Priority Inheritance with Backtracking for Iterative Multi-agent Path Finding

Keisuke Okumura[1], Manao Machida[2], Xavier Défago[1], Yasumasa Tamura[1]

1. Tokyo Institute of Technology
2. NEC Corporation

IJCAI-19
Aug. 13th, 2019

Multi-agent Path Finding (MAPF)

given

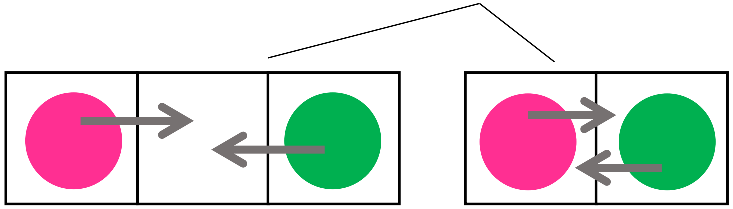
□ graph

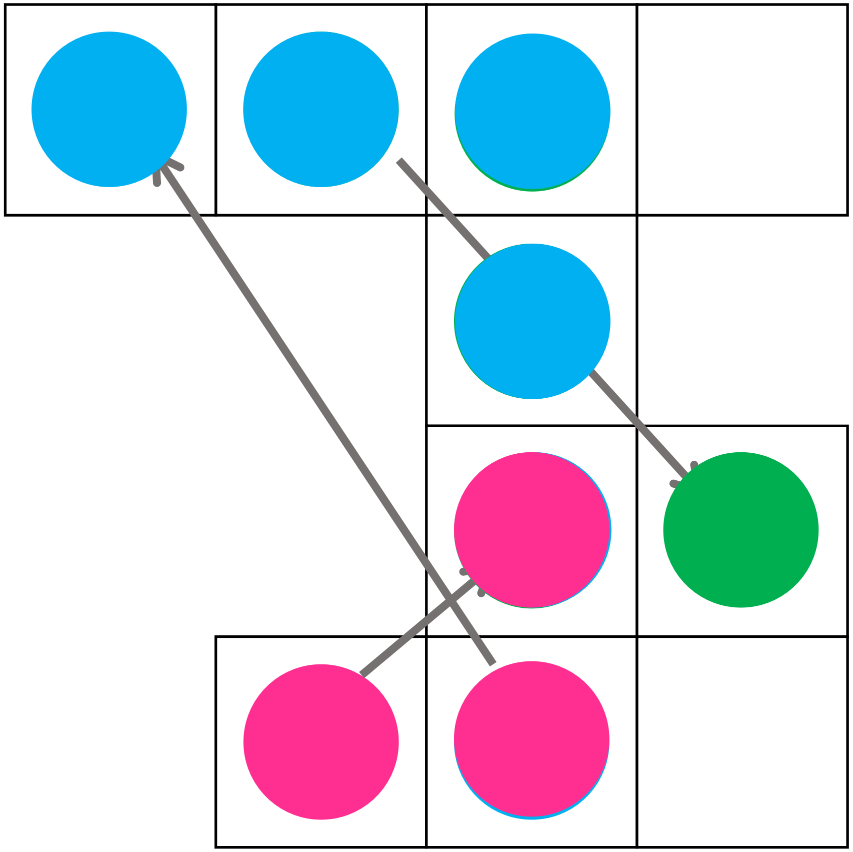
● agents (starts)

■ goals

obtain

1. paths without collisions





s.t.

all agents are on their goals **simultaneously**

computationally **DIFFICULT** to obtain optimal solutions

Multi-agent Path Finding (MAPF)

Systems with moving agents will be more and more common.

YouTube/Mind Blowing Videos

Twitter/People's Daily, China

In practical scenarios, MAPF must be solved

iteratively

OUR FOCUS

one-shot MAPF

iterative MAPF

Iterative Multi-agent Path Finding

given

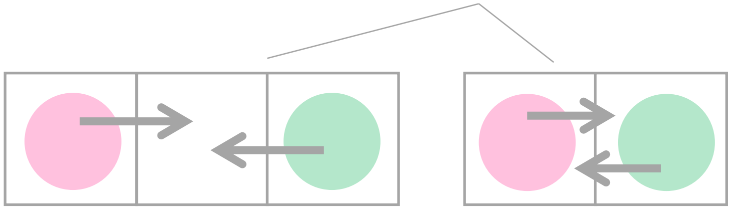
□ graph

● agents (starts)

☒ **tasks (set of goals)**

obtain

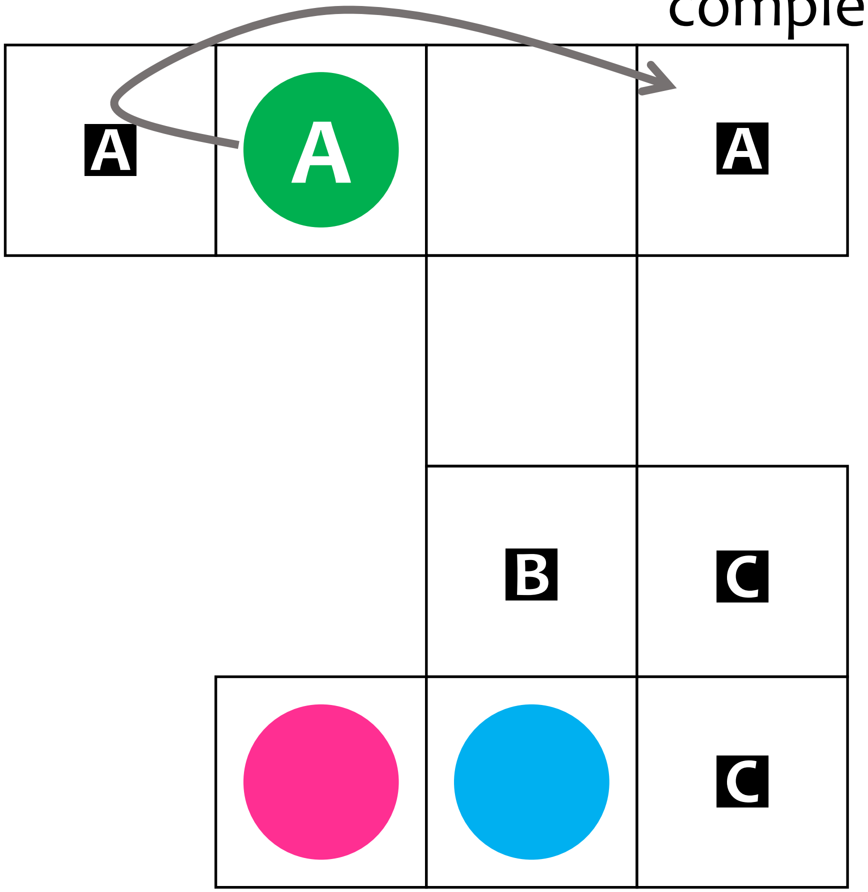
1. paths without collisions



2. task allocation

s.t.

all tasks are completed in finite time



Overview

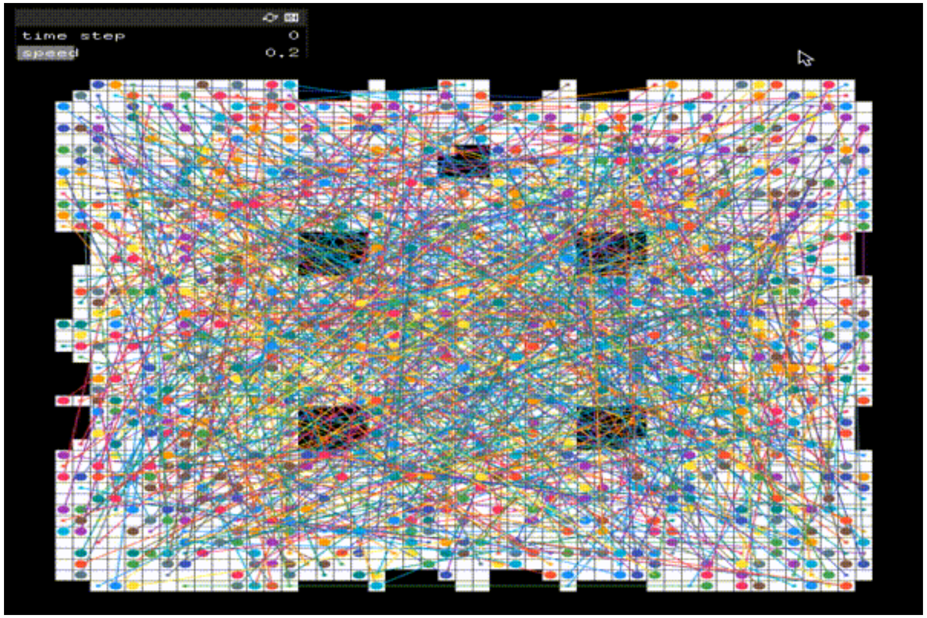
Priority **I**nheritance with **B**ack**t**racking

✓ for solving iterative MAPF, propose an algorithm **PIBT** that ensures **reachability**

(in biconnected-like graph)

all agents reach their goals in finite time after being given

fast, scalable

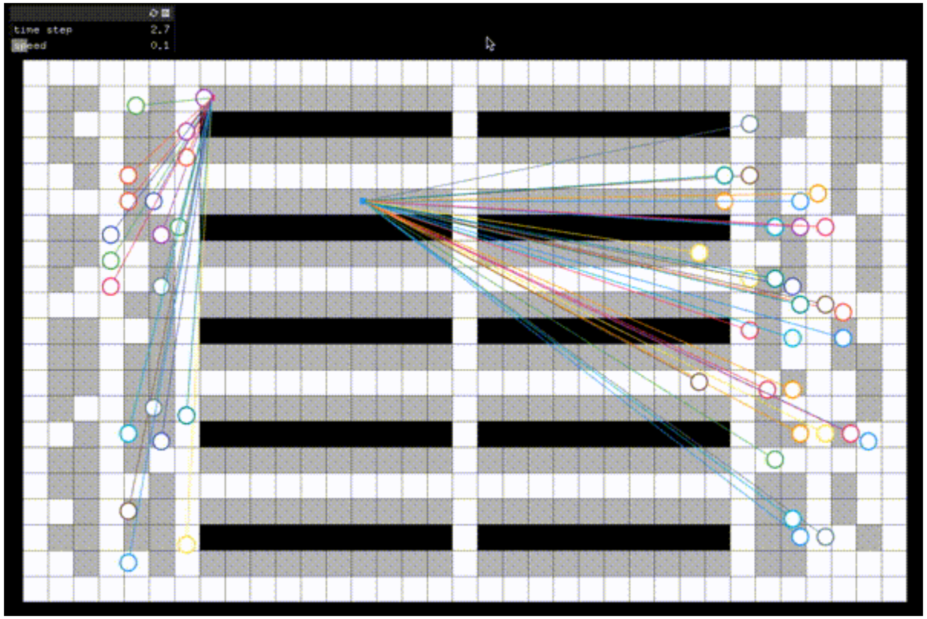


time step 0

speed 0.2

500 agents, within 0.5 sec

adaptive for iterative use



time step 2.7

speed 0.1

Multi-agent Pickup & Delivery

[Ma et al., AAMAS17]

Design Choice of Algorithm

tradeoff

optimality vs [✓ speed]

style

centralized vs [✓ decentralized]

communication

globally vs [✓ locally]

+

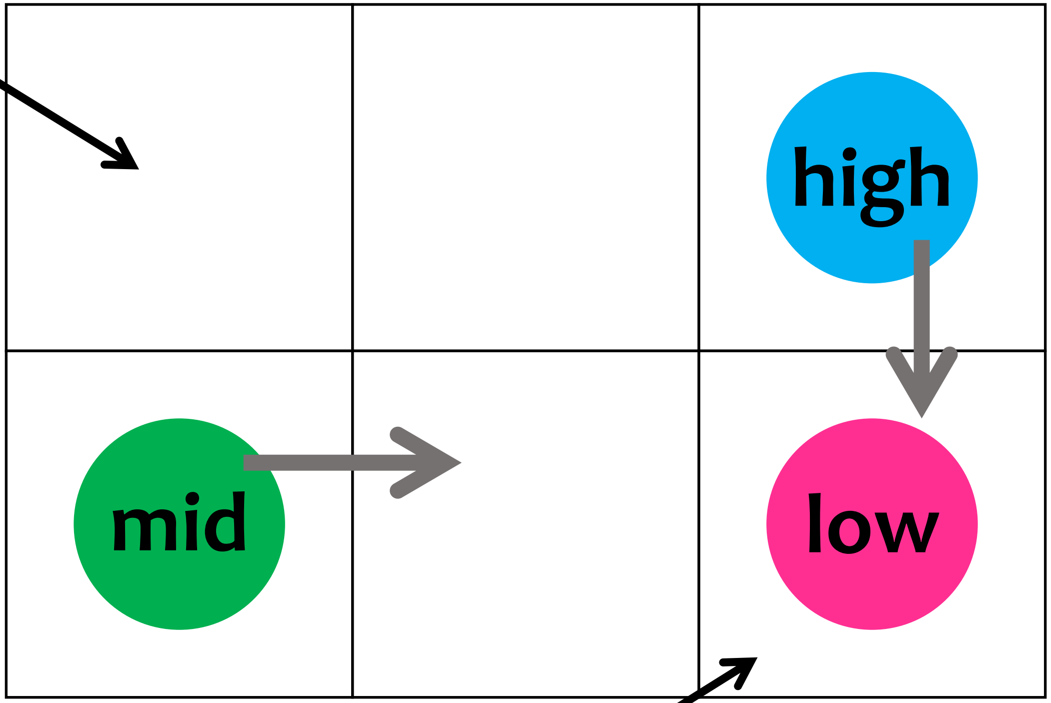
Adaptivity for Iterative Use

Prioritized Planning

incomplete

single time-window
(PIBT relies on)

node = resource



STUCK

priority inversion in resource scheduling

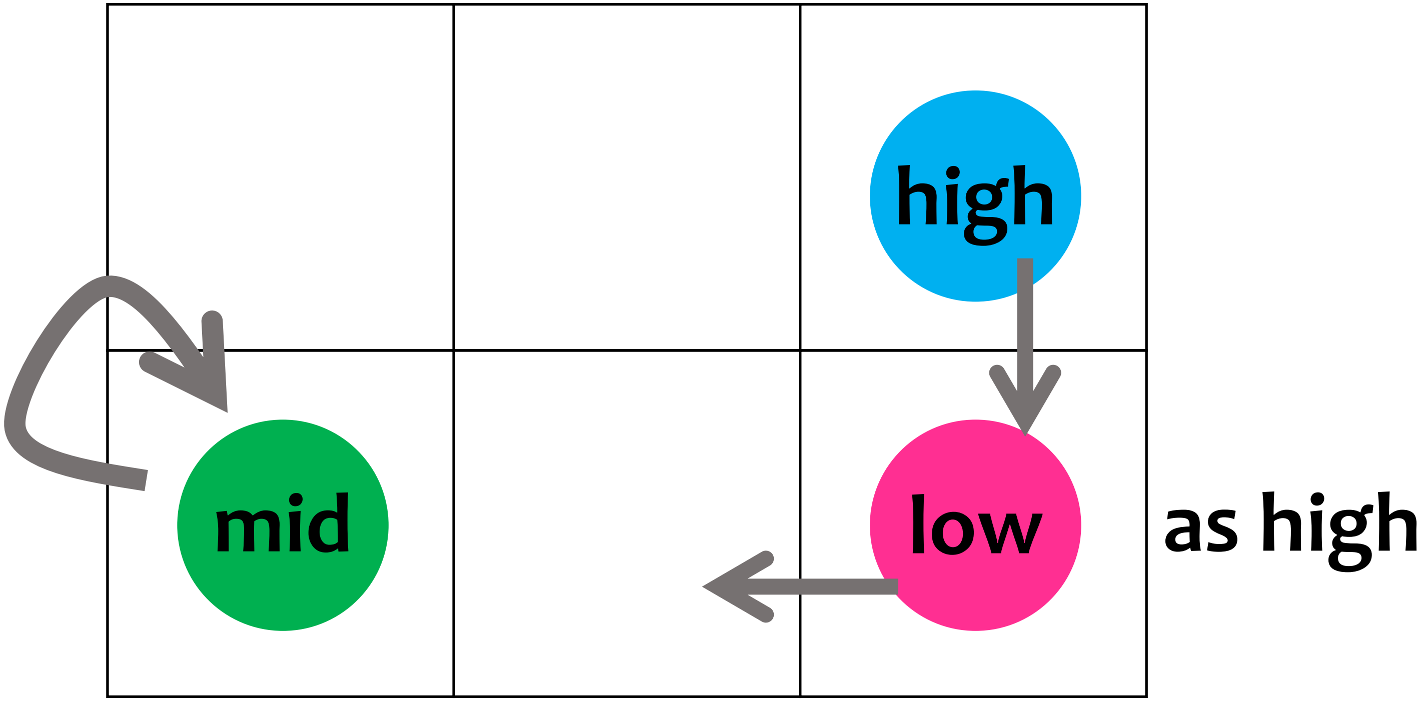
Priority Inheritance

countermeasure to priority inversion

⬇

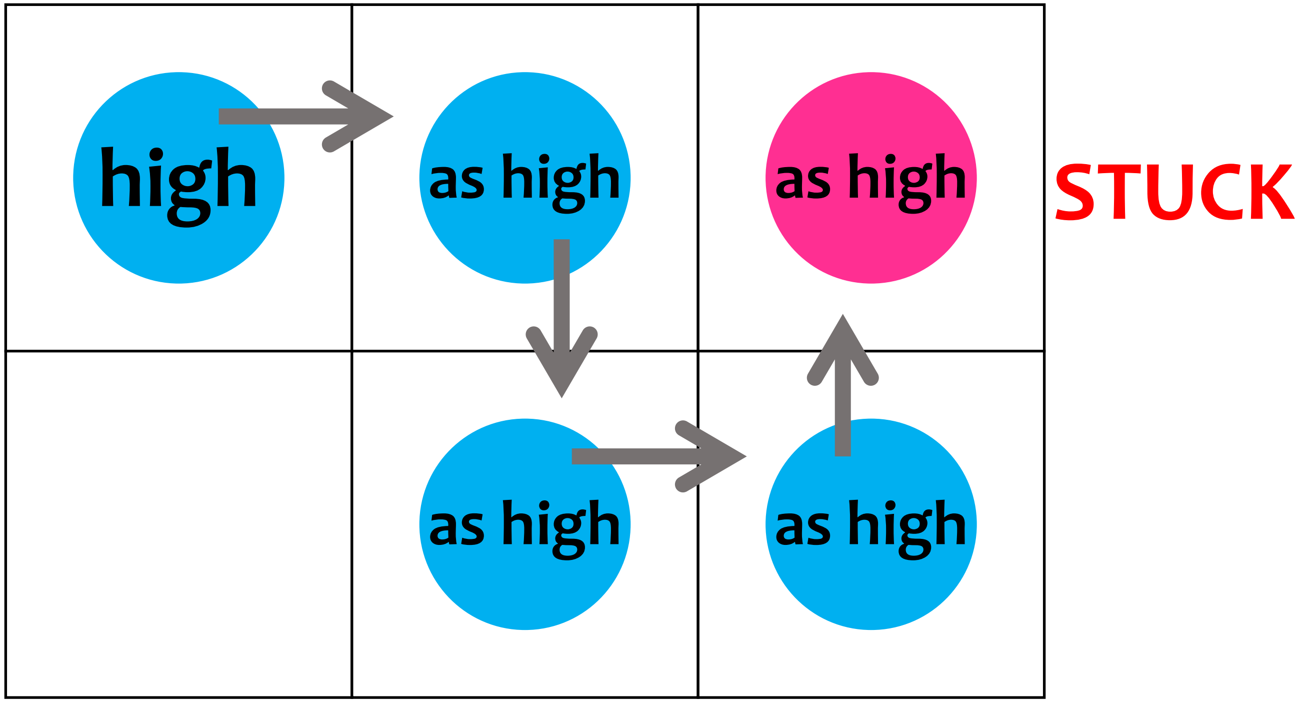
priority inheritance

[Sha et al., 1990]



Stuck Again

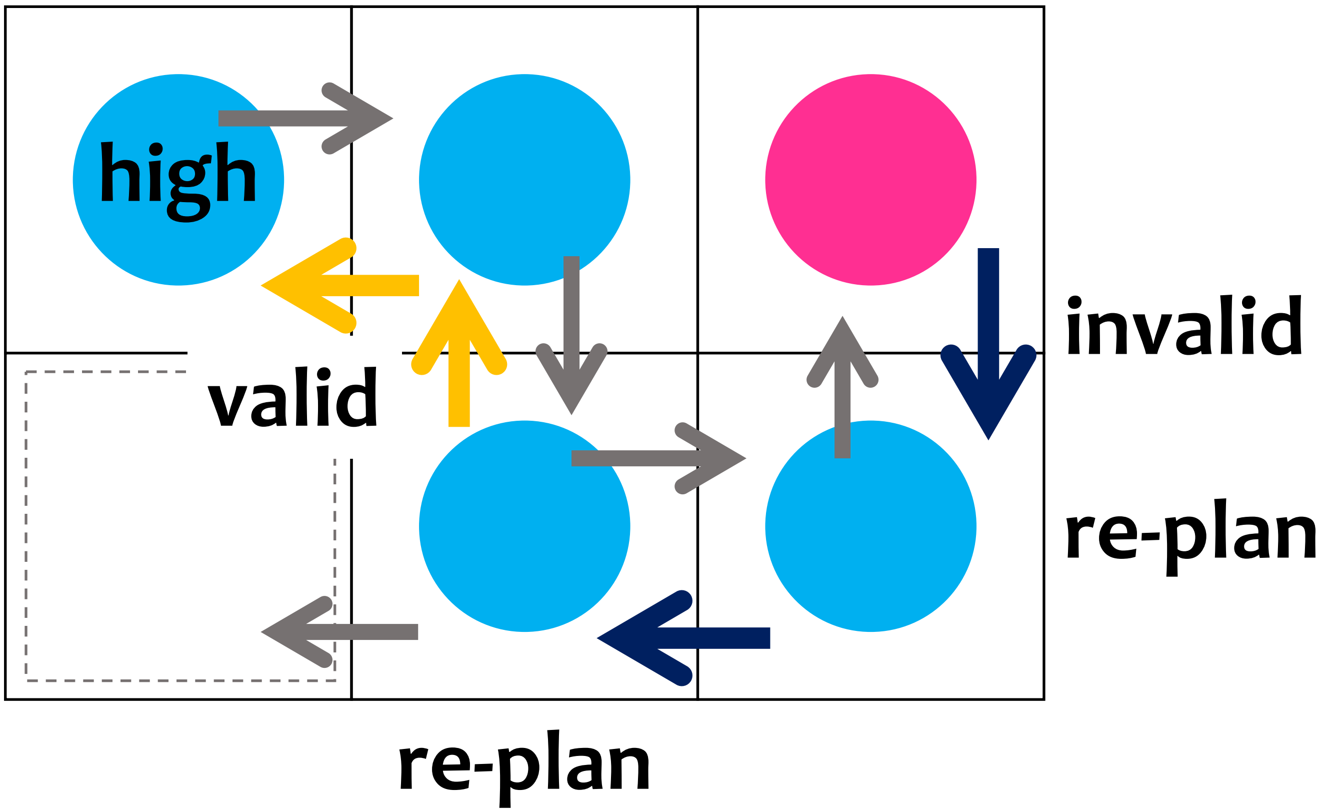
Priority inheritance is **insufficient** to prevent all stuck situations.



with Backtracking

Agents with priority inheritance have to wait for **backtracking**.

→	valid	You can move
→	invalid	You must re-plan, I will stay

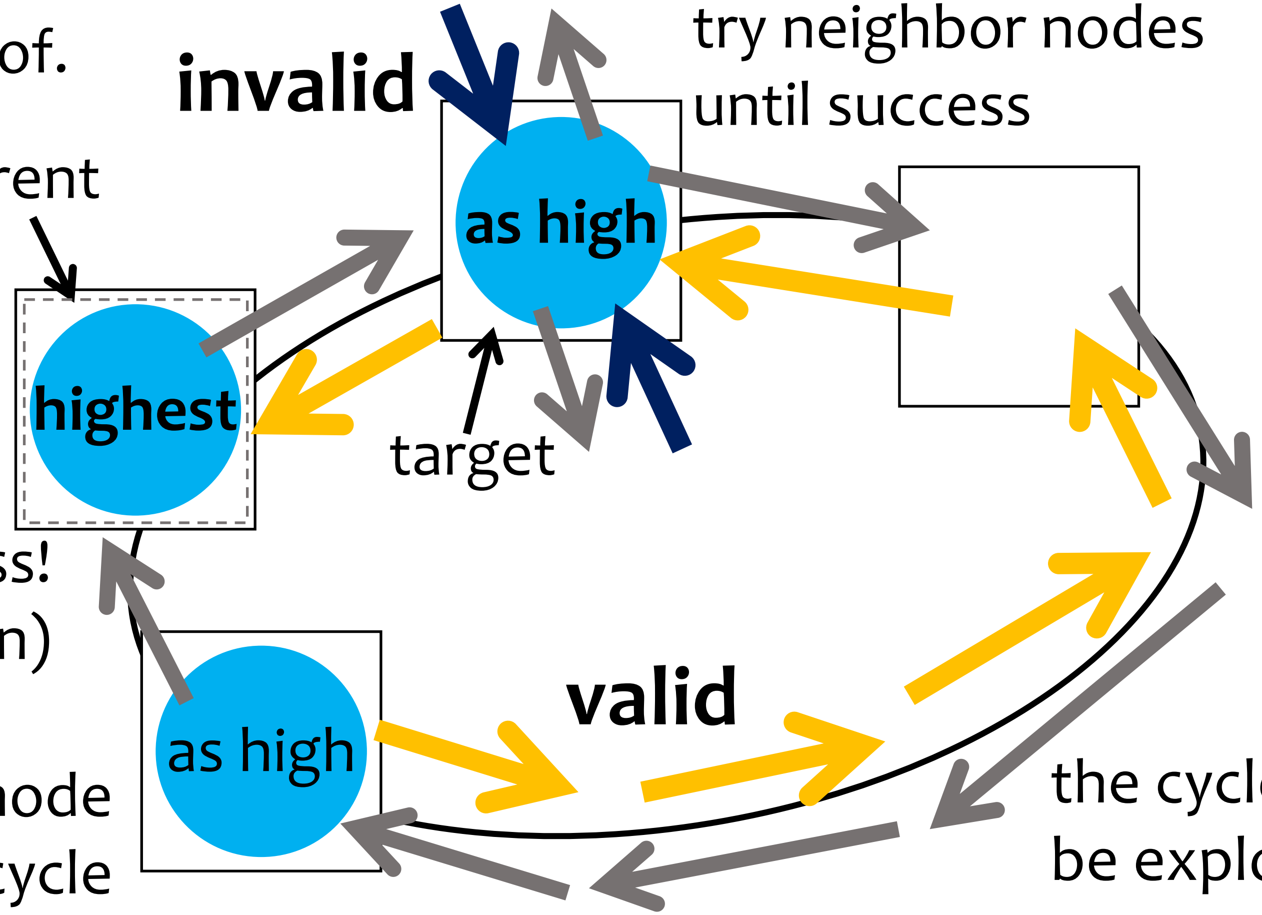


Guarantee of Local Movement

Lemma 1

The agent with **highest priority** successfully moves to a target node if the two nodes (current, target) belong to a simple cycle.

sketch of proof.



try neighbor nodes until success

always success! (rotation)

last node of the cycle

the cycle must be explored

Local to Global Movement

Lemma 1

⬇

(depends on graph)

the agent with highest priority can move to an **arbitrary neighbor** node in next timestep.

⬇

such an agent always reach its **goal** with the **shortest path**

↑ how to select?

⬇

by **dynamic priorities,**

(it is enabled to ensure that)

all agents reach their own goals in finite time

Dynamic Priorities

Once an agent reaches its goal, it **drops** priority.

⬇

Agents that have not reached goal have higher priority.

unique between agents
for every timestep

$$p_i(t) \leftarrow \eta_i(t) + \epsilon_i$$

for an agent a_i at timestep t

elapsed timestep
since a_i last updated its goal,
or, be zero if no goal

tie-breaker, in [0, 1)
unique between agents

Reachability

Theorem 2

If a graph G has a **simple cycle for all pairs of adjacent nodes**, then, with PIBT, **all agents** reach their own goals within $\mathrm{diam}(G) \cdot |A|$ timesteps **after being given**.

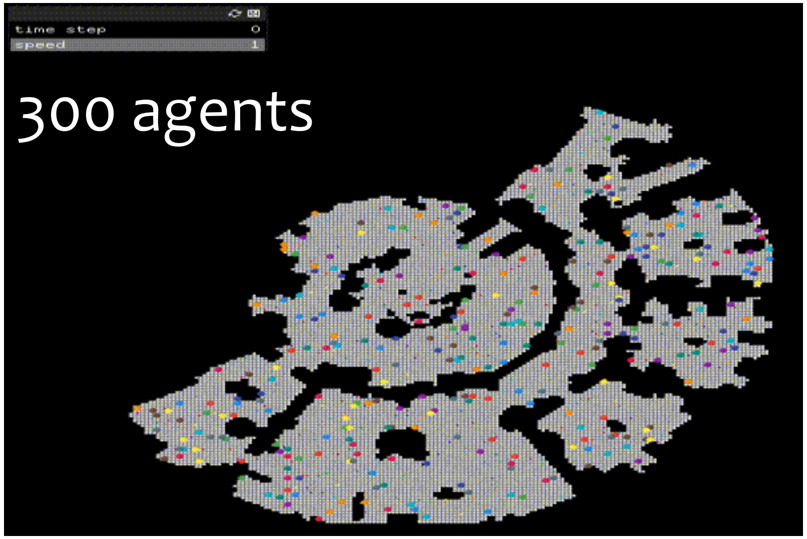
#agents

example

- undirected: biconnected
- directed: ring

NOT ensure that all agents be on their goals **simultaneously**!

Exp.1 one-shot MAPF



ost003d, 194x194 [Sturtevant, 2012]

● PIBT

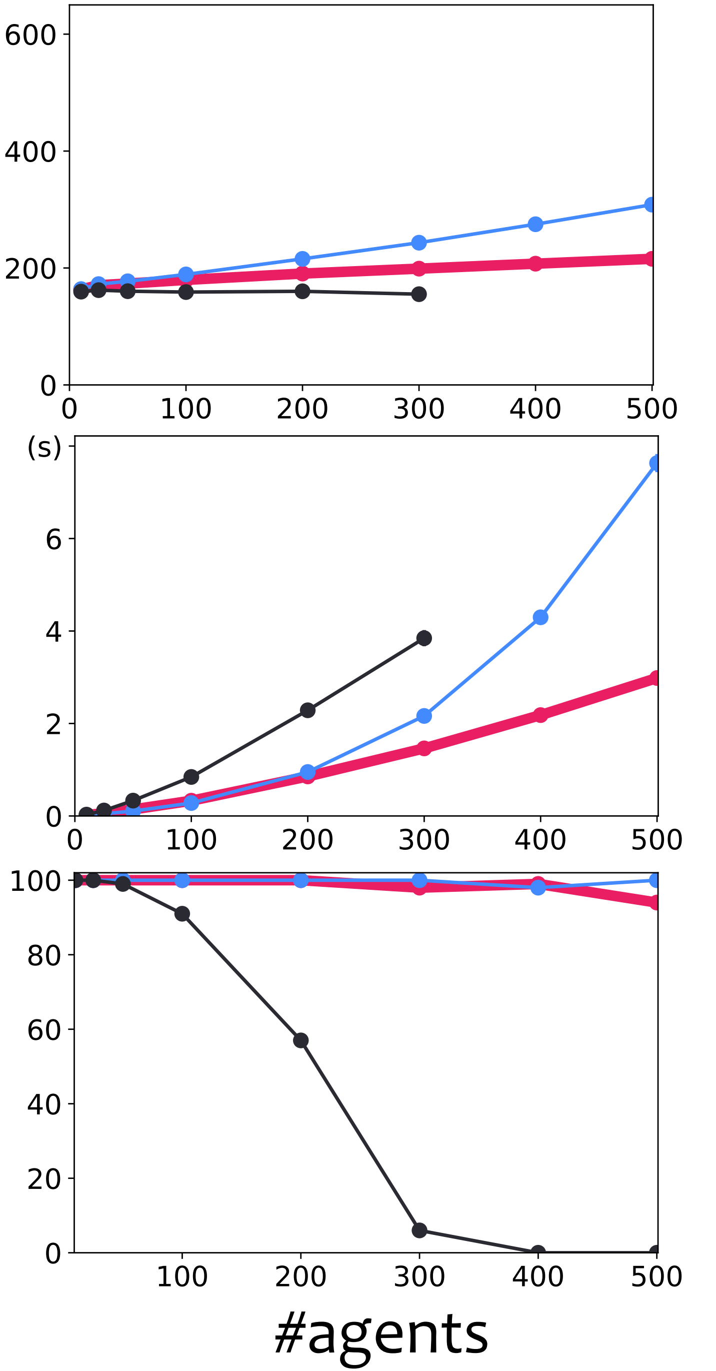
● Parallel Push & Swap
[Sajid et al., SoCS12]

● WHCA*-10
[Silver, AIIDE05]

path cost
(steps)

computation time
(sec.)

success
(%)

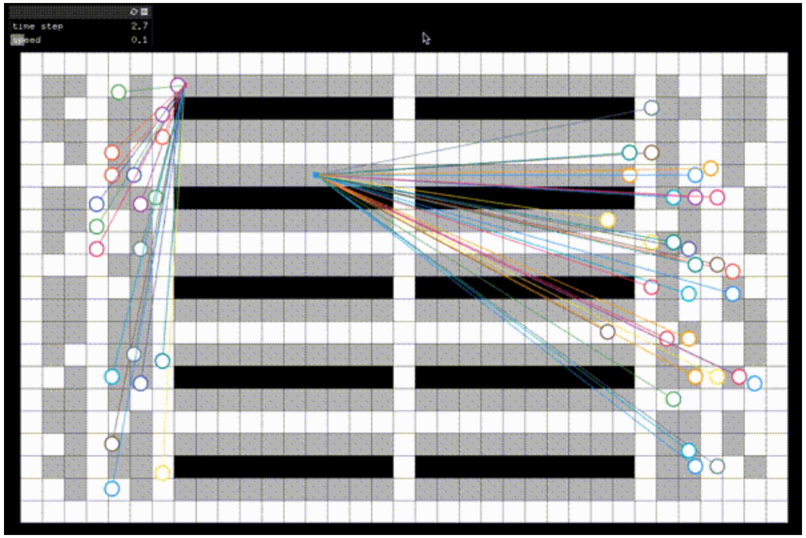


*average over only success cases within the solver (in 100 times)

Exp.2 Multi-agent Pickup & Delivery

[Ma et al., AAMAS17]

task = (pickup loc., delivery loc.)



PIBT with task allocation

let free agents move to the **nearest pickup** loc.

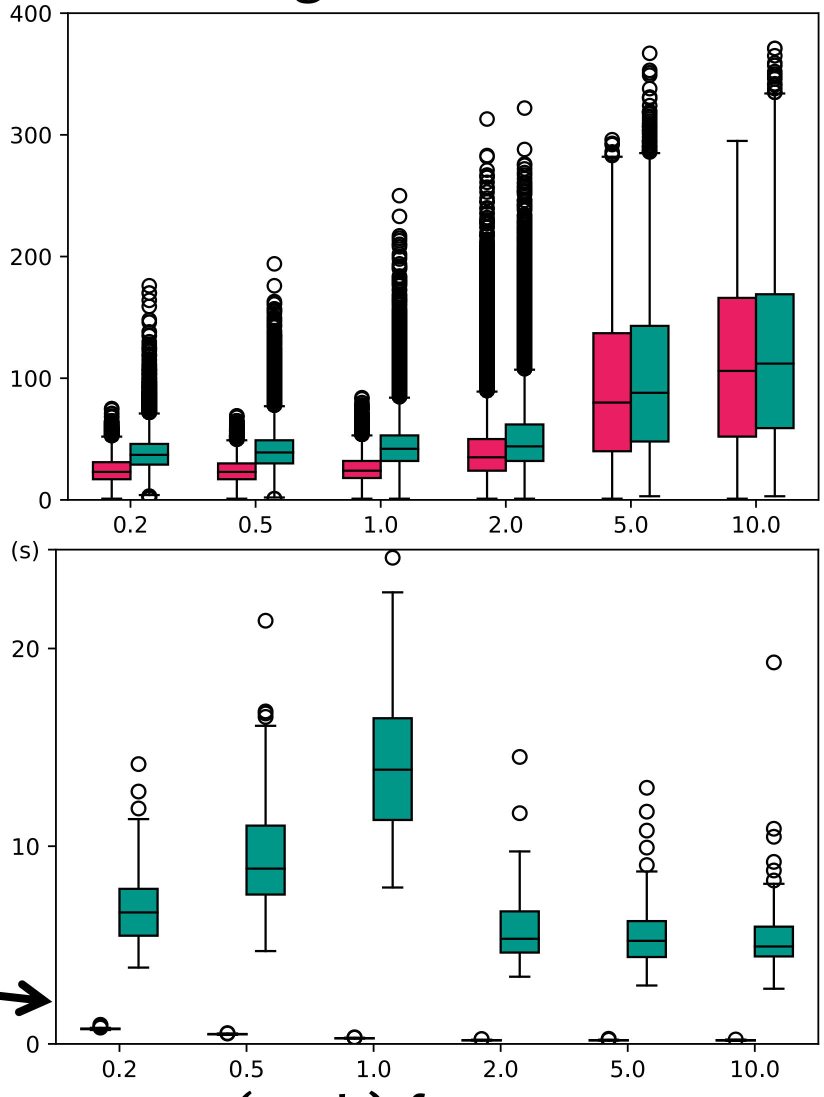
🟥 PIBT

🟩 TP
[Ma et al., AAMAS17]

50 agents, 500 tasks

service time (steps)

comp. time (sec)

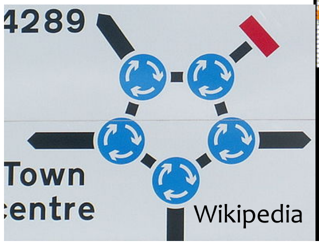


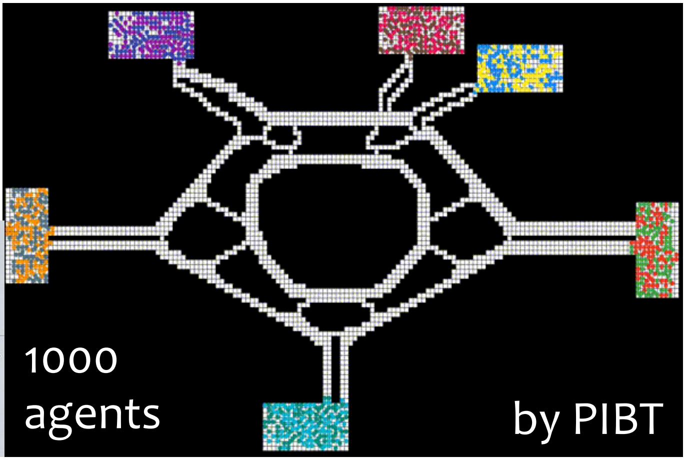
(task) frequency

Conclusion

- ✓ **PIBT** is an algorithm for solving **iterative MAPF**
- ✓ PIBT ensures **reachability** in **biconnected**-like graph

Magic Roundabout



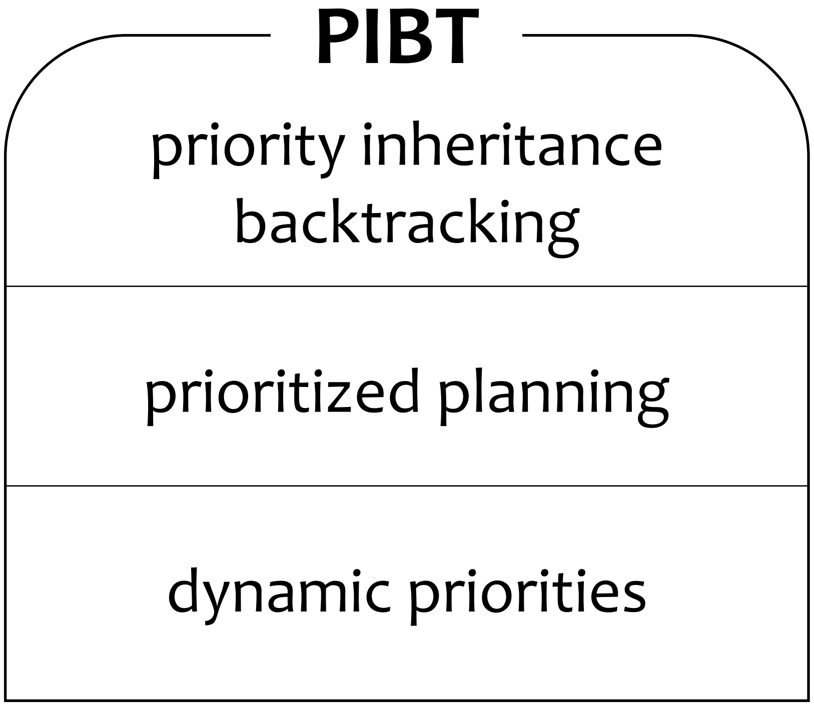


future work

- expand time window

  technical paper is available, called **winPIBT** [arXiv:1905.10149]

- relax graph conditions
- adapt to full asynchrony



PIBT

priority inheritance backtracking

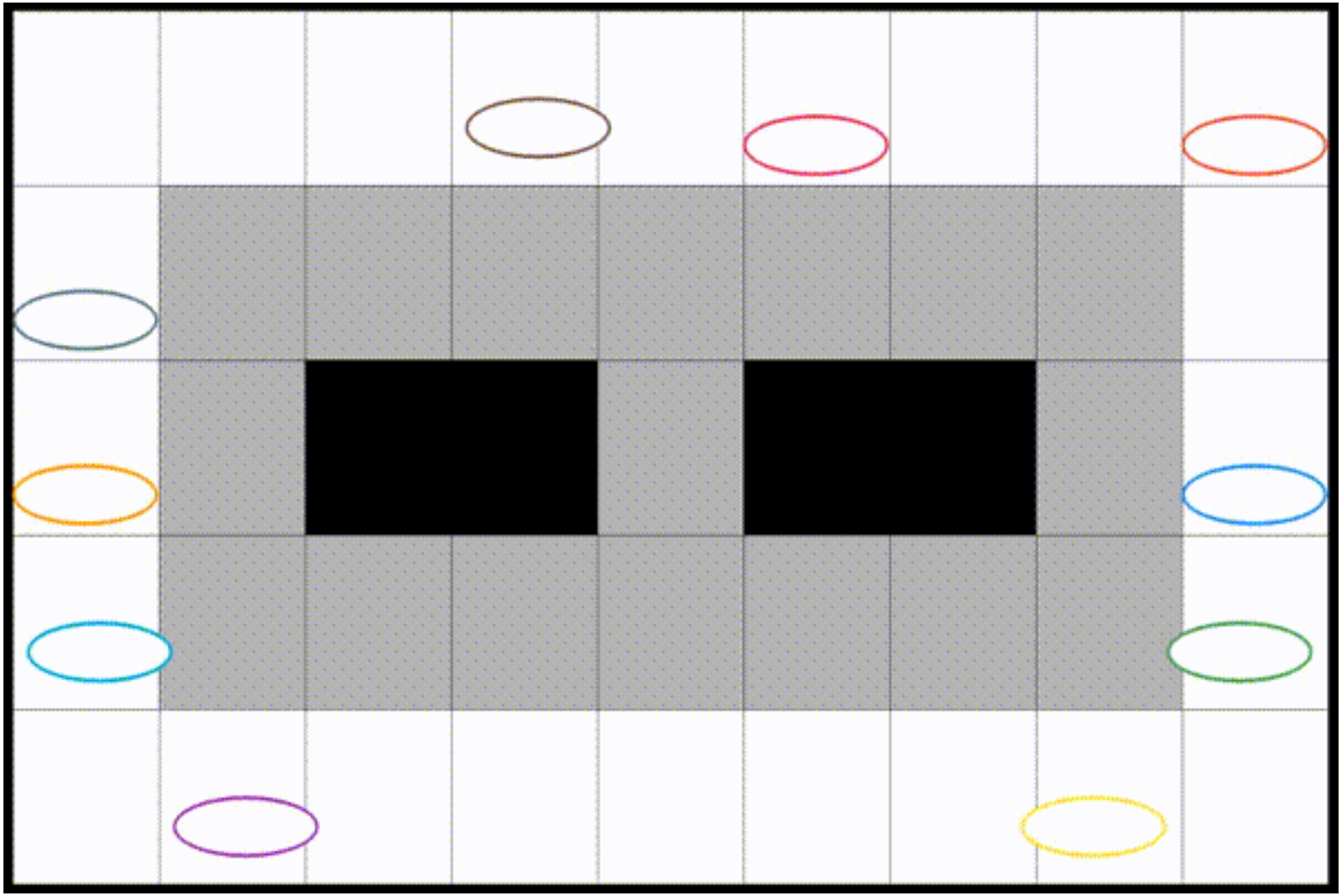
prioritized planning

dynamic priorities

Appendix

Multi-agent Sushi Pickup & Delivery

inspired by conveyor belt sushi



Simple Sketch of PIBT

An agent starts its planning with

- receiving priority inheritance
- becoming the highest in unplanned agents

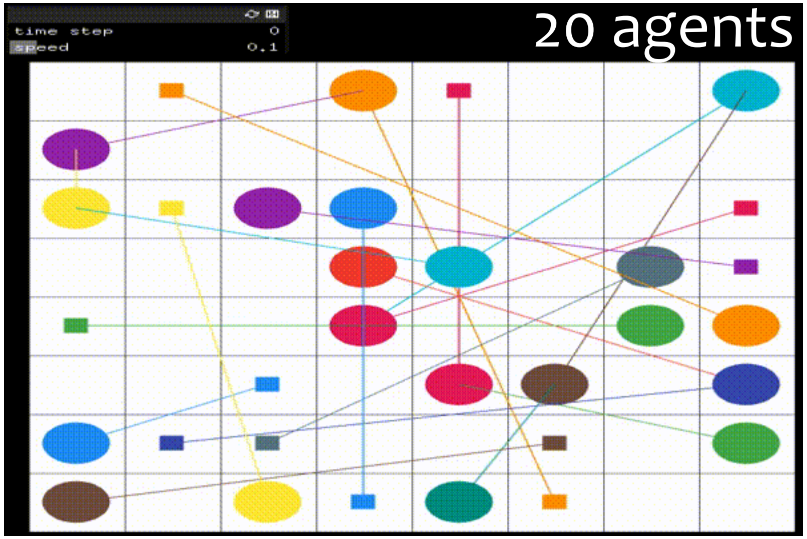
1. Initialize/update candidate nodes for next timestep
   Sends invalid as backtracking if no candidates

2. Pickup one node from candidates

3. If no agent there, move there (fin.)
   Otherwise, do priority inheritance and wait for backtracking

4. Receive backtracking
   valid: move there (fin.)
   invalid: back to 1

Exp.1 one-shot MAPF

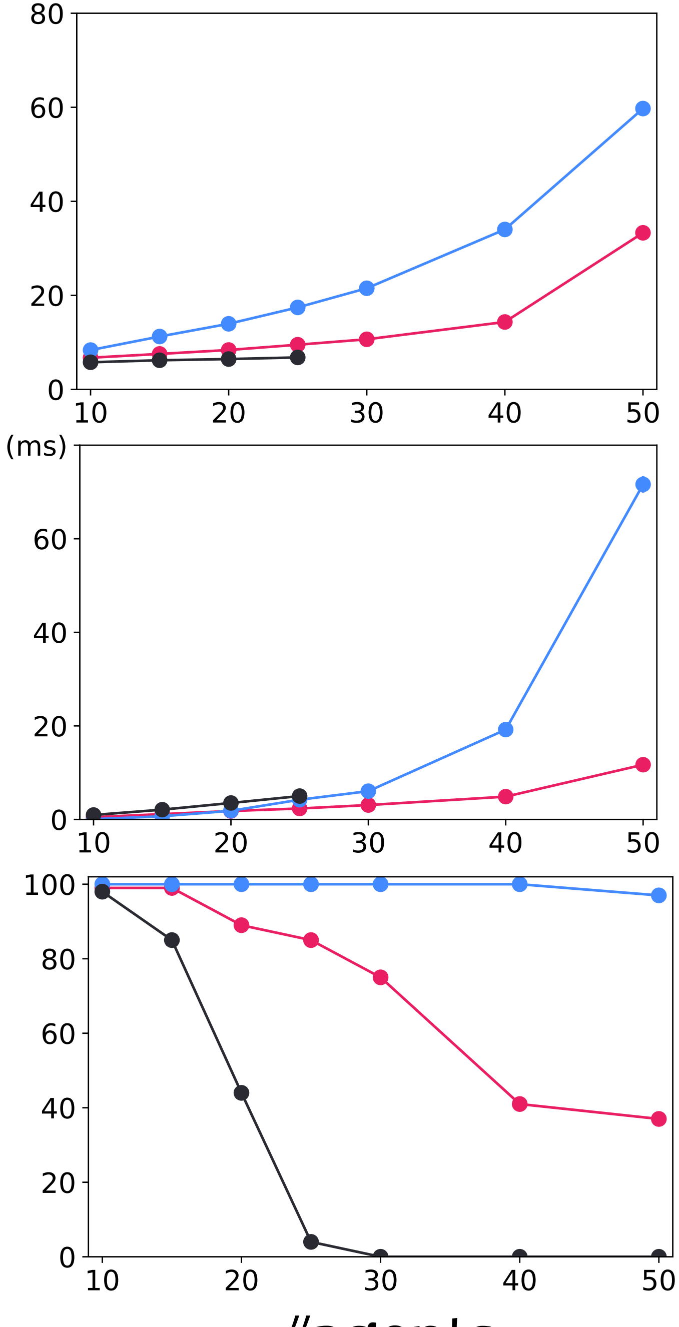


simple grid, 8x8

path cost
(steps)

runtime
(ms)

success
(%)



#agents

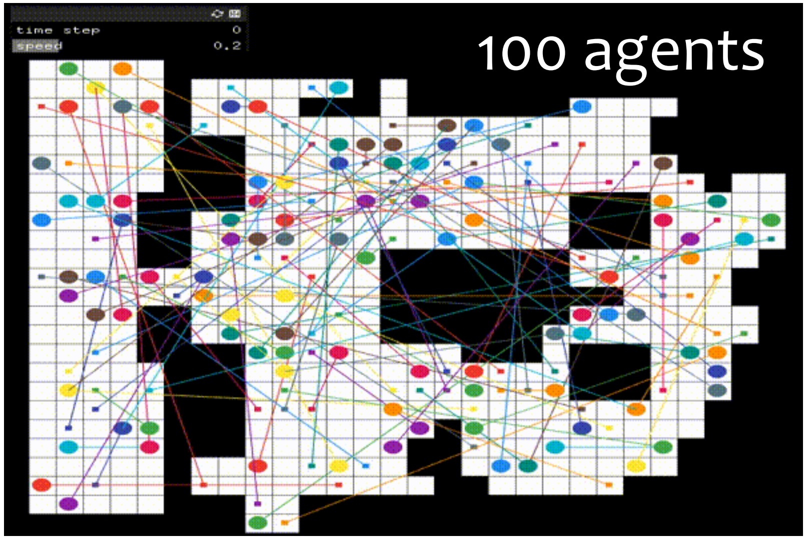
● PIBT
[Okumura et al., IJCAI19]

● Parallel Push & Swap
[Sajid et al., SoCS12]

● WHCA-10
[Silver, AIIDE05]

*average over only success cases within the solver (in 100 times)

Exp.1 one-shot MAPF



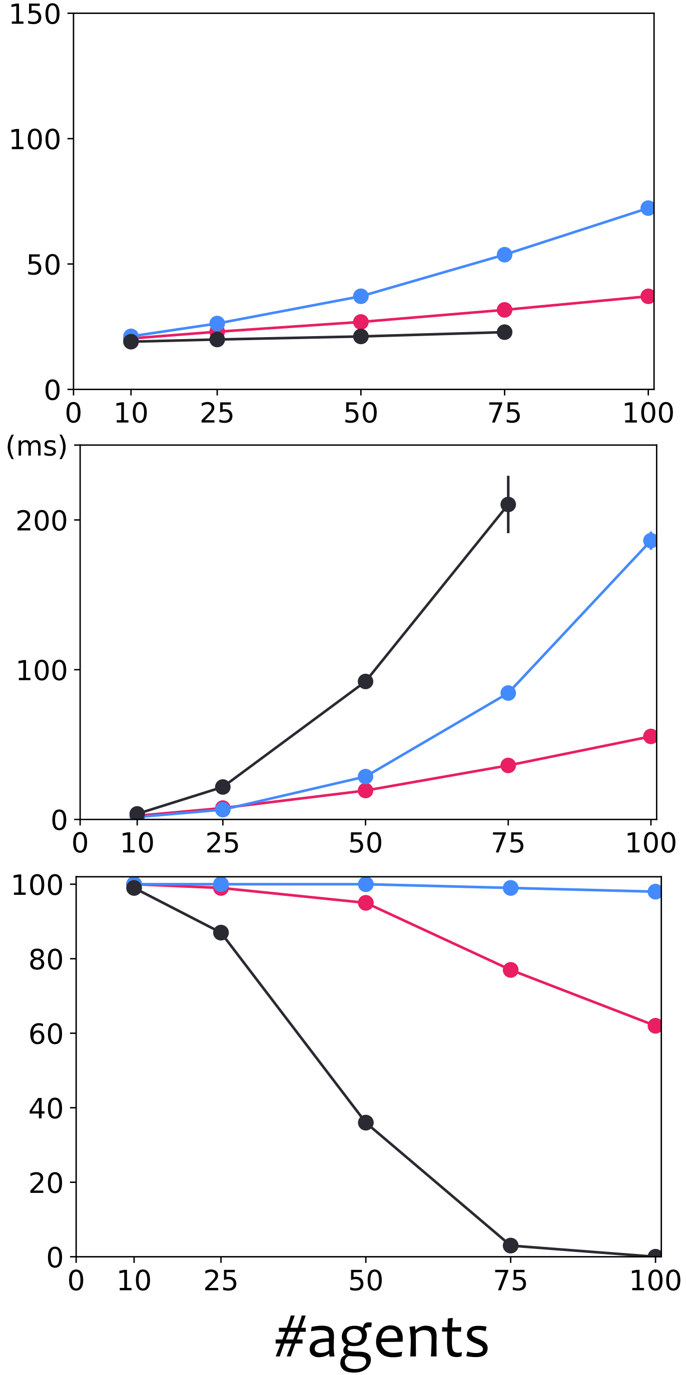
100 agents

lak105d, 25x31 [Sturtevant, 2012]

path cost
(steps)

runtime
(ms)

success
(%)



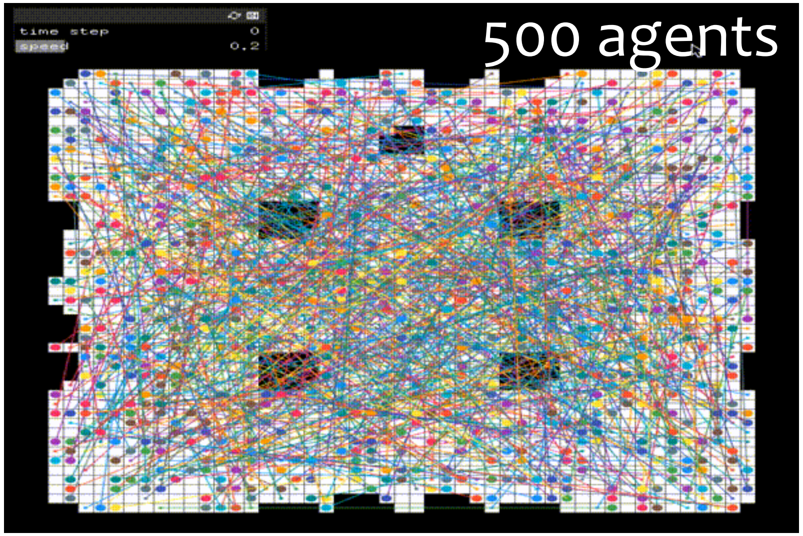
● PIBT
[Okumura et al., IJCAI19]

● Parallel Push & Swap
[Sajid et al., SoCS12]

● WHCA-10
[Silver, AIIDE05]

*average over only success cases within the solver (in 100 times)

Exp.1 one-shot MAPF

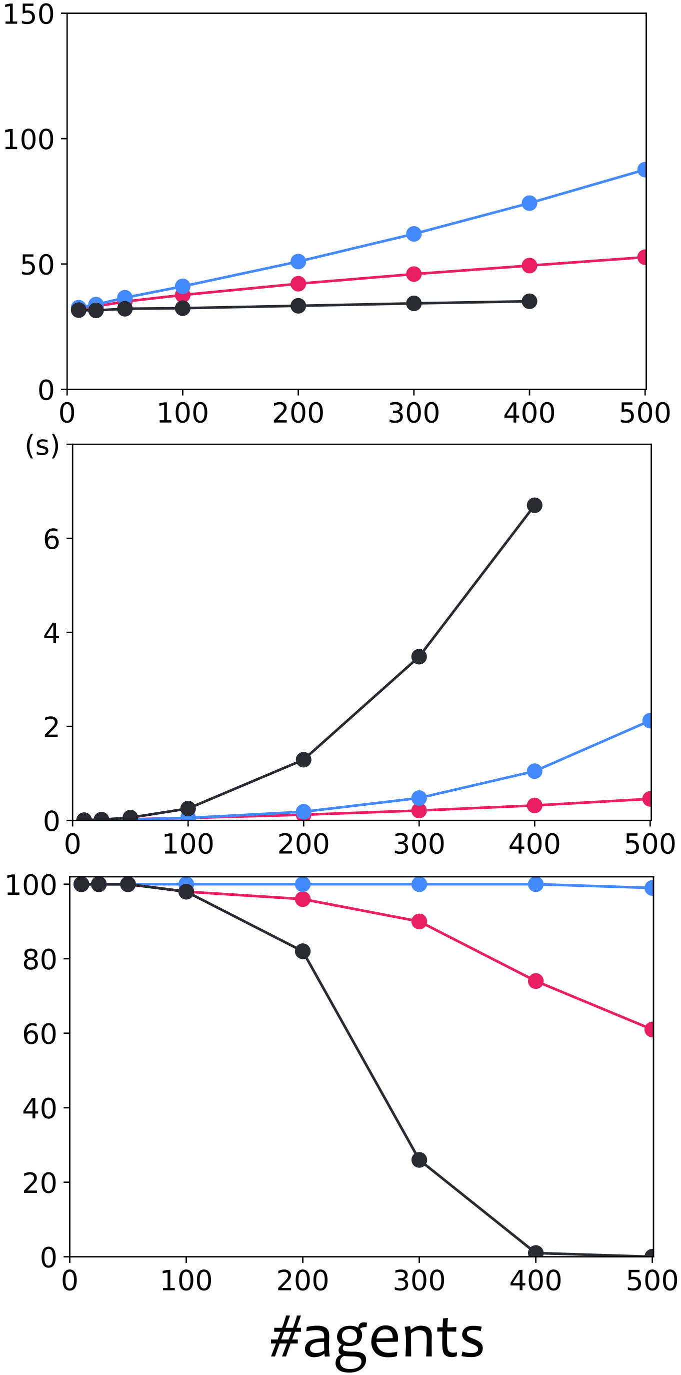


arena, 49x49 [Sturtevant, 2012]

path cost
(steps)

runtime
(s)

success
(%)



● PIBT
[Okumura et al., IJCAI19]

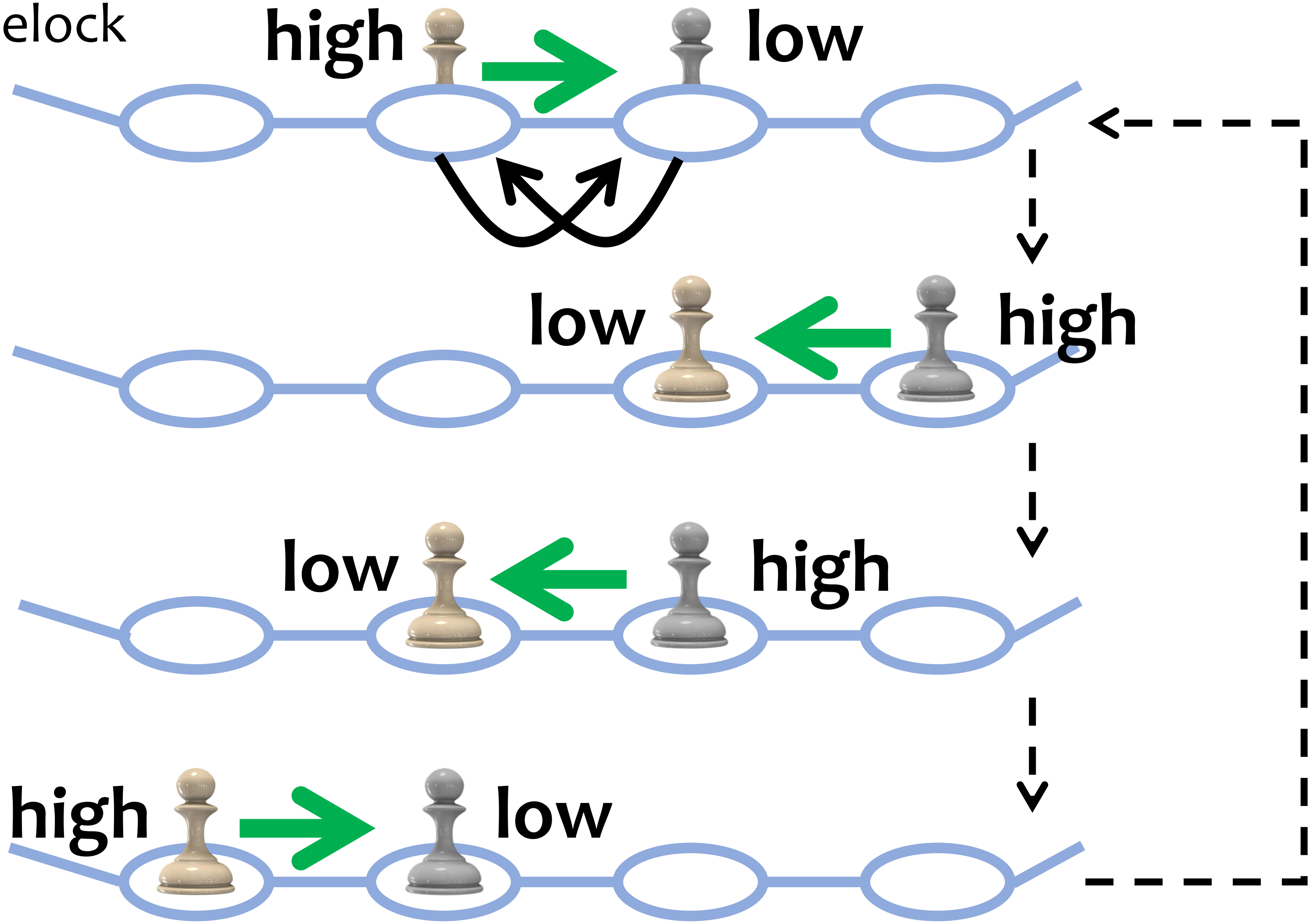
● Parallel Push & Swap
[Sajid et al., SoCS12]

● WHCA-10
[Silver, AIIDE05]

*average over only success cases within the solver (in 100 times)

Cons of PIBT

cons-1. livelock



cons-2. tree (without cycles) ← potentially cause deadlock

Computational Complexity

Proposition

Assume that PIBT performs in a centralized way.

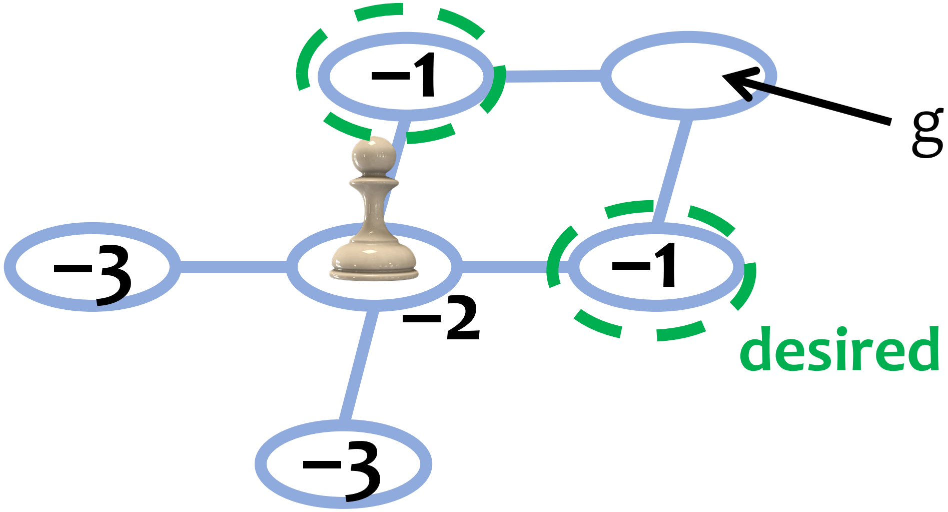
The computational complexity of PIBT **in one step** is

$$O(|A| \cdot \Delta(G) \cdot F)$$

maximum degree of G

Maximum time required to choose the next target node

e.g., let node evaluation function be –cost(current, goal)



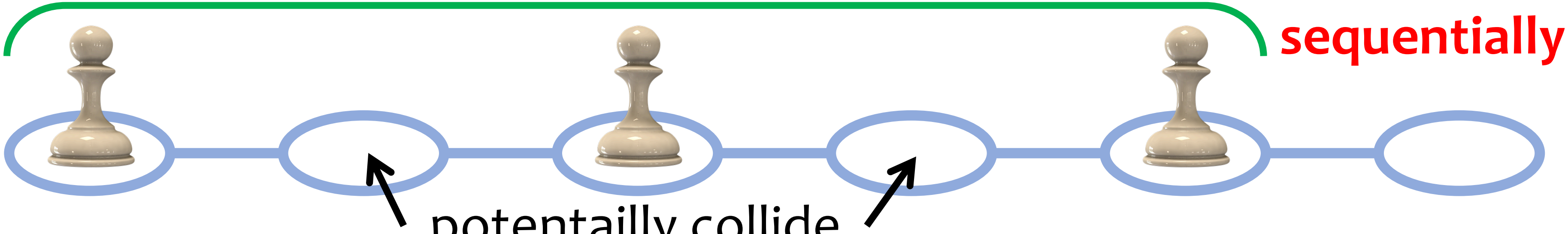
Communication

PIBT can be performed **locally** to some extent. (for decentralized)

Case 1.

interacting agents

must plan paths **sequentially**

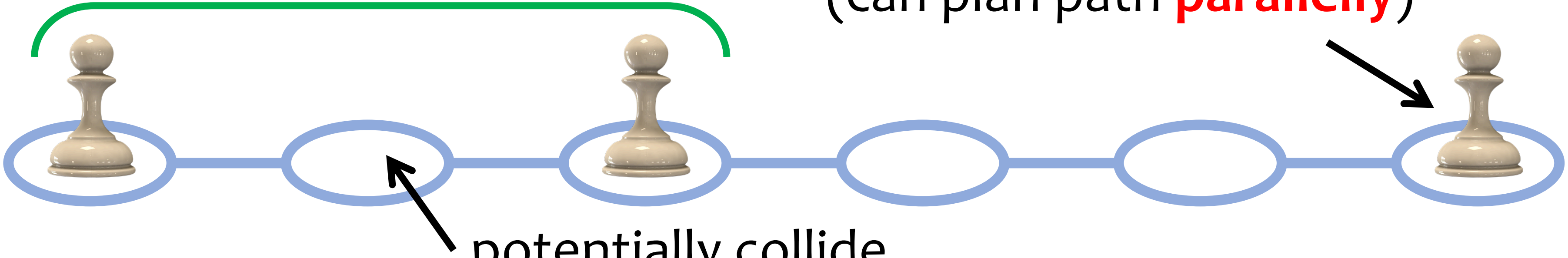


potentailly collide

Case 2.

interacting agents

unaffected by the left group (can plan path **parallelly**)



potentially collide

Communication assumption is sufficient that two agents in close proximity can talk directly and utilize multi-hop communication.

Adapt Decentralized Fashion

PIBT relatively smoothly adapts to decentralized fashion.
This transition requires three steps.

✓ detect interacting groups ⟵ might be difficult

✓ collect priorities

Updating rule of priorities relaxes this effort,
e.g., take care only when $\eta_i(t)$ (primally term) becomes zero.

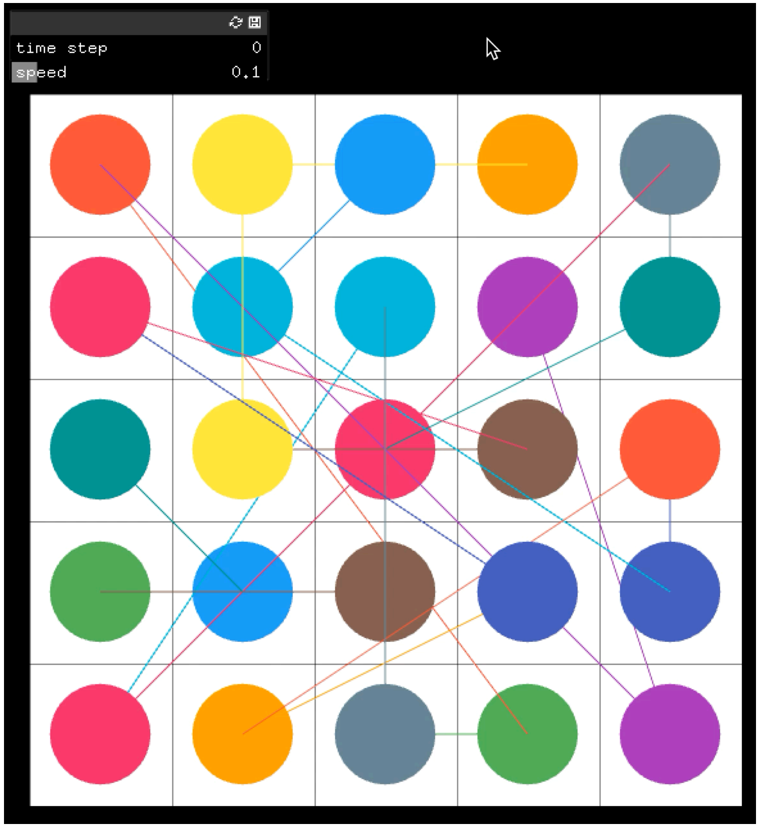
✓ plan paths sequentially within groups

Priority inheritance and backtracking is performed
by **token passing**.

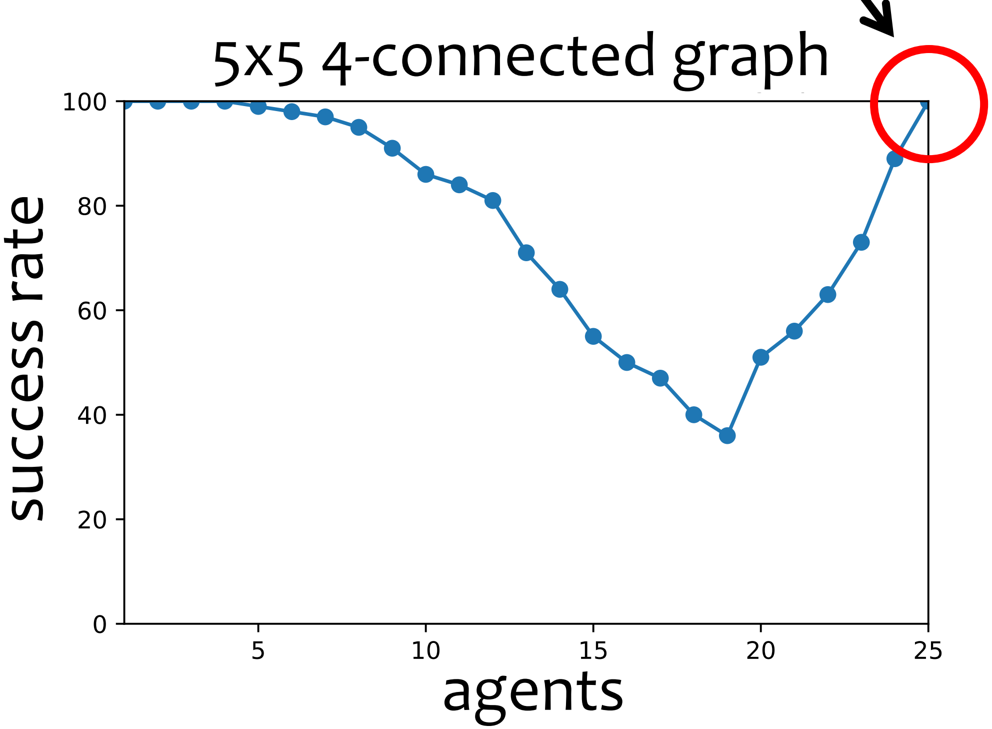
message counts: $O(|A|)$

PIBT on MAPF in Simple Grids

I don't know why but PIBT seems to solve MAPF in **100% without any vacant** in **simple grids**.



```
time step     0
speed       0.1
```



In 8x8 4-connected graph, we obtain similar results.