Pathfinding and Routing
NAIL137

Jiří Švancara

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

Sub-optimal solvers

- Adapt optimal solvers
- Rule-based (for specific graphs)
- “Other approaches“

ECBS

Enhanced CBS

- Replace both top level and low level with bounded sub-optimal search

- Focal search
  - Keep OPEN and FOCAL
  - FOCAL consists of nodes from open with values of $f(n) \leq w*f(\text{best in OPEN})$
  - Select from FOCAL based on heuristic only

EECBS

Explicit Estimation CBS

- Aims to improve top level Focal search
  - Distance-to-go in ECBS is number of conflicts
- Use better estimate of distance-to-go
- Explore best in OPEN sooner to increase the bound

- Low level keeps the same Focal search

J. Li, W. Ruml, S. Koenig. EECBS: A Bounded-Suboptimal Search for Multi-Agent Path Finding. AAAI, 2021

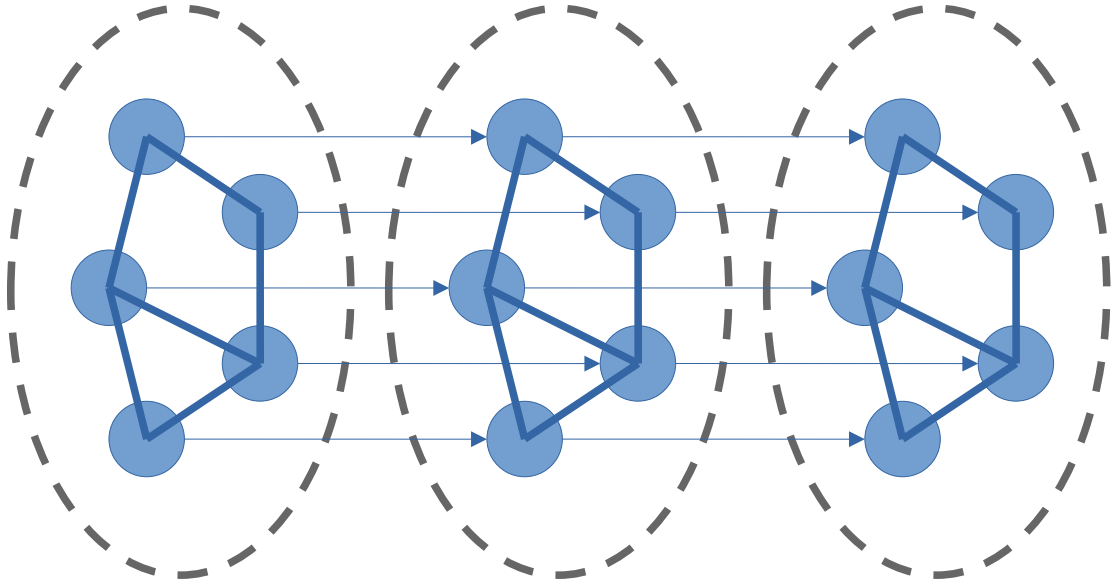
Reduction to SAT - sum of costs

- Remove bounds on sum of costs
- Keep the same preprocessing as in sum of costs optimal variant
  - Fewer variables than makespan
  - Less restriction than sum of costs

P. Surynek, A. Felner, R. Stern, E. Boyarski. Sub-Optimal SAT-Based Approach to Multi-Agent Path-Finding Problem. SOCS, 2018

Reduction to SAT - alternative expansion

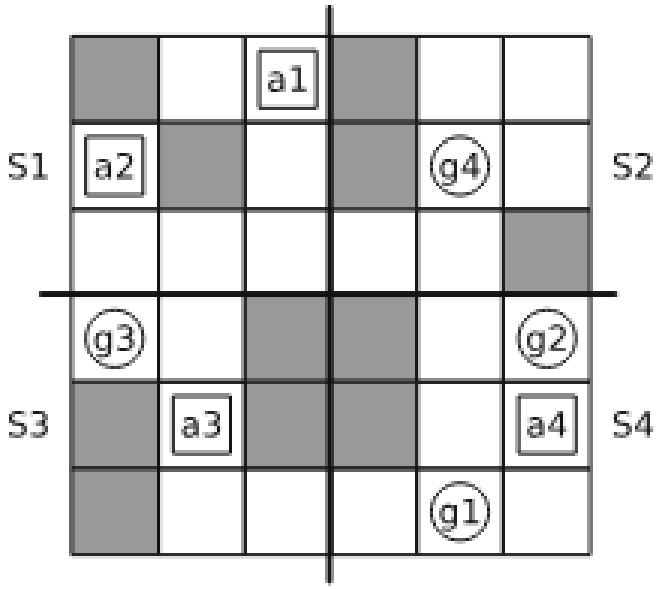
- Different encoding
- Graph is expanded by number of revisits instead of time



DMAPF

Distributed MAPF

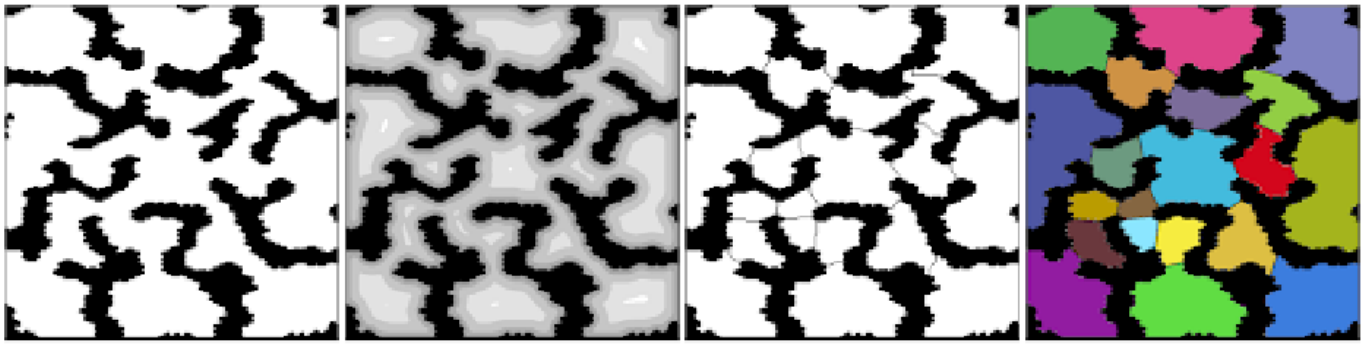
- Split map into segments
- For each agent, create abstract plan over segments
- Small dense instances → use reduction-based solvers (ASP in the paper)



P. Pianpak, T. Son. DMAPF: A Decentralized and Distributed Solver for Multi-Agent Path Finding Problem with Obstacles. ICLP, 2021

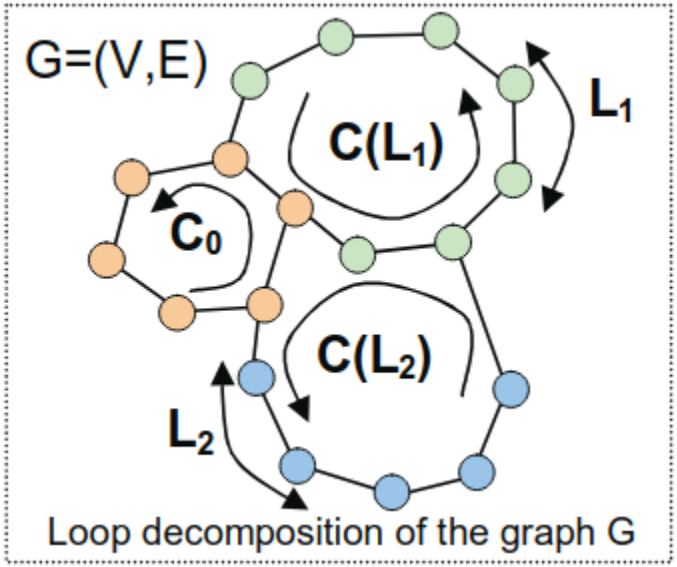
DMAPF - challenges

- How to select abstract plan
  - Keep track of congestion
- How to hand over agents
  - Synchronize time steps (inefficient!)
- How to split map into segments
  - Grid
  - Water level decomposition



BIBOX

- Assumes bi-connected graph with 2 free vertices
- Places agents ear by ear (1 free vertex)
- Reorders agents in the final loop (2 free vertices)



P. Surynek. A Novel Approach to Path Planning for Multiple Robots in Bi-connected Graphs . ICRA, 2009

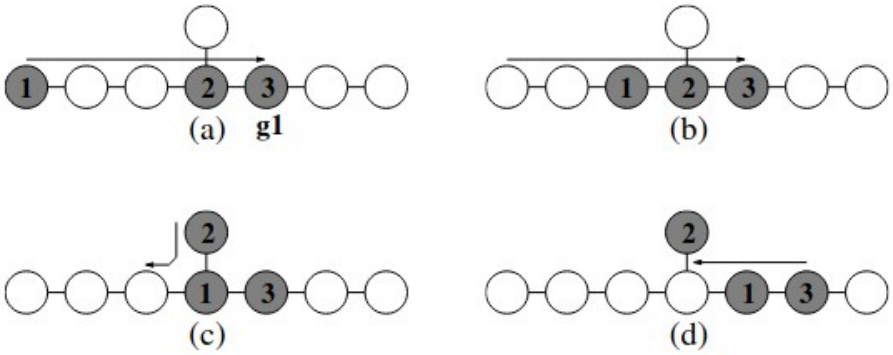
Push and swap

- Move along shortest path
- Push – move lower priority agents out of the way
- Rotate – rotate with higher priority agents

- Later shown to be incomplete!

Algorithm 1 PUSH_AND_SWAP($\mathcal{G}, \mathcal{R}, \mathcal{S}, \mathcal{T}$)

```
1: A ← S
2: Π* ← {A}
3: U ← ∅
4: for all r ∈ R do
5:    while A[r] ≠ T[r] do
6:       if PUSH(Π*, G, A, T, r, U) == FALSE then
7:          if SWAP(Π*, G, A, T, r, U) == FALSE then
8:             return ∅ (i.e., Failure)
9:    U ← U ∪ A[r]
10: return Π* (i.e., Success)
```



R. Luna, K. Bekris. Push and Swap: Fast Cooperative Path-Finding with Completeness Guarantees. IJCAI, 2011

Push and rotate

- Find unsolvable instances
- Similar to push and swap
- Added rotate along cycles (similar to BIBOX)

B. Wilde, A. ter Mors, C. Witteveen. Push and Rotate: a Complete Multi-agent Pathfinding Algorithm. JAIR, 2014

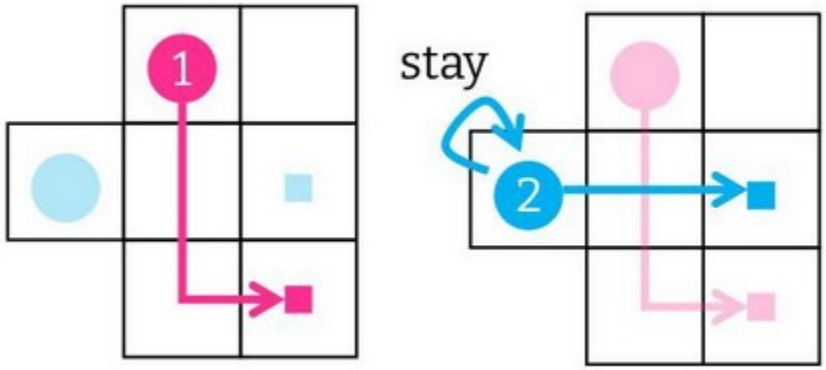
PP

Prioritized planning

1) Assign priorities to agents

2) Plan based on priorities, previously planned agents are moving obstacles

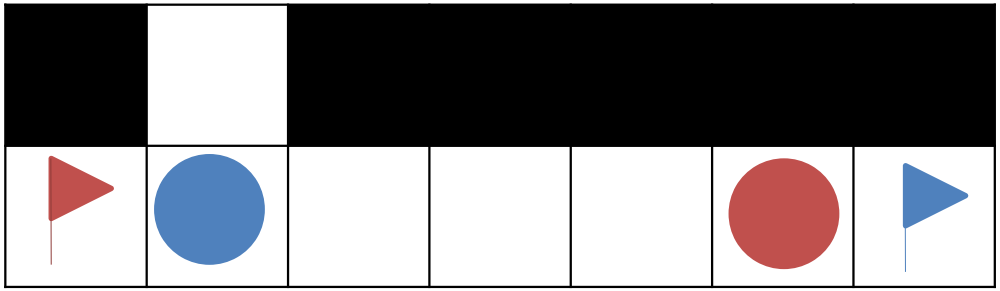
3)(A* or SIPP)



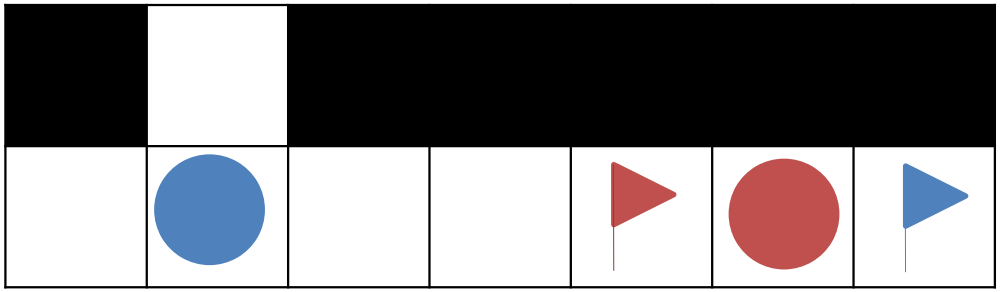
D. Silver. Cooperative Pathfinding. AIIDE 2005

PP

- The order matters!

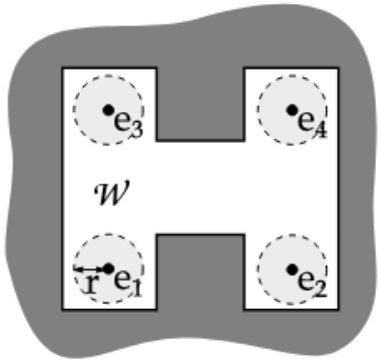


- Incomplete regardless of the order!

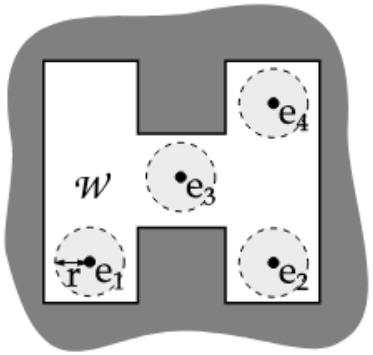


PP - well formed infrastructure

- Well formed
  - There is a path between start-goal that does not pass through other start or goals
- Complete regardless of the order!



(a) Valid infrastructure: The workspace $\mathcal{W}$ and endpoints $\{e_1, e_2, e_3, e_4\}$ for robots having radius r form a valid infrastructure.



(b) Non-valid Infrastructure: The workspace $\mathcal{W}$ and endpoints $\{e_1, e_2, e_3\}$ do not form a valid infrastructure because there is no path from e_1 to e_2 with $2r$-clearance to e_3 for a robot having radius r.

MAPF-LNS

- Large neighborhood search
  - Given a valid solution, try to improve it

  1) Select part of solution to be deleted (destroy operator)

  2) Plan the missing part, do not change the rest (repair operator)
- Is anytime algorithm

J. Li, Z. Chen, D. Harabor, P. Stuckey, S. Koenig. Anytime Multi-Agent Path Finding via Large Neighborhood Search. IJCAI, 2021.

MAPF-LNS - destroy operator

- What part to destroy (neighborhood)
  - Subset of agents
    1) The ones that are far from optimum
    2) The ones that interact a lot
    3) Random

MAPF-LNS - repair operator

- How to plan for the unplanned agents?
  - Any MAPF algorithm

  1) CBS – optimal but slow
  2) PP – fast but sub-optimal

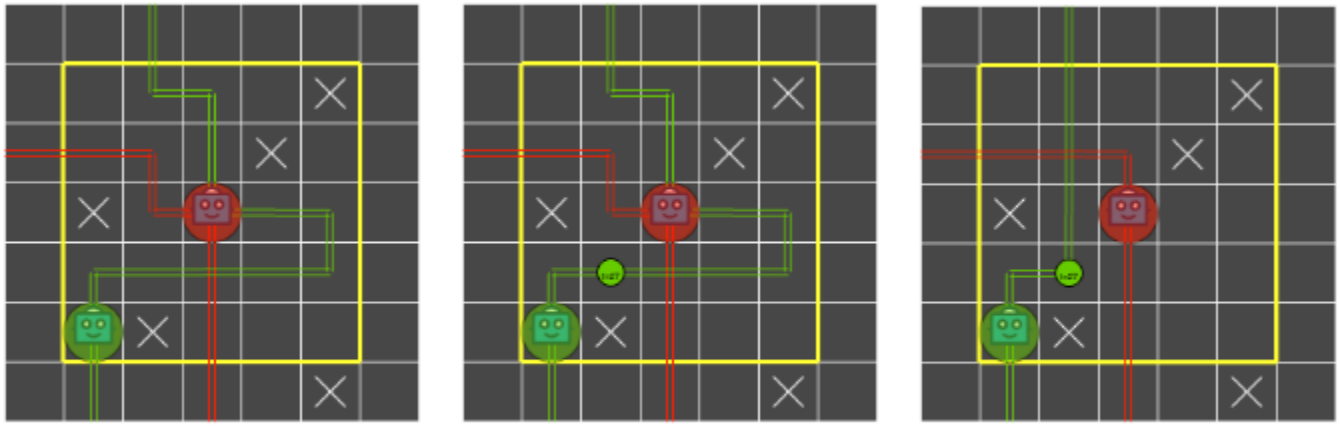
MAPF-LNS2

- Not necessary to start with a valid solution
- Allow conflicts at beginning, try to repair those first
- Then continue with classic MAPF-LNS

J. Li, Z. Chen, D. Harabor, P. Stuckey, S. Koenig. MAPF-LNS2: Fast Repairing for Multi-Agent Path Finding via Large Neighborhood Search. AAAI, 2022.
https://github.com/Jiaoyang-Li/MAPF-LNS2

MAPF-LNS2 - challenges

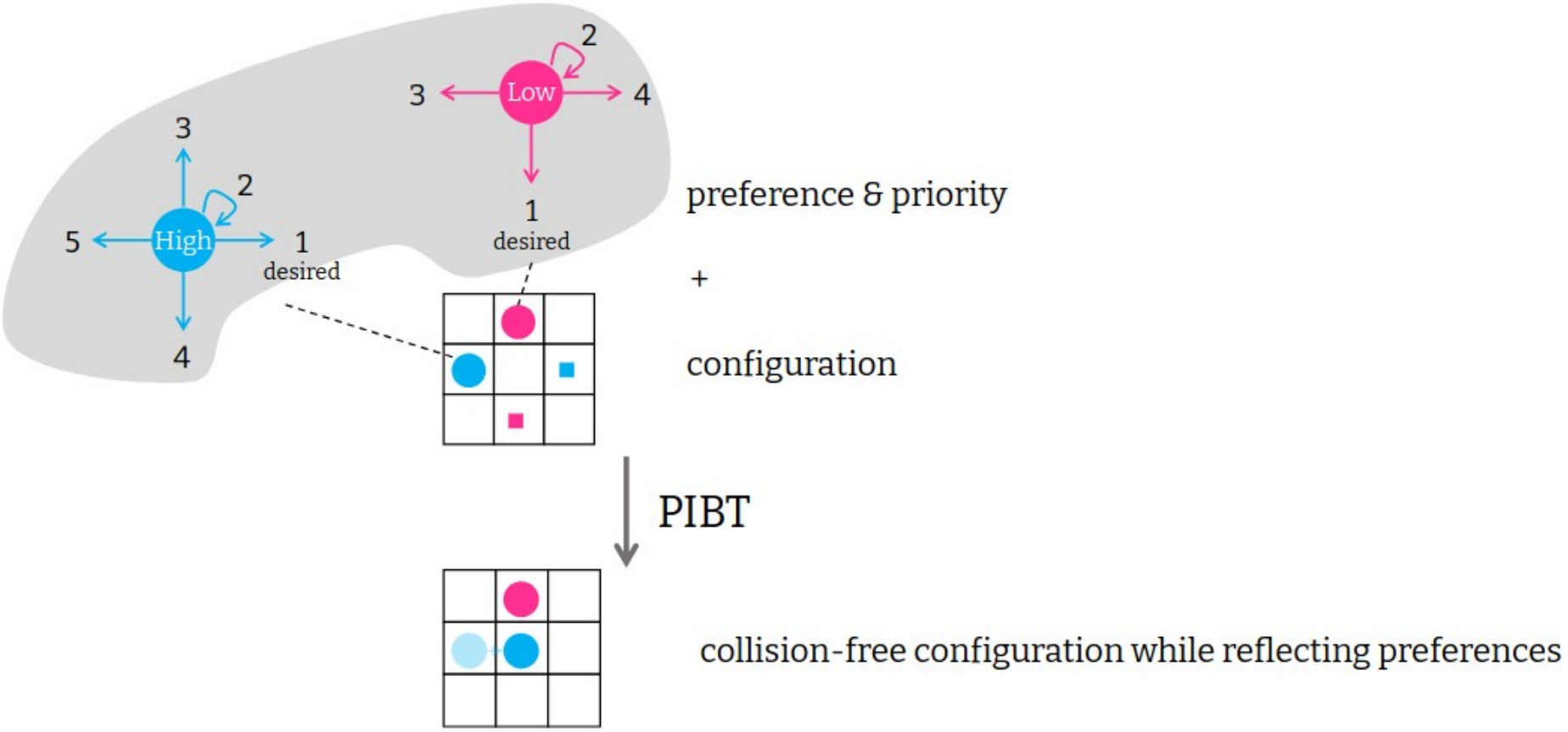
- “Random” is a good selection
  - Different selection
  - Room for ML
- Change what a neighborhood is



(a) Sub-map extraction **(b)** Timestep detection ($t = 27$) **(c)** Replanned paths

PIBT

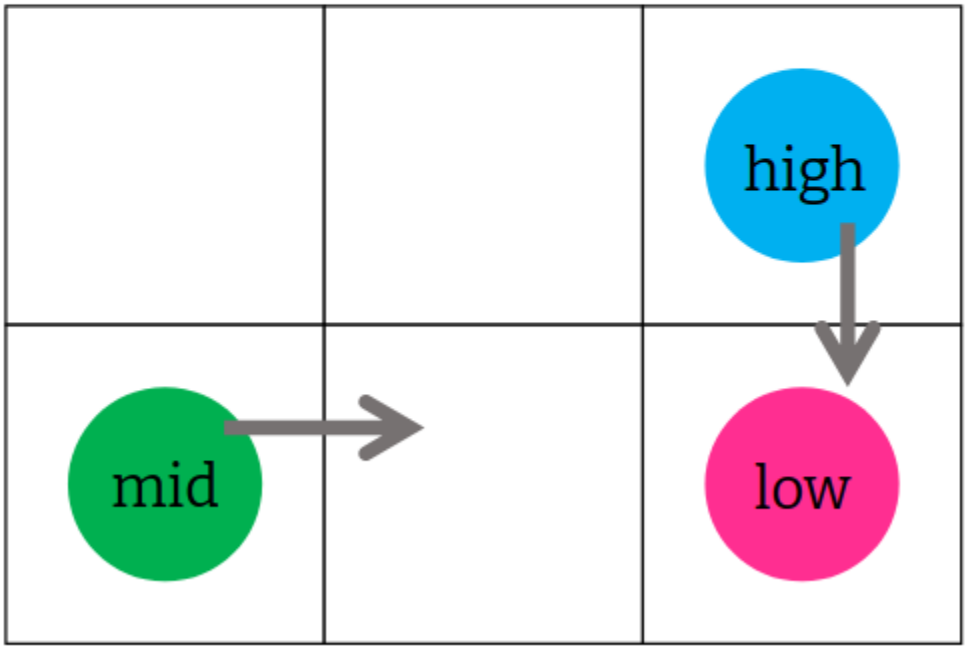
- Priority inheritance with backtracking



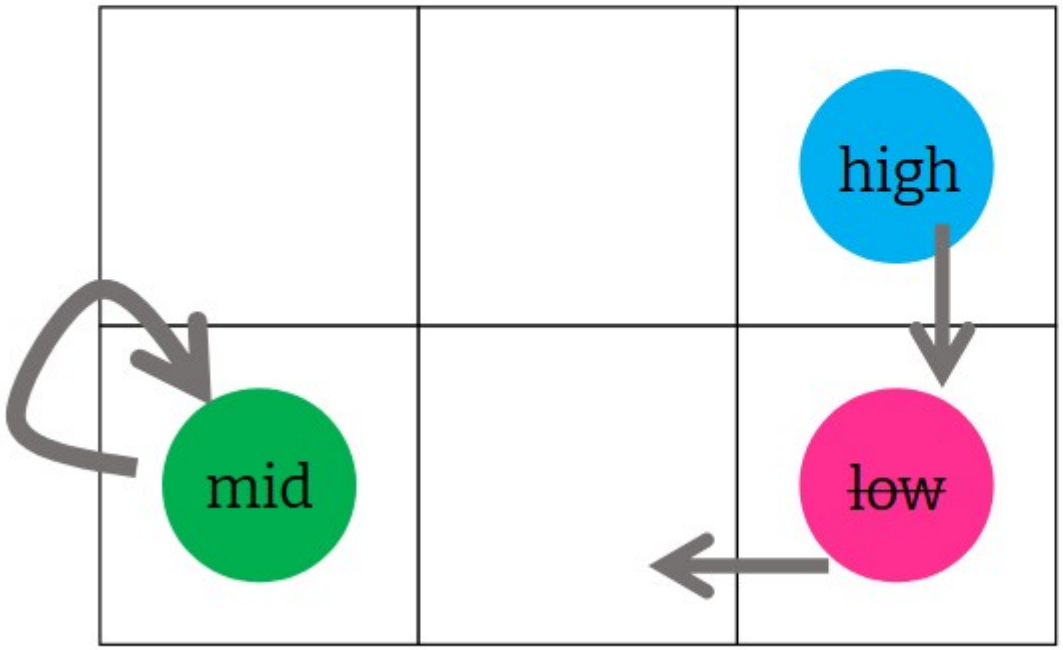
K. Okumura, M. Machida, X. Défago, Y. Tamura. Priority inheritance with backtracking for iterative multi-agent path finding. Artif. Intell., 2022.

How PIBT works – 1/4

greedy assignment with priorities is incomplete



How PIBT works – 2/4



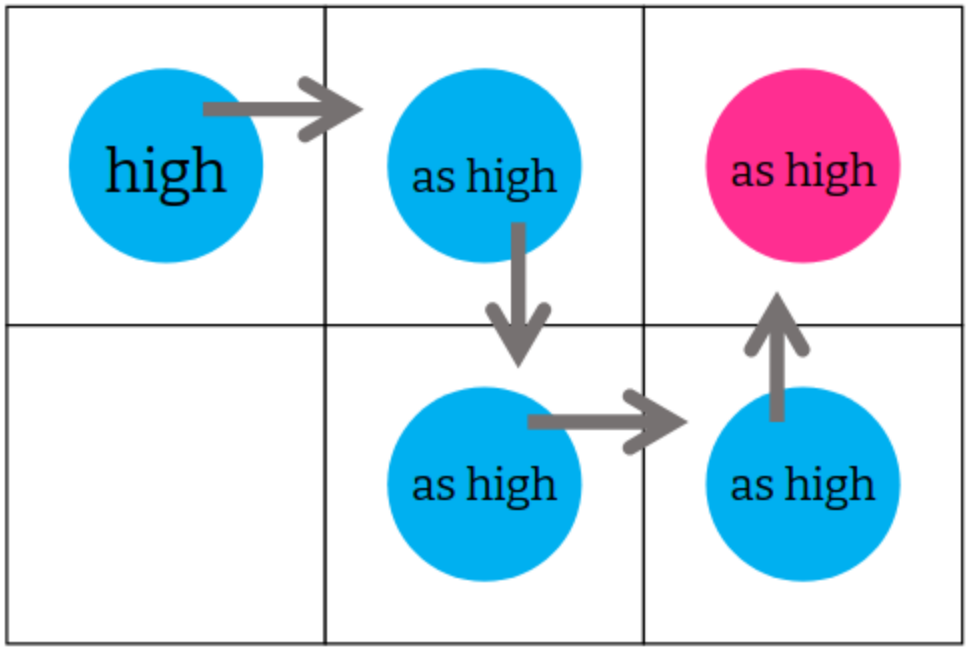
priority inheritance

Sha+ IEEE Trans Comput-90

as high

How PIBT works – 3/4

but still not feasible

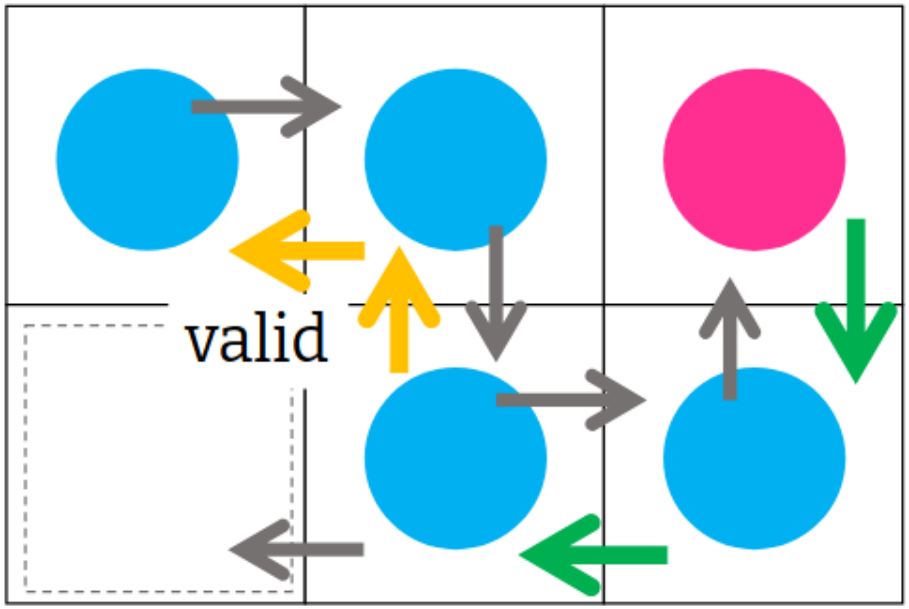


How PIBT works – 4/4

introduce backtracking

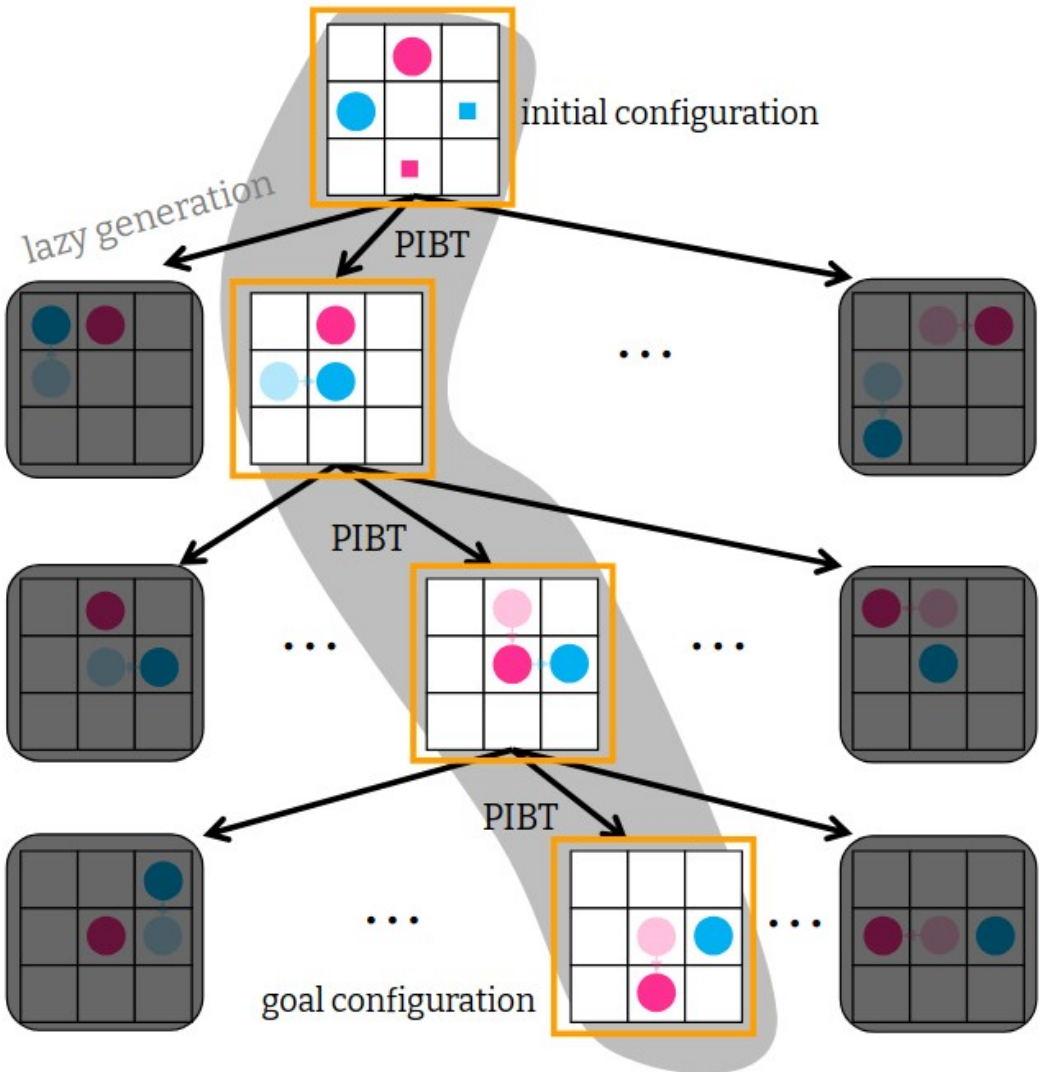
valid — You can move

invalid — You must re-plan, I will stay



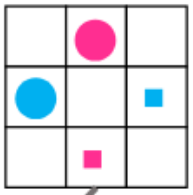
✓ always generate collision-free configurations

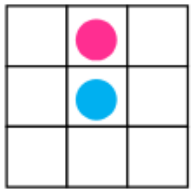
LaCAM



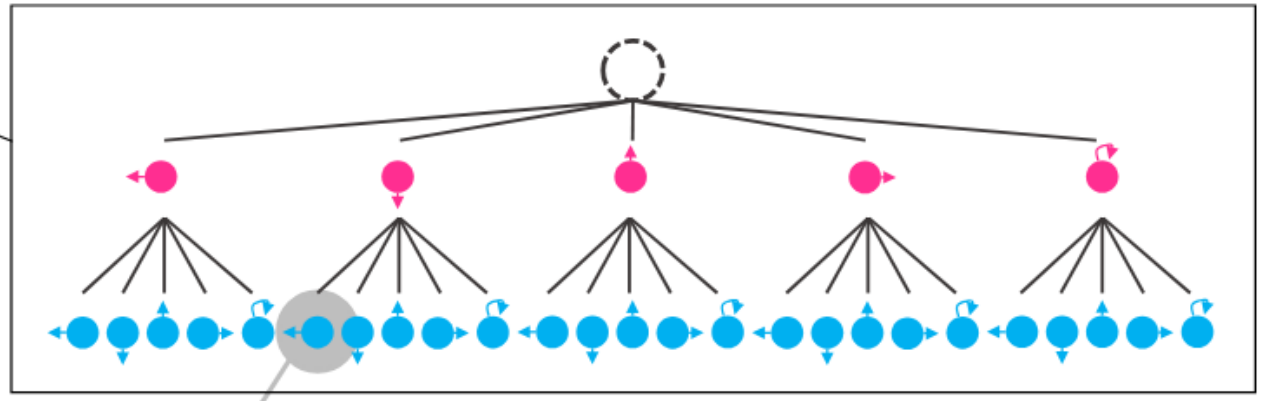
K. Okumura. LaCAM: Search-Based Algorithm for Quick Multi-Agent Pathfinding. AAAI, 2023

Lazy constraints addition – 1/4

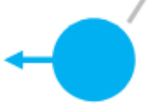




invoked multiple times
during the search

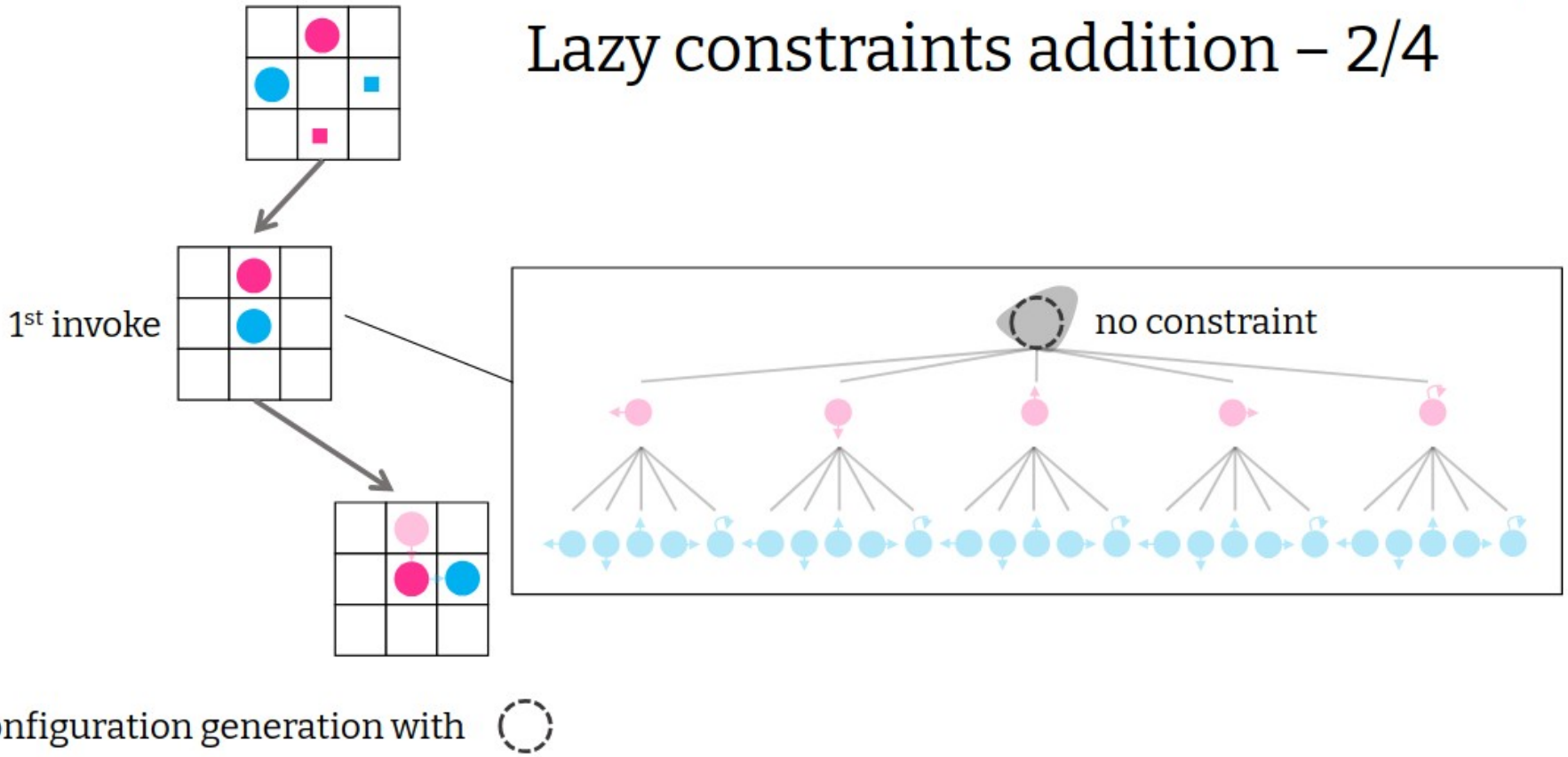


constraint tree
(maintained *implicitly*)

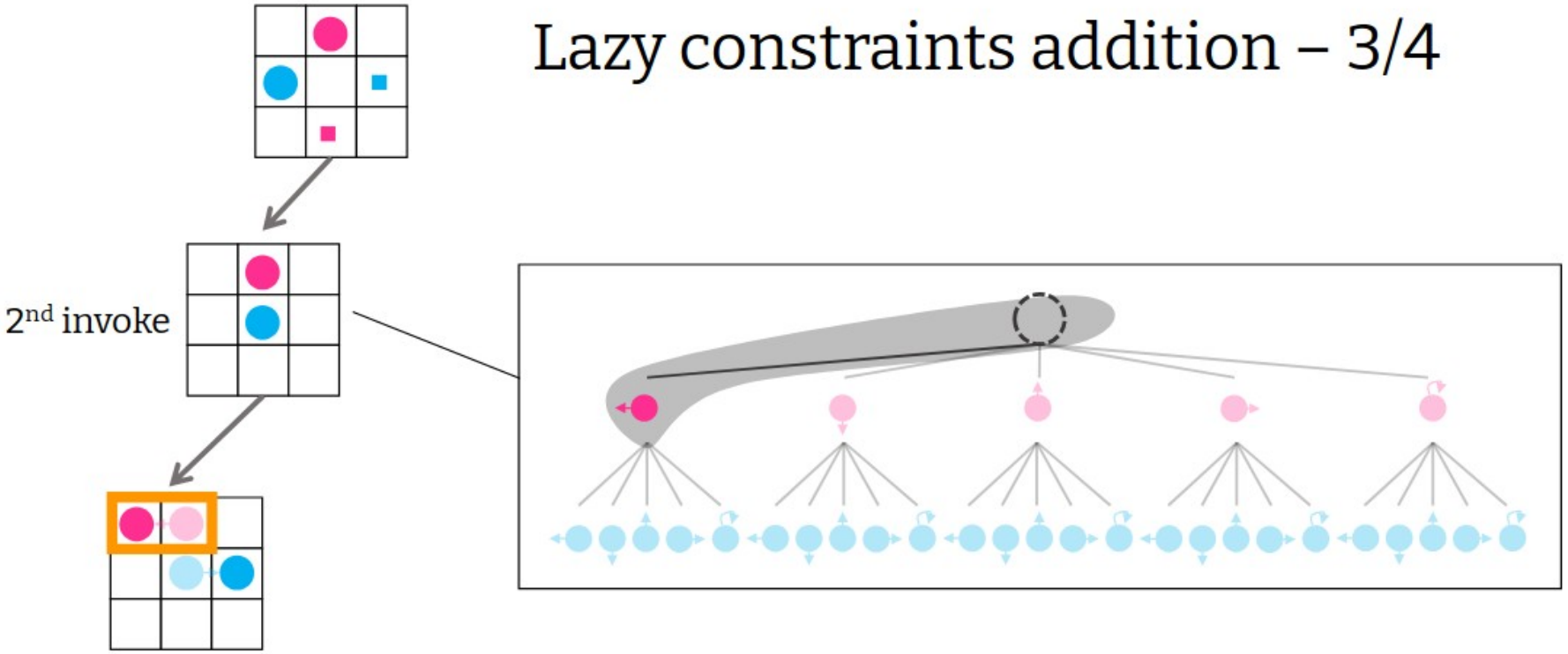


must go left in the next config.

LaCAM

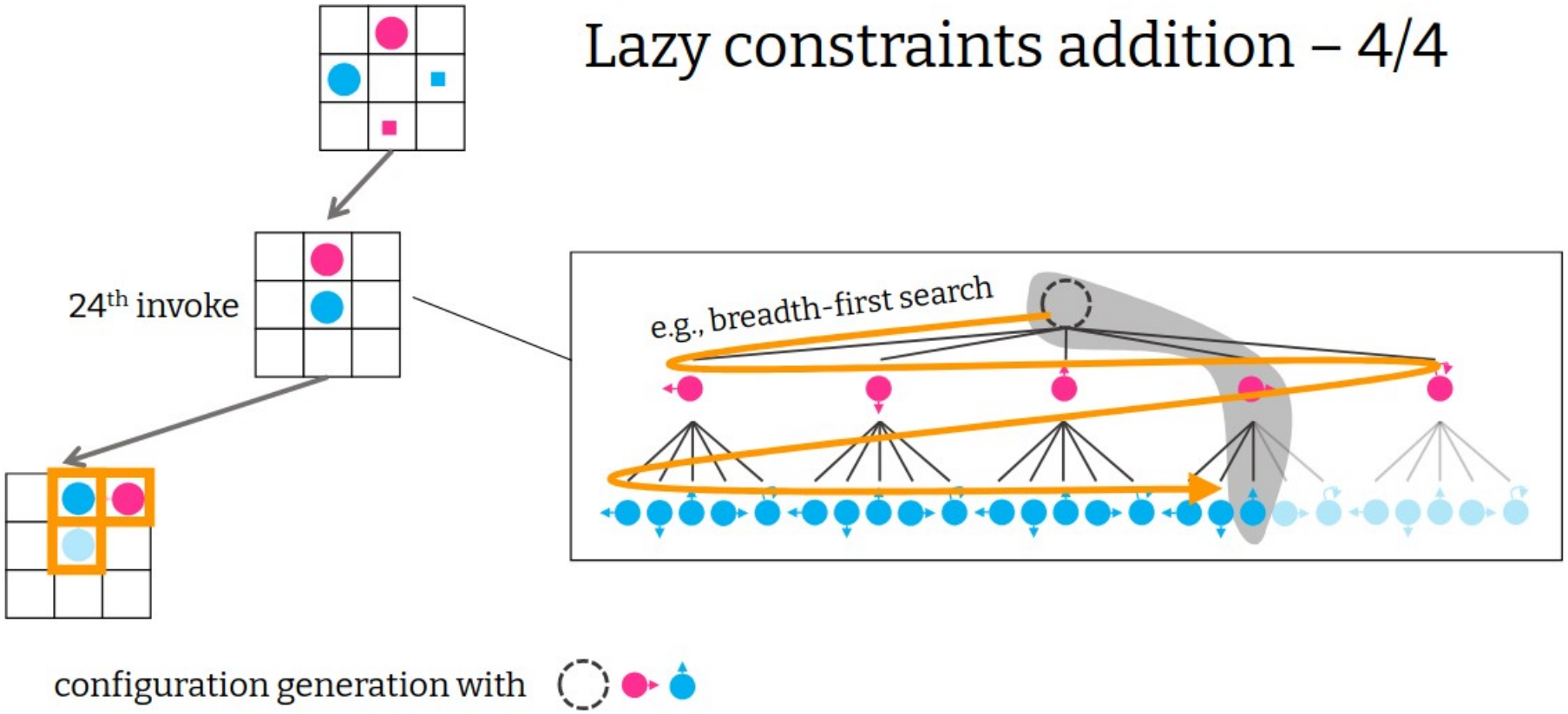

Lazy constraints addition – 2/4
1st invoke
no constraint
configuration generation with

Lazy constraints addition – 3/4



configuration generation with 

Lazy constraints addition – 4/4



LaCAM