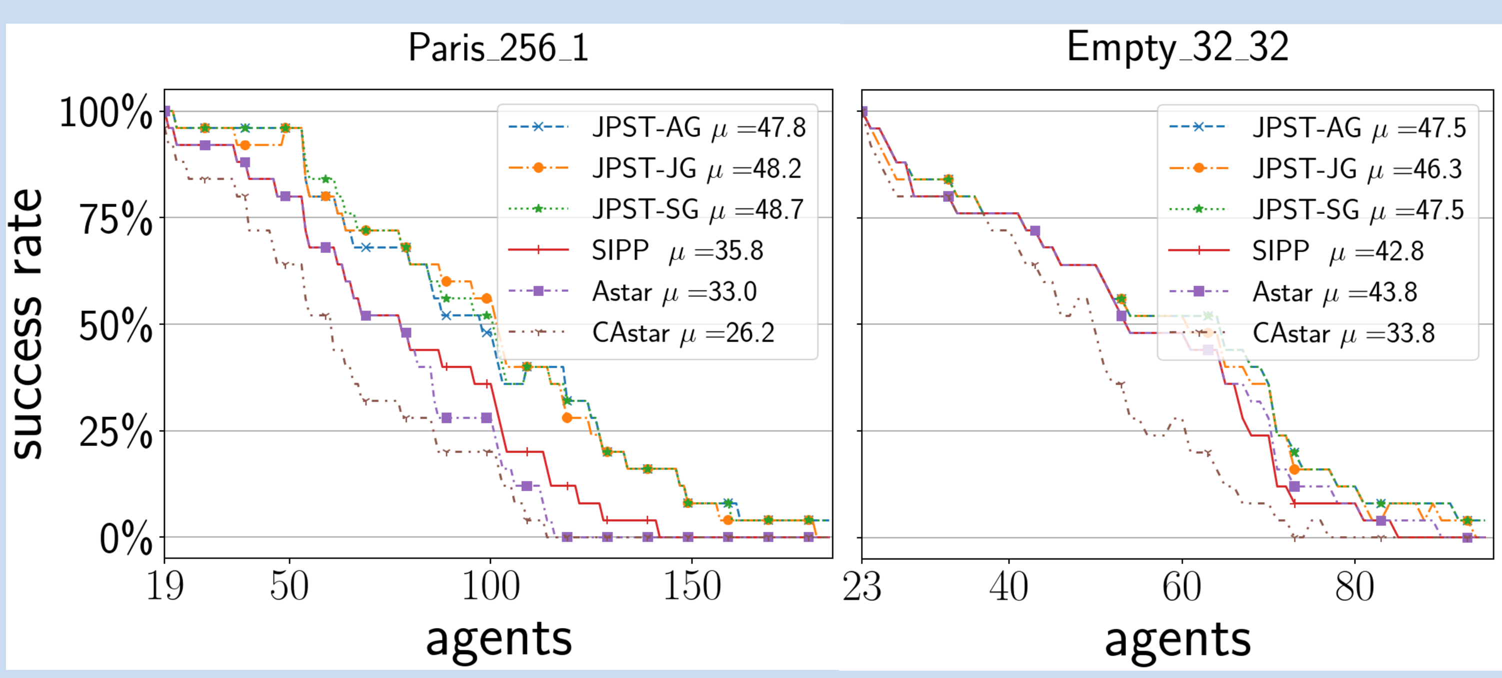


Figure 3: Success rates using 25 scenario files per map.