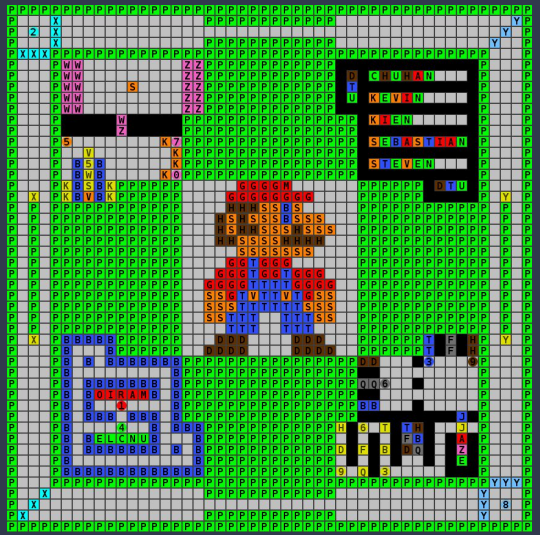
02285 Artificial Intelligence and Multi-Agent Systems

Programming project: Group 9

Group members:
Chuhan Zhou [s230357]
Kevin Ferneding [s242823]
Sebastian Giovanni Murgia Kristensen [s243884]
Steven Wang [s243874]
Trung Kien Luyen [s243416]

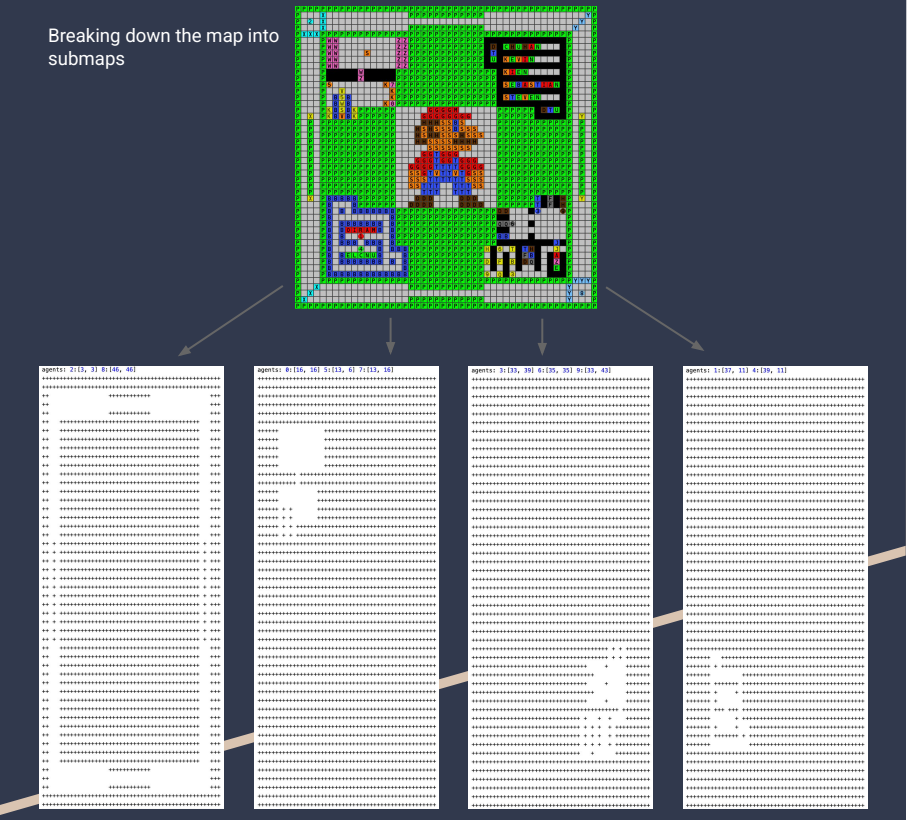
Minchia:
A fast, JPS-based algorithm for Multi-Agent Path Finding



Work division:
All group members had a more or less equal contribution to all aspects of the project

Overview

Breaking down the map into submaps



Map analysis: Break down to submaps

Find optimal assignments: Create box-goal assignment

Create goal graph

Create proposals for currently solvable goals & pick best

Create conflict-free plan or (if conflict occurs) create subgoal to be solved first

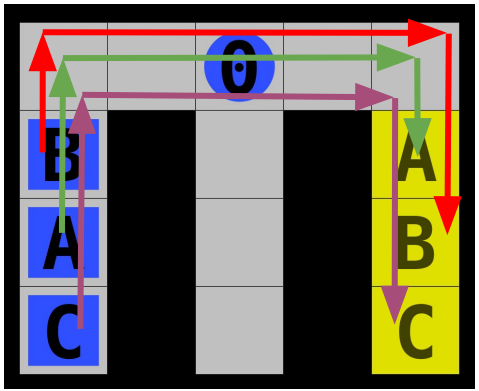
Goal Graph: Concept

Idea: A graph that expresses dependencies between goals

During box-goal assignment, we retrieve optimal paths

Check if any other goals are on that path

- If yes, create dependency from passed goal to current goal



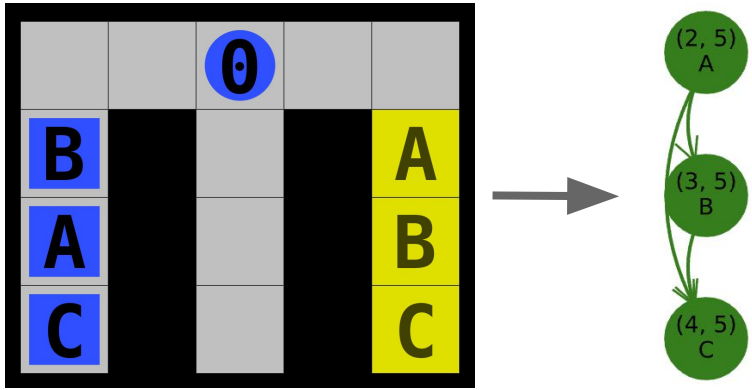
- On path for B we pass through A
- On path for A we pass through no other goal
- On path for C we pass through A and B

Goal Graph: Concept

On path for B we pass through A
⇒ **A depends on B being solved first**

On path for A we pass through none
⇒ **No other goal has to wait until A is solved**

On path for C we pass through A and B
⇒ **A and B depend on C being solved first**

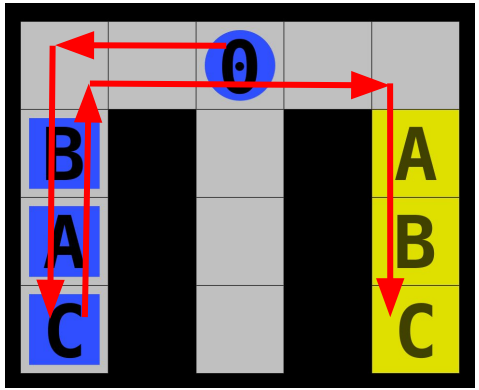


Goals with outgoing edges cannot be solved until the linked goals are solved

⇒ Always only try to solve goals without outgoing edges

Goal Graph: Subgoals

Try to solve C:

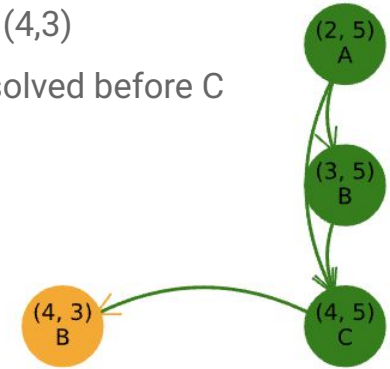


First conflict: Box B

⇒ Find a storage position and create a subgoal:

- B has to be moved to (4,3)
- Subgoal B has to be solved before C

  ⇒ Add dependency

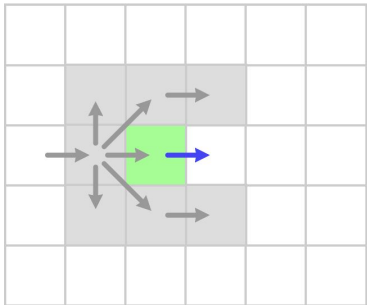


Jump Points: Basics

- Fast path exploration by exploiting logical relationships of neighbor cells to prune redundant paths

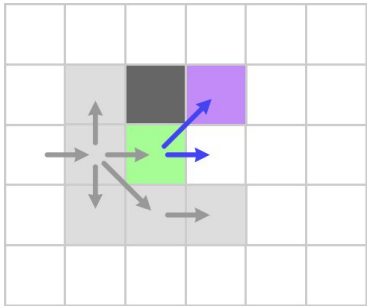
Vertical / horizontal moves cost 1,
diagonal moves cost √2

⇒Only straight moves are explored, since other cells could be explored faster by using a previous diagonal step



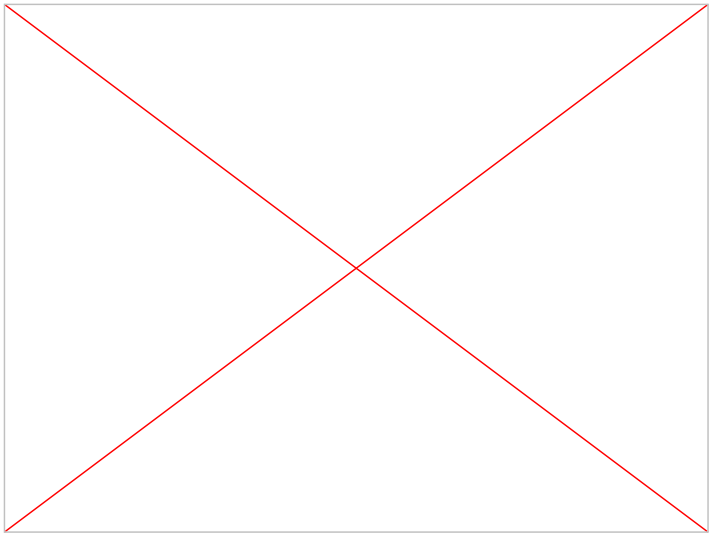
If a neighbor is a wall, our assumption breaks

⇒We add the current position as a jump point and continue in our queue

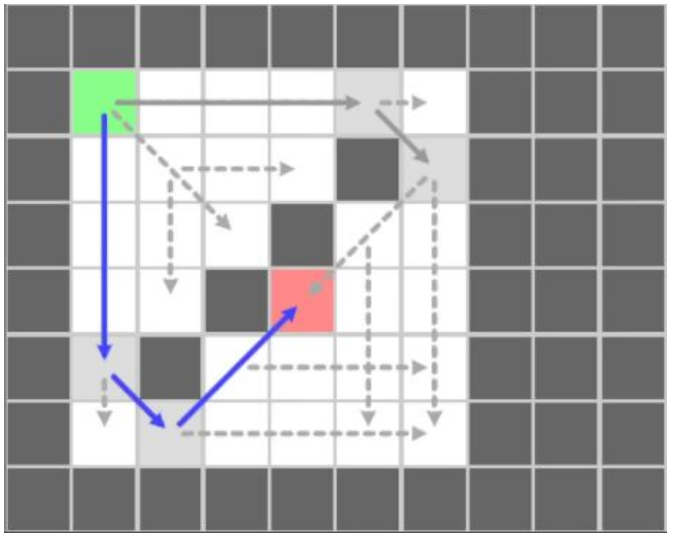


Jump Points: Example

A* has to explore many neighbors, often exploring redundant paths

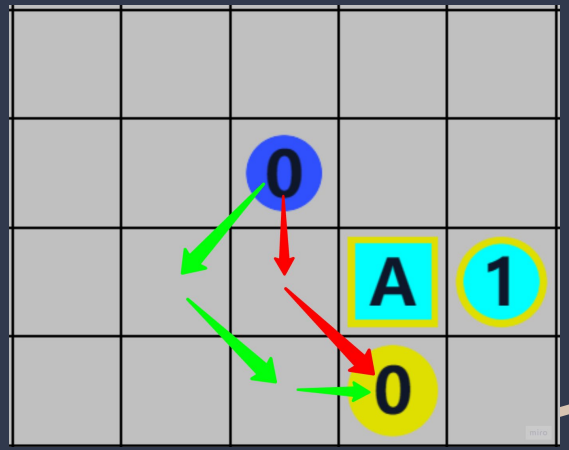


JPS only needs to explore four jump points to find two optimal paths

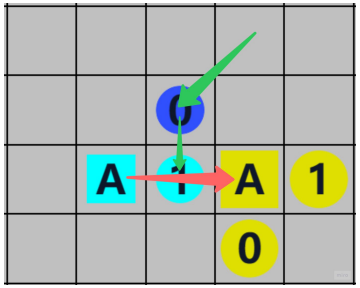


Demo source: https://zerowidth.com/2013/a-visual-explanation-of-jump-point-search/

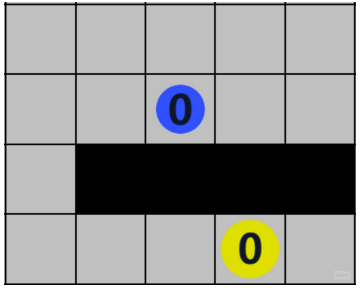
Conflict Resolution: JPCBS



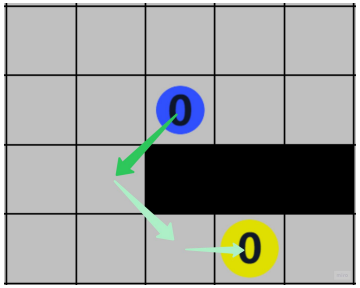
Time cost of green path = 5
Time cost of red path = 6



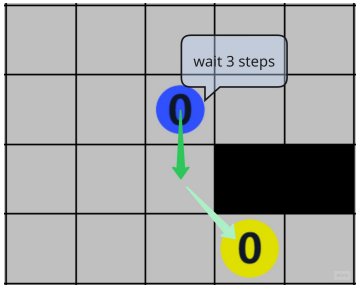
step n



step n+1

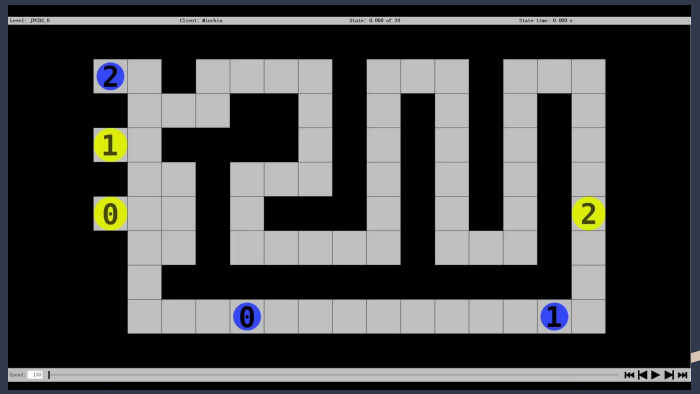


step n+2

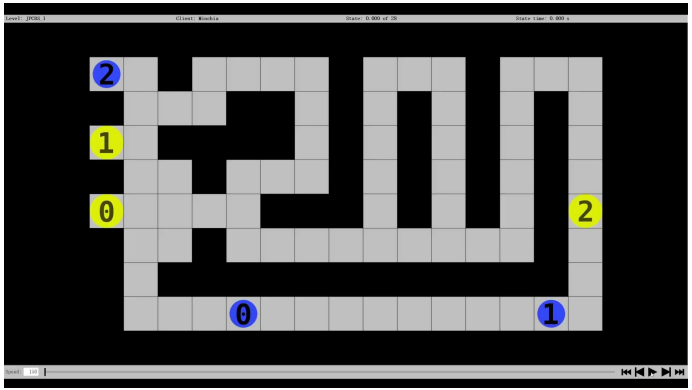


step n+3

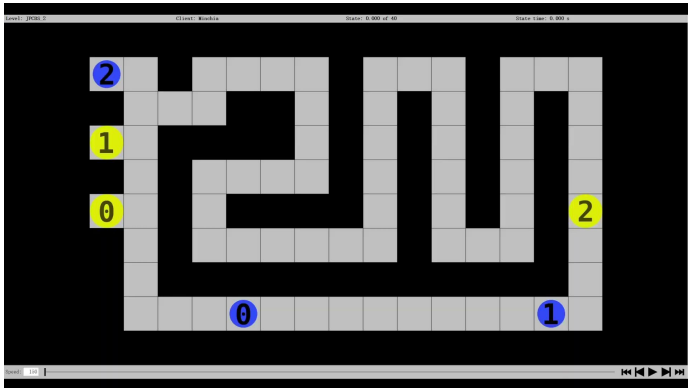
Conflict Resolution: JPCBS



avoid first conflict and wait second conflict disappear

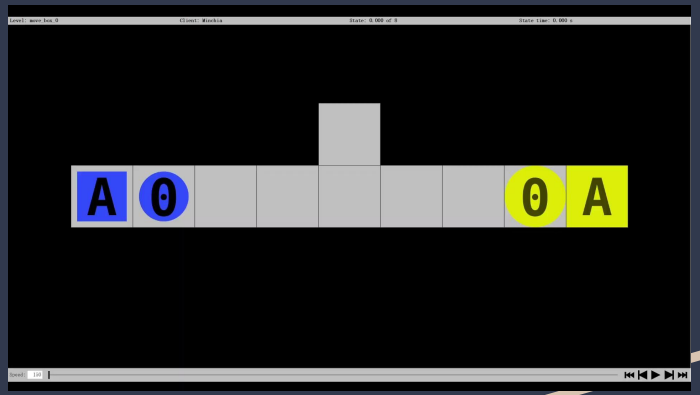


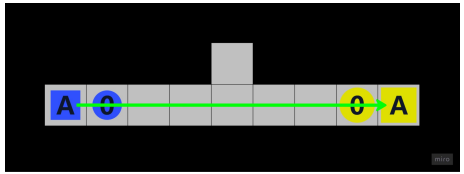
avoid two conflicts



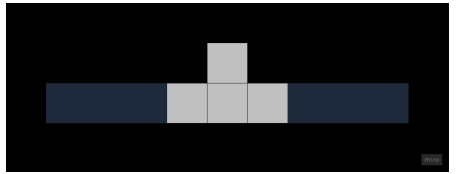
wait until two conflicts disappear

Conflict Resolution: JPCBS

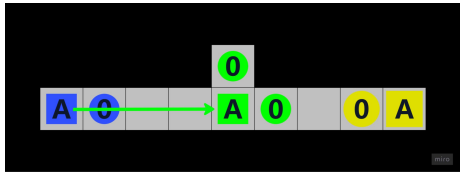




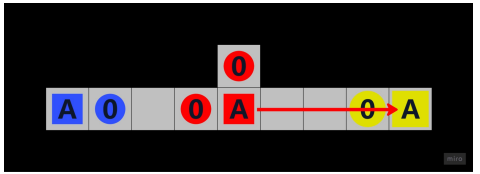
get box path by JPCBS



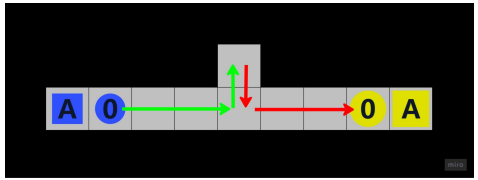
find a spatial structure like this



get first part of path by JPCBS

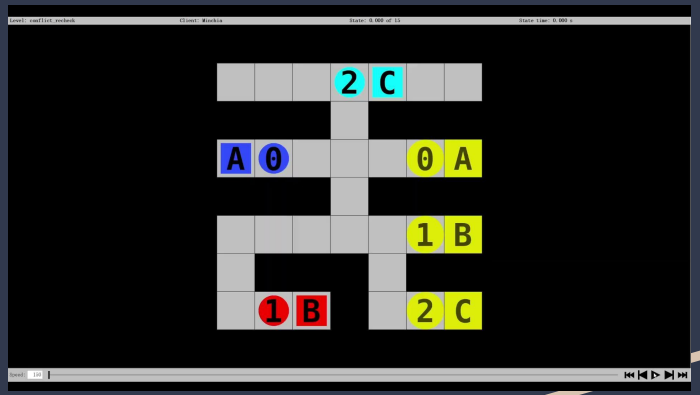


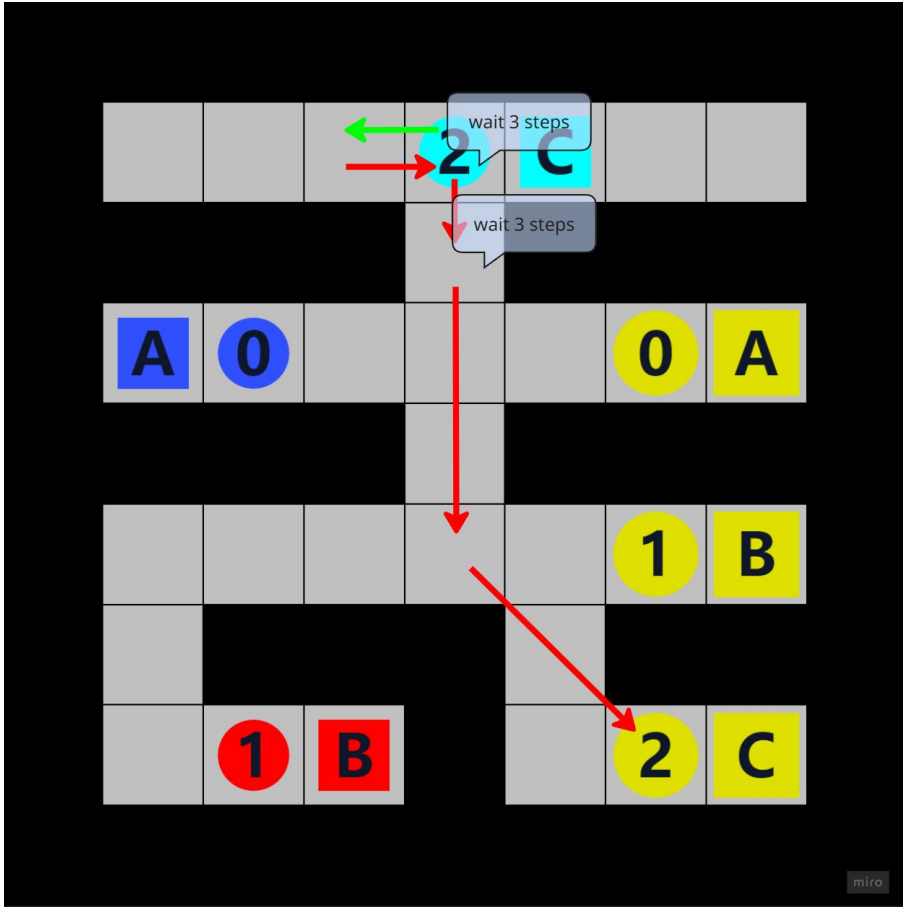
get second part of path by JPCBS



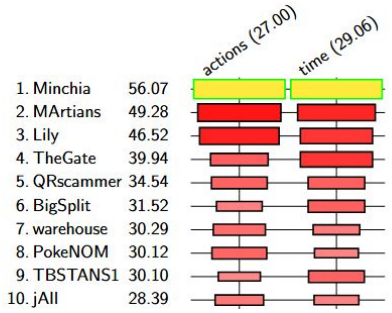
connect two parts together

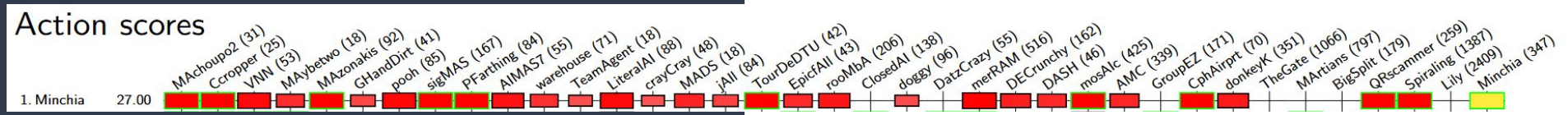
Conflict Resolution: conflict recheck

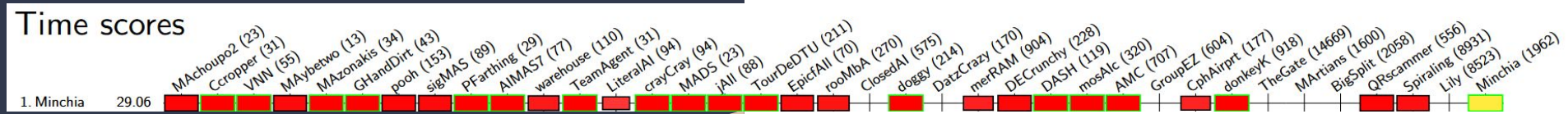




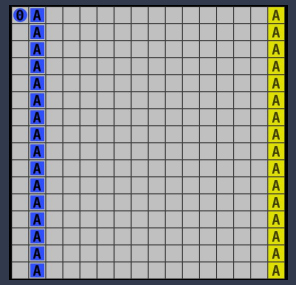
Benchmarks


actions (27.00)
time (29.06)
1. Minchia 56.07
2. MArtians 49.28
3. Lily 46.52
4. TheGate 39.94
5. QRscammer 34.54
6. BigSplit 31.52
7. warehouse 30.29
8. PokeNOM 30.12
9. TBSTANS1 30.10
10. jAll 28.39

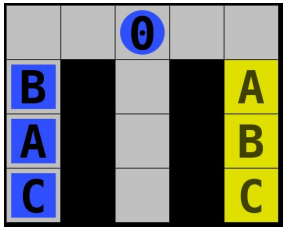

Action scores
1. Minchia 27.00
MAchoupo2 (31)
Ccropper (25)
VNN (53)
MAybetwo (18)
MAzonakis (92)
GHandDirt (41)
pooh (85)
sigMAS (167)
PFarthing (84)
AIMAS7 (55)
warehouse (71)
TeamAgent (18)
LiteralAI (88)
crayCray (48)
MADS (18)
jAll (84)
TourDeDTU (42)
EpicfAll (43)
rooMbA (206)
ClosedAI (138)
doggy (96)
DatzCrazy (55)
merRAM (516)
DECrunchy (162)
DASH (46)
mosAlc (425)
AMC (339)
GroupEZ (171)
CphAirprt (70)
donkeyK (351)
TheGate (1066)
MArtians (797)
BigSplit (179)
QRscammer (259)
Spiraling (1387)
Lily (2409)
Minchia (347)


Time scores
1. Minchia 29.06
MAchoupo2 (23)
Ccropper (31)
VNN (55)
MAybetwo (13)
MAzonakis (34)
GHandDirt (43)
pooh (153)
sigMAS (89)
PFarthing (29)
AIMAS7 (77)
warehouse (110)
TeamAgent (31)
LiteralAI (94)
crayCray (94)
MADS (23)
jAll (88)
TourDeDTU (211)
EpicfAll (70)
rooMbA (270)
ClosedAI (575)
doggy (214)
DatzCrazy (170)
merRAM (904)
DECrunchy (228)
DASH (119)
mosAlc (320)
AMC (707)
GroupEZ (604)
CphAirprt (177)
donkeyK (918)
TheGate (14669)
MArtians (1600)
BigSplit (2058)
QRscammer (556)
Spiraling (8931)
Lily (8523)
Minchia (1962)

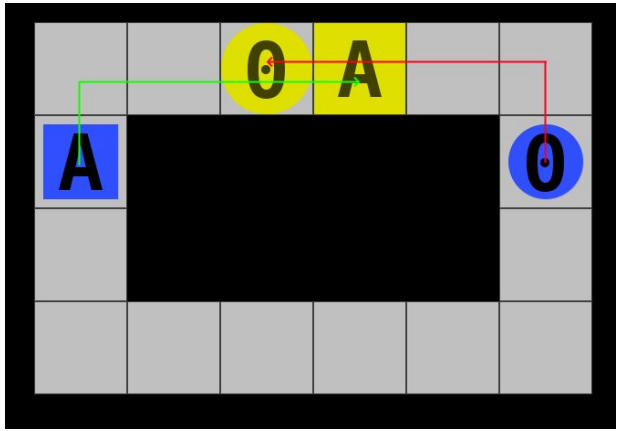
Benchmarks

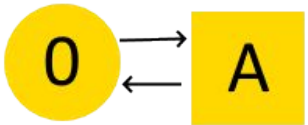


Level	Eval	Points/States Generated	Time/s	Solution length
SAsoko3_16	BFS	-	-	-
	A*	10,167	3,530	449
	Minchia	5,873	1,116	421
SAsoko3_32	BFS	-	-	-
	A*	-	-	-
	Minchia	39,393	10,282	1861
SAsoko3_64	BFS	-	-	-
	A*	-	-	-
	Minchia	287,681	137,894	7813
SAsoko3_128	BFS	-	-	-
	A*	-	-	-
	Minchia	-	-	-
SAtowersOfSaigon03	BFS	20,818	1,482	62
	A*	13,199	1,155	132
	Minchia	510	0,130	62
SAtowersOfSaigon05	BFS	-	-	-
	A*	-	-	-
	Minchia	1,564	0,206	188
SAtowersOfSaigon10	BFS	-	-	-
	A*	-	-	-
	Minchia	7,408	1,023	1028
SAtowersOfSaigon26	BFS	-	-	-
	A*	-	-	-
	Minchia	72,896	22,473	13.908
SAbispebjergHospital	BFS	-	-	-
	A*	11,746	33,143	1526
	Minchia	24,633	2,360	1407

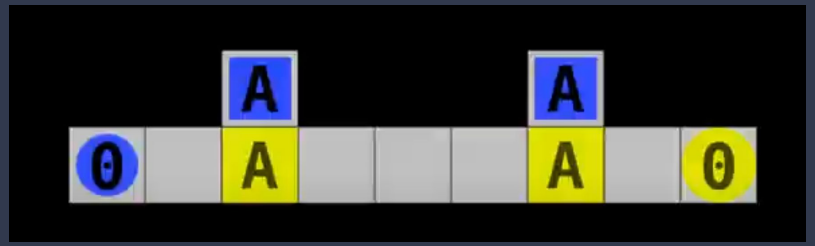


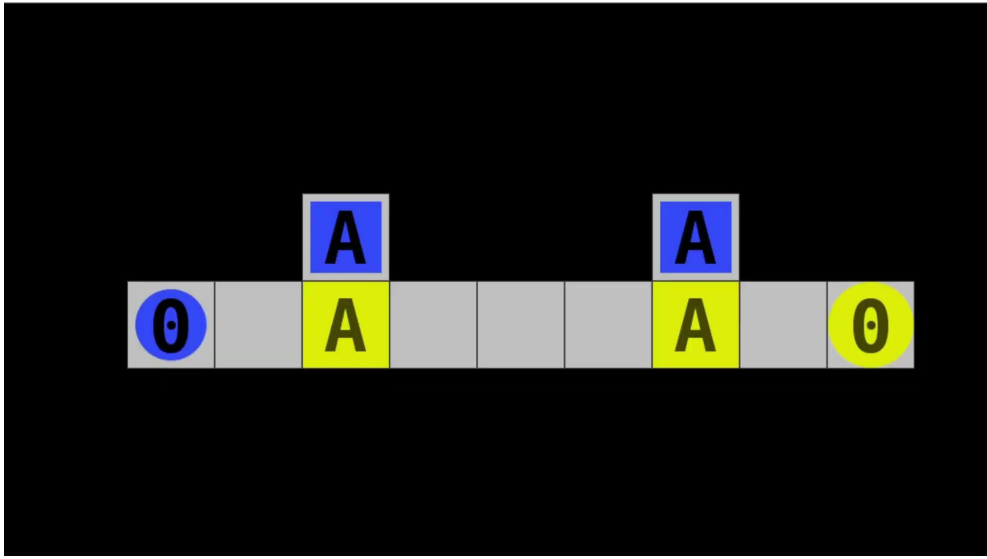
Flaws with the algorithm

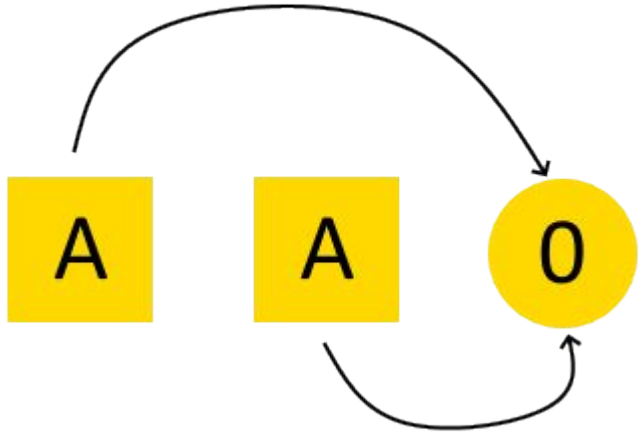




Flaws with the algorithm



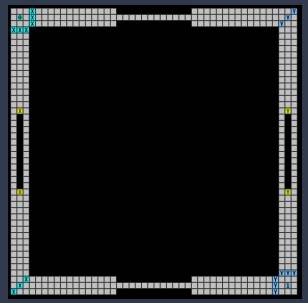




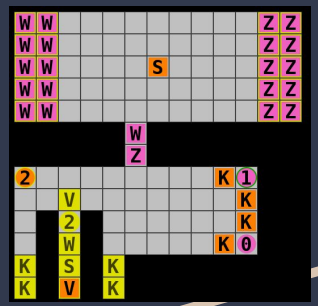
Demo

Level	Points Generated	Time/s	Solution length
MinchiaSubmap_0	2,363	0,464	182
MinchiaSubmap_1	45,206	3,571	324
MinchiaSubmap_2	27,849	2,190	347
MinchiaSubmap_3	6,785	0,544	113
Minchia	83,100	7,106	347

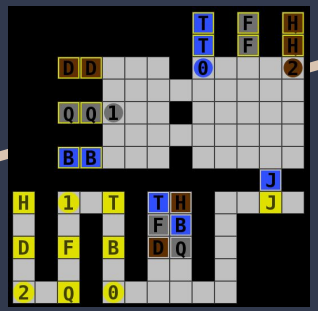
0:



1:



2:



3: