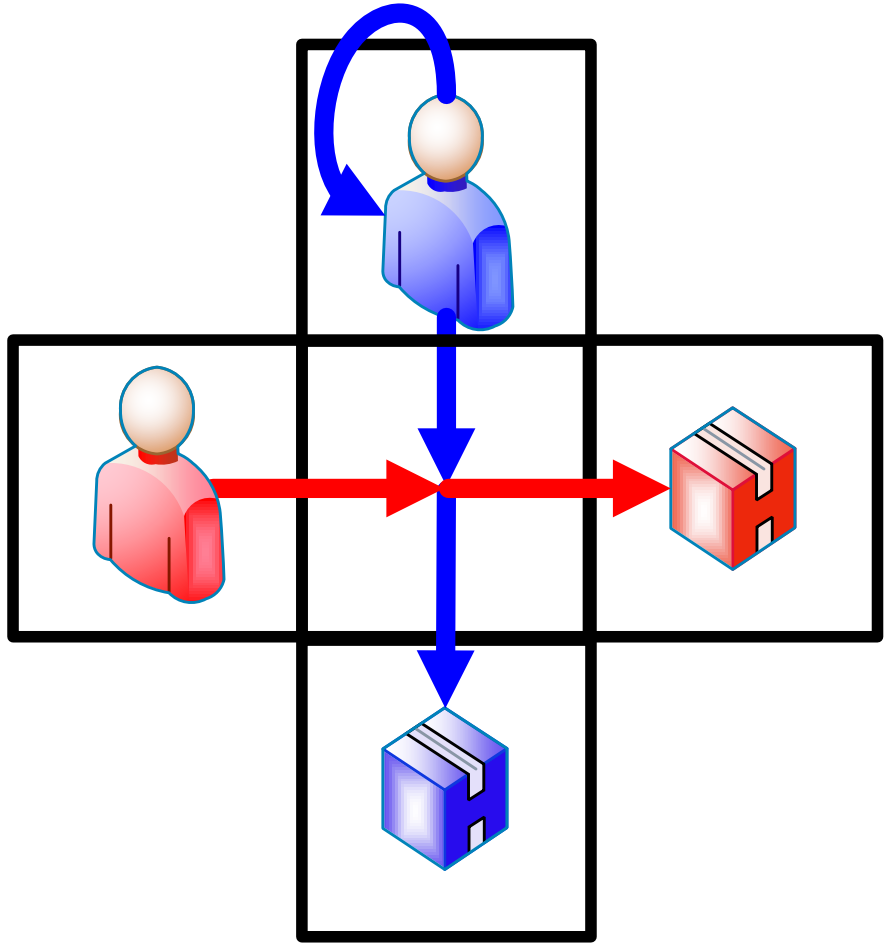
Optimal Target Assignment and Path Finding for Teams of Agents

Hang Ma Sven Koenig
University of Southern California

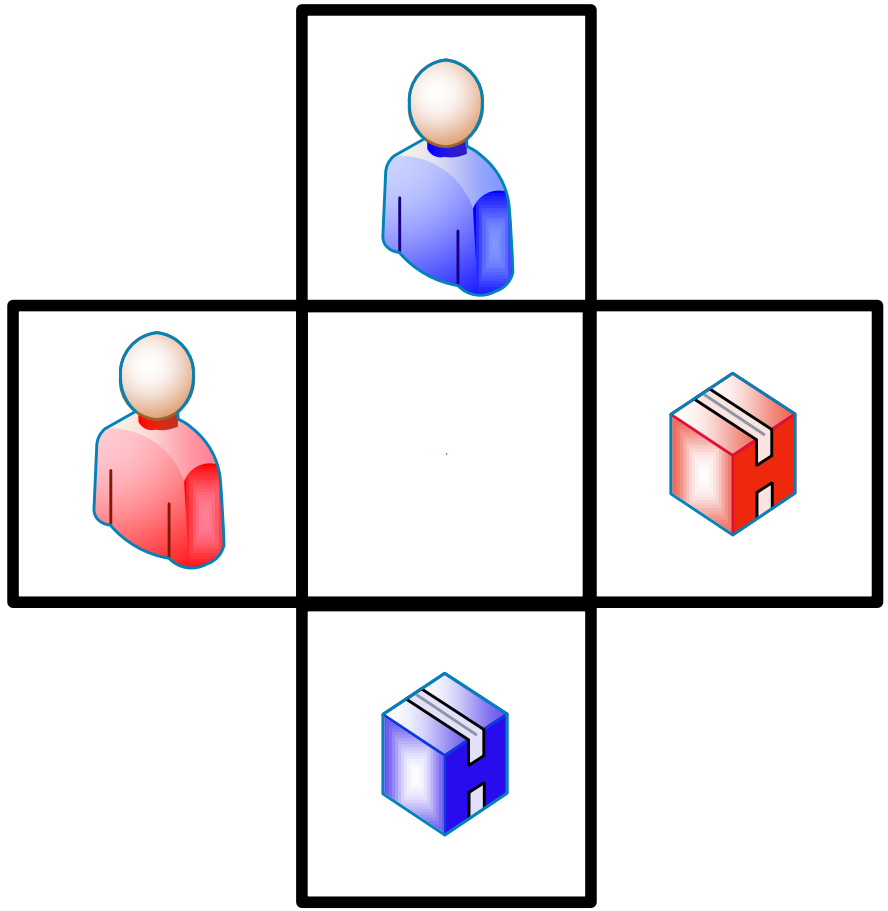
May 12, 2016
AAMAS, Singapore

Multi-Agent Path Finding (MAPF)

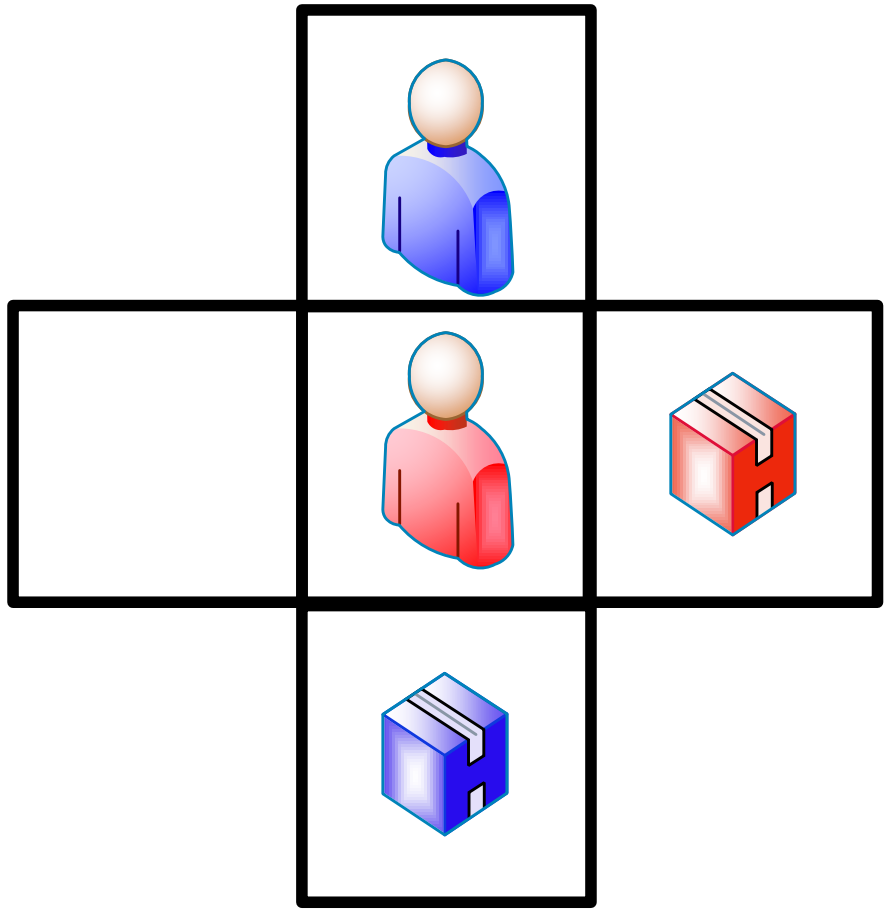
Find collision-free paths for all agents from their start vertices to their targets.



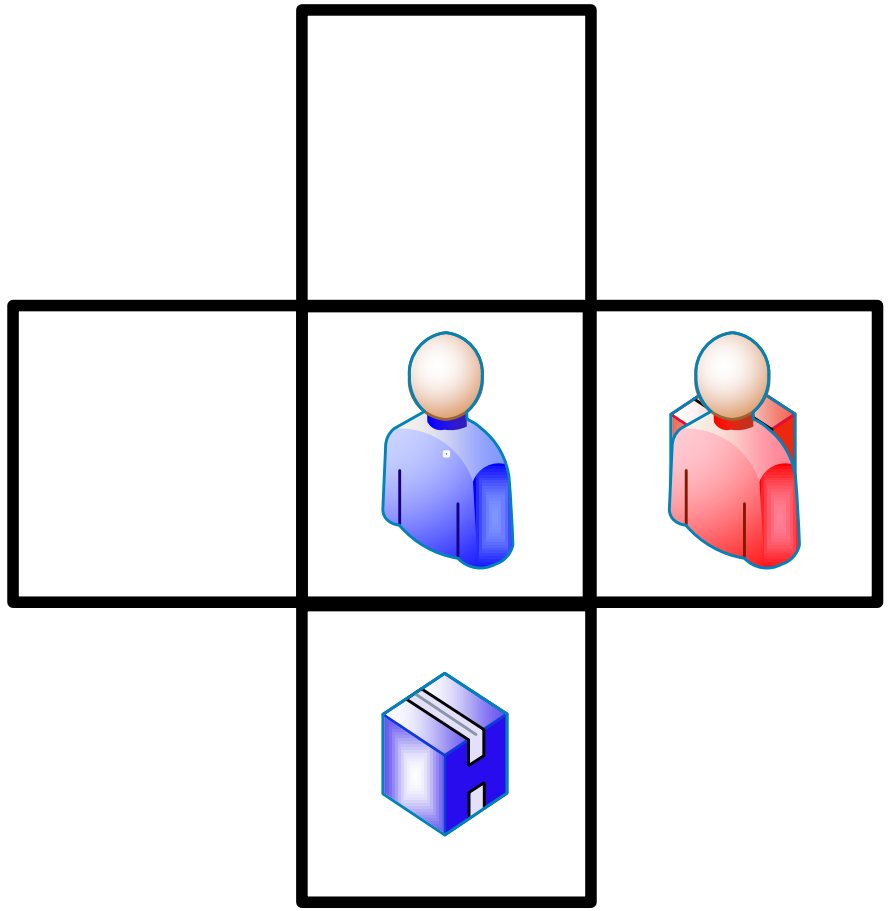
Time Step 0



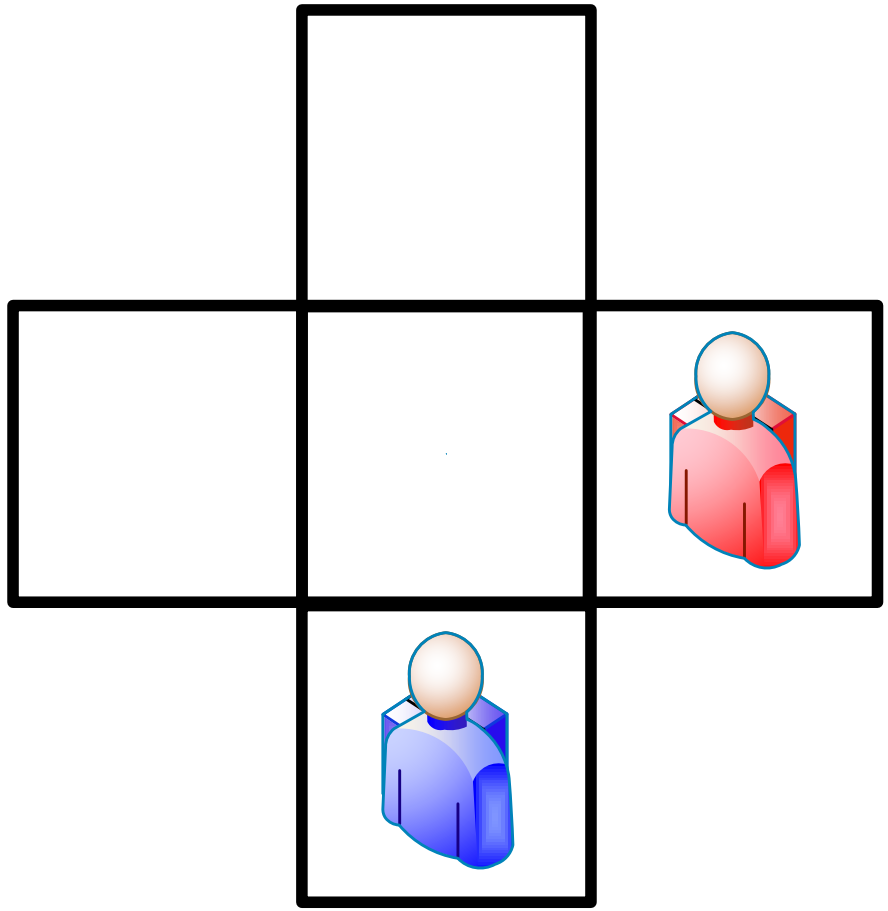
Time Step 1



Time Step 2

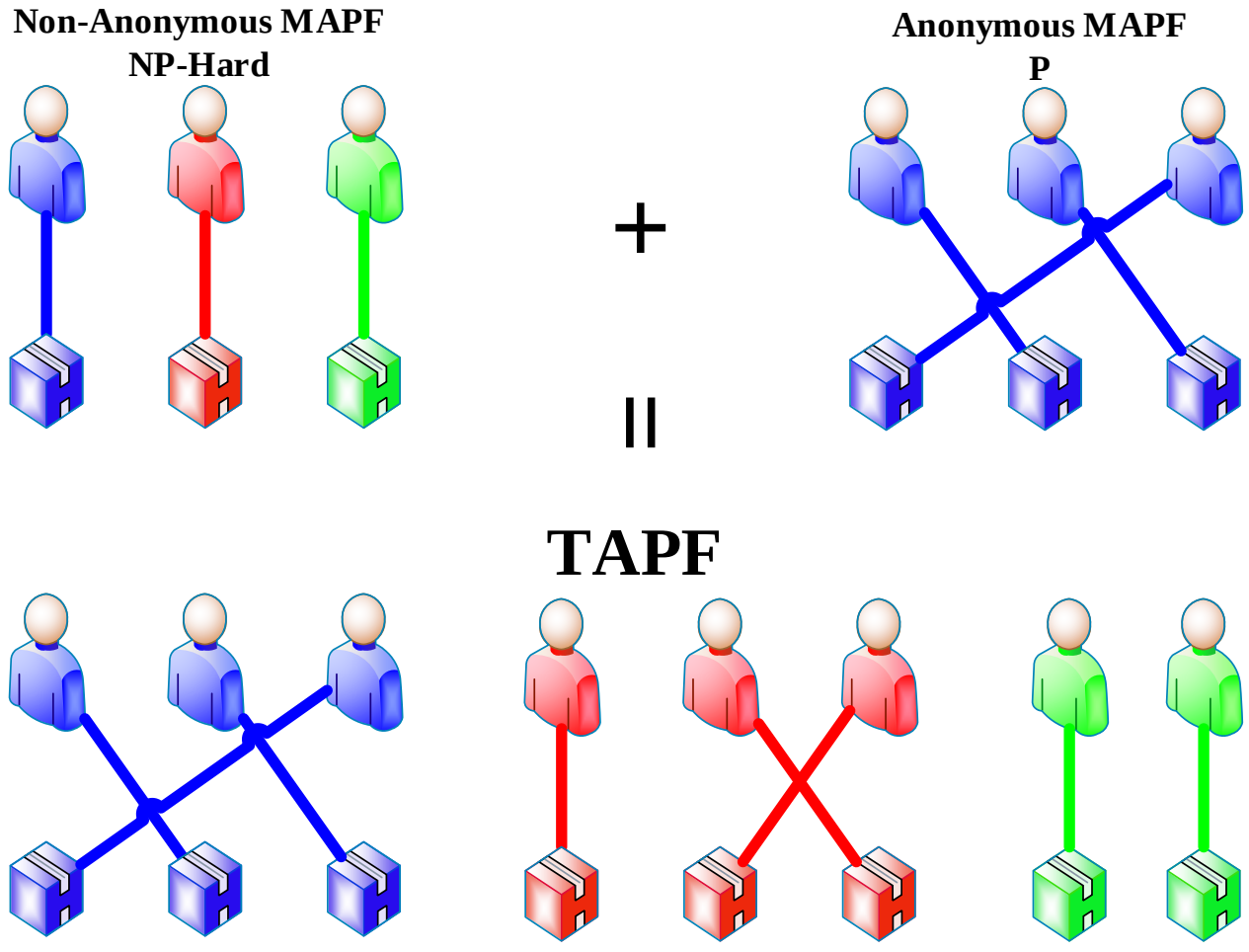


Time Step 3



Target Assignment and Path Finding (TAPF)

TAPF — A mix of non-anonymous MAPF and anonymous MAPF.



An Example of TAPF

Team0: Agents that move from the stations to the storage locations.
Team1: Agents that move from the storage locations to Station 1.
Team2: Agents that move from the storage locations to Station 2.
Team3: Agents that move from the storage locations to Station 3.
. . .

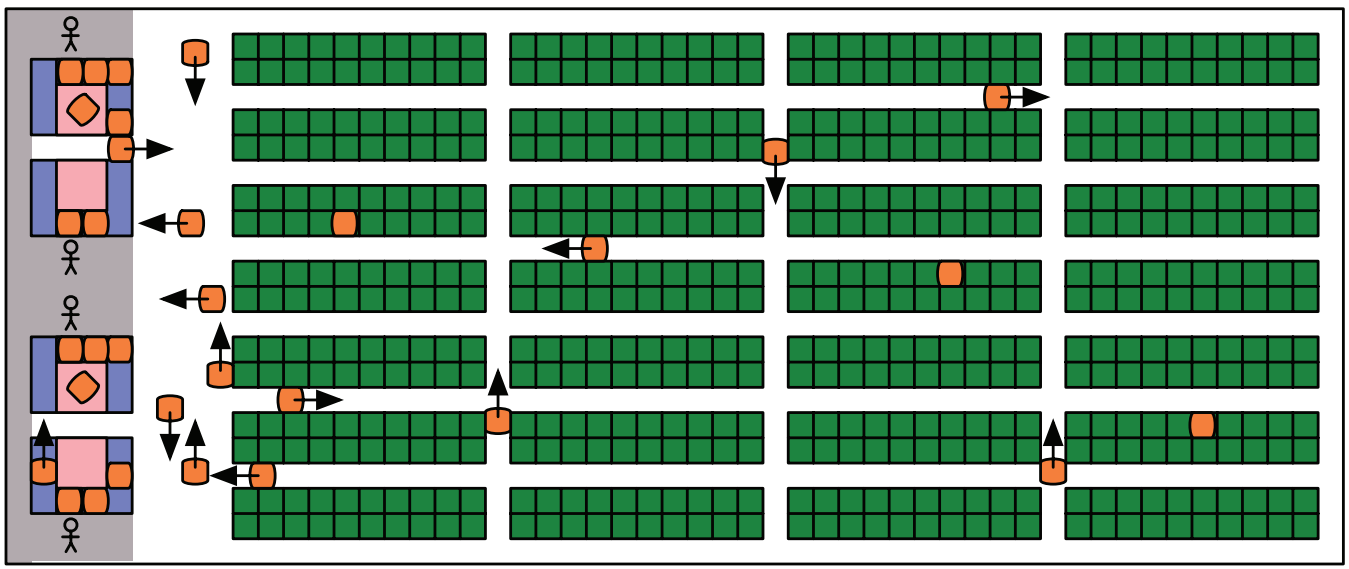


Figure: Kiva (Amazon Robotics) Automated Warehouse System[1].

[1] P. R. Wurman, R. D'Andrea, and M. Mountz. "Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses". In: *AI Magazine* 29.1 (2008), pp. 9–20.

Task of TAPF

Find the **target assignments** and **collision-free paths** that minimize the **makespan**.
The **makespan** = the earliest time step when all agents have reached their targets.

How to Solve TAPF?

Ideas from:

- Conflict-Based Search (CBS)[2] for solving **non-anonymous MAPF** (**NP**-hard).
- Max-flow algorithm[3] for solving **anonymous MAPF** (**P**).

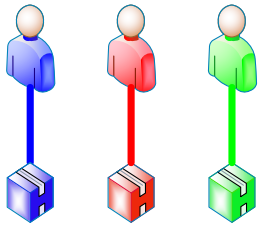
[2] G. Sharon et al. "Conflict-based search for optimal multi-agent pathfinding". In: *Artificial Intelligence* 219 (2015), pp. 40–66.

[3] J. Yu and S. M. LaValle. "Multi-agent Path Planning and Network Flow". In: *Algorithmic Foundations of Robotics X, Springer Tracts in Advanced Robotics*. Vol. 86. 2013, pp. 157–173.

Conflict-Based Min-Cost Flow (CBM) for TAPF

Our algorithm — Conflict-Based Min-Cost Flow (CBM) = Conflict-Based Search (CBS) + (min-cost) max flow

CBS for Non-Anonymous MAPF

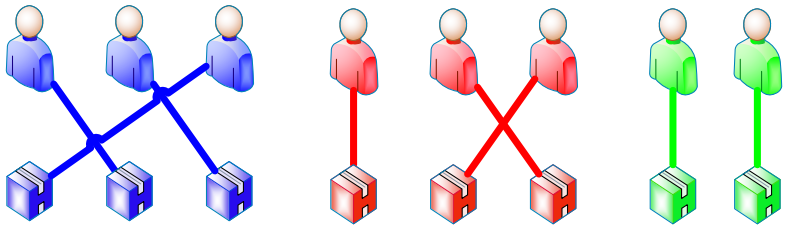


CBS:

1. Find *paths* for each single agent separately.
2. Look for **collisions** in *paths*.
3. If there is a **collision** between a_1 and a_2:
   Option 1 or Option 2 to avoid collision.

- **Collision**: $\langle Agent\ a_1, Agent\ a_2, Location\ x, Time\ t \rangle$
- **Constraint**:
  $\langle Agent, Location, Time \rangle$
  Option 1: a_1 cannot stay in x at time step t.
  Option 2: a_2 cannot stay in x at time step t.

Conflict-Based Min-Cost Flow (CBM) for TAPF



Our algorithm:
Conflict-Based Min-Cost Flow (CBM) considers each **team** to be a meta-agent.
A best-search on a search tree, nodes stored in a priority queue.
The key of each tree node is the makespan of the paths stored in the node.
For every tree node:

1. Find *paths* for a single **team** separately.
2. Look for **collisions** in *paths*.
3. If there is a **collision** between $team_1$ and $team_2$:
   Option 1 or Option 2 to avoid collision.

Conflict-Based Min-Cost Flow (CBM) for TAPF

- **Collision**: $\langle \textit{Team } \textit{team}_1, \textit{Team } \textit{team}_2, \textit{Location } x, \textit{Time } t \rangle$
- **Constraint**:
  $\langle \textit{Team}, \textit{Location}, \textit{Time} \rangle$
  Option 1: agents in $\textit{team}_1$ cannot stay in x at time step t.
  Option 2: agents in $\textit{team}_2$ cannot stay in x at time step t.

Finding Paths for Single Teams

"Find paths for each single **team** $team_i$ separately" **=**

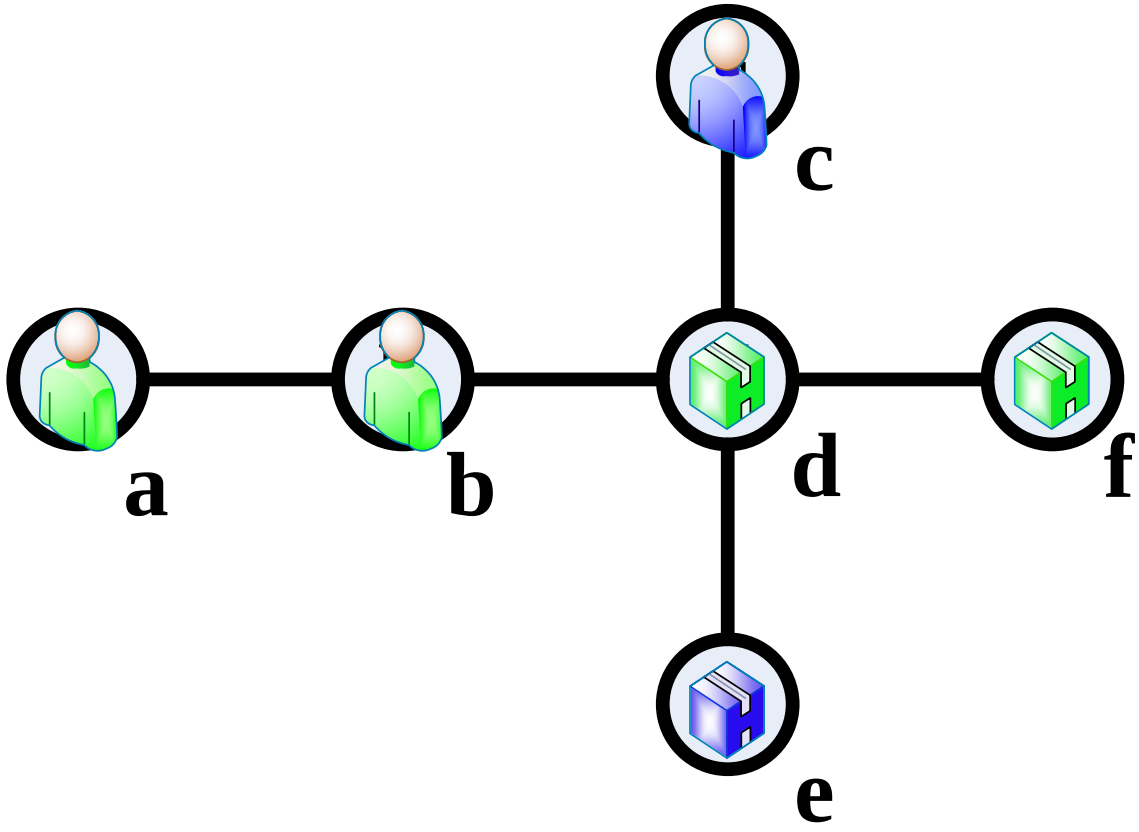
1. Assign agents in $team_i$ to targets given to $team_i$

AND

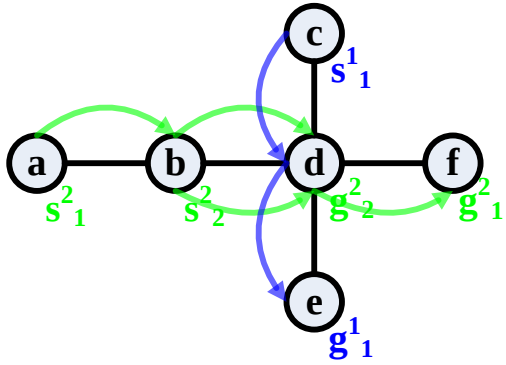
2. Find paths for $team_i$ that have no collisions among agents in $team_i$, according to the target assignment.

Use a polynomial-time min-cost max-flow algorithm on a time-expanded network.

An Example

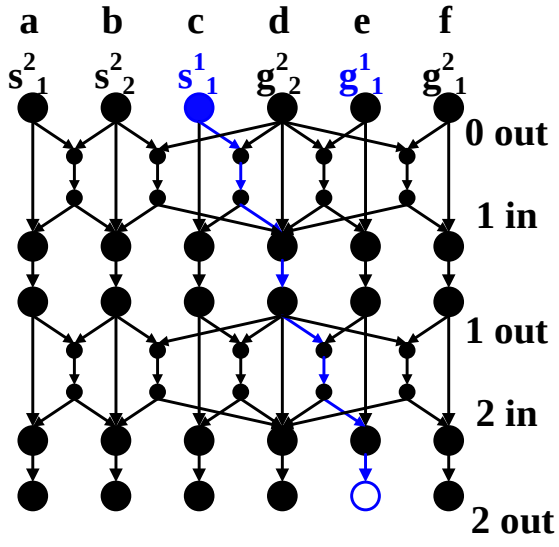


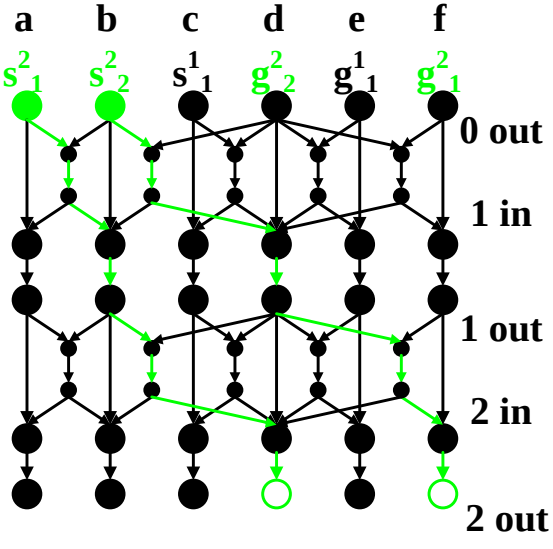
Finding Paths for Single Teams Separately



team1
{c,d,e}

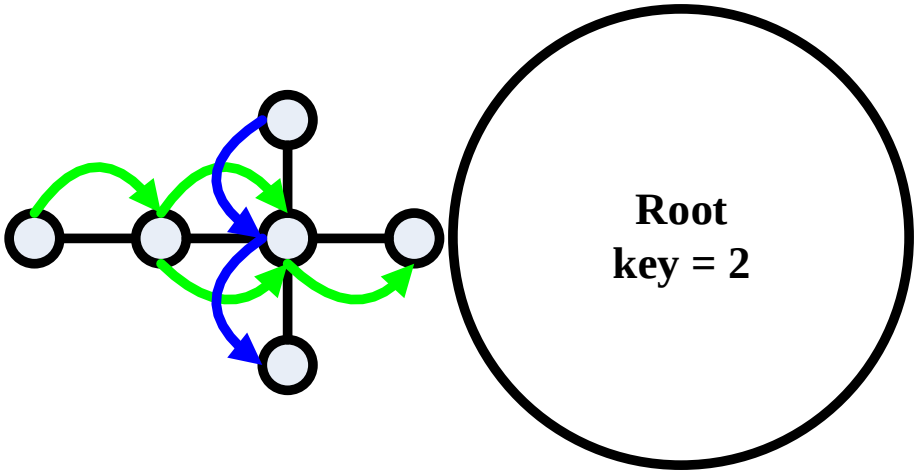
team2
{a,b,d}
{b,d,f}



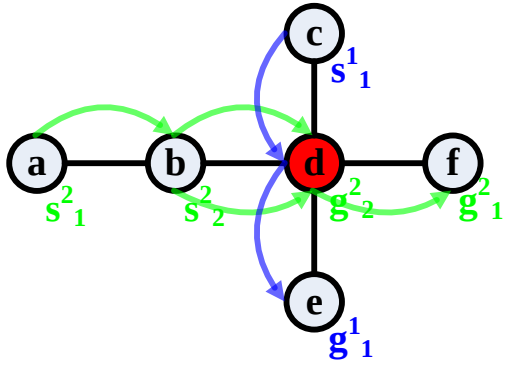


team1 **team2**

Storing Paths and Key

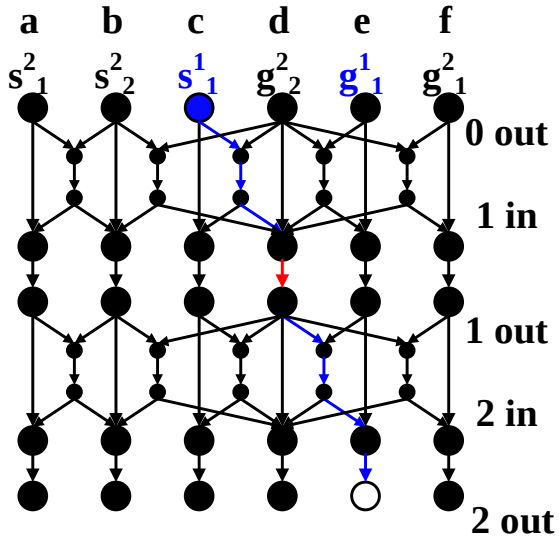


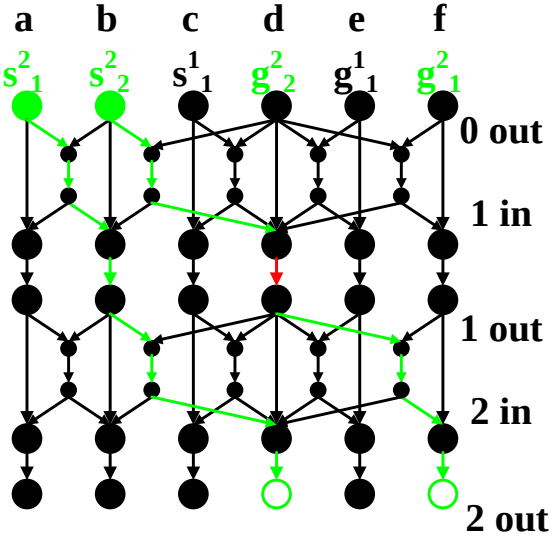
Looking for Collisions in Paths



team1
{c,d,e}

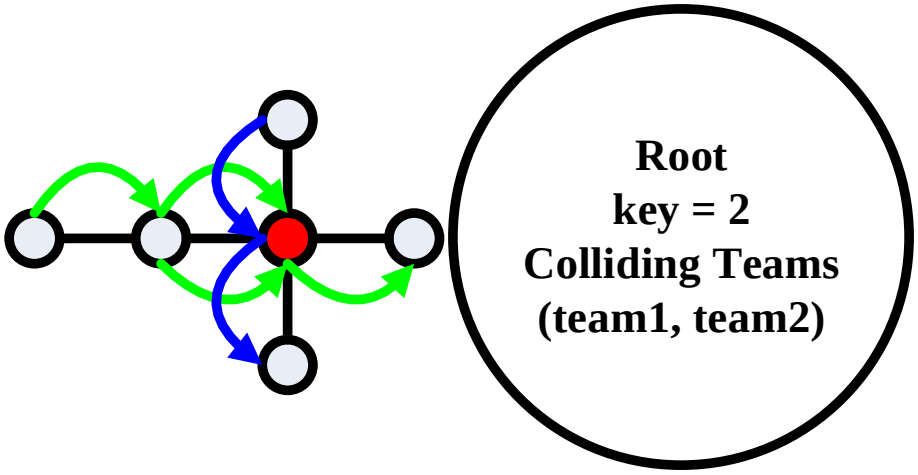
team2
{a,b,d}
{b,d,f}



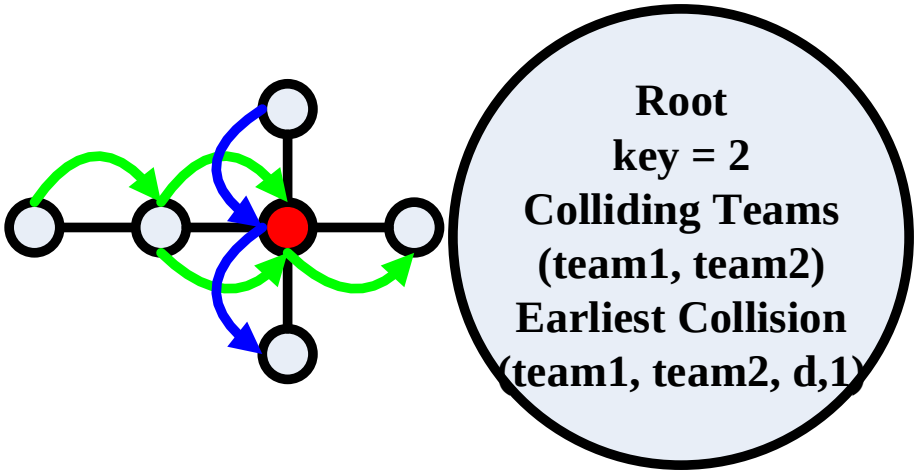


team1 **team2**

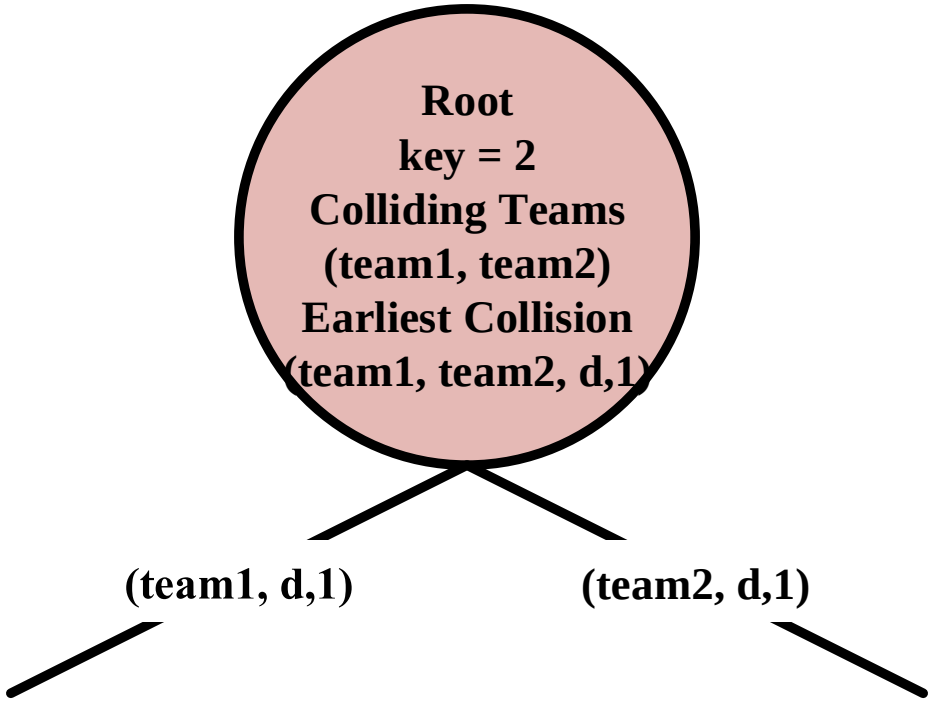
Storing Colliding Teams



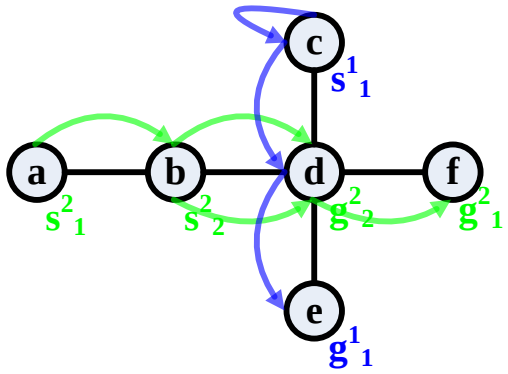
Poping a Tree Node



Two Options



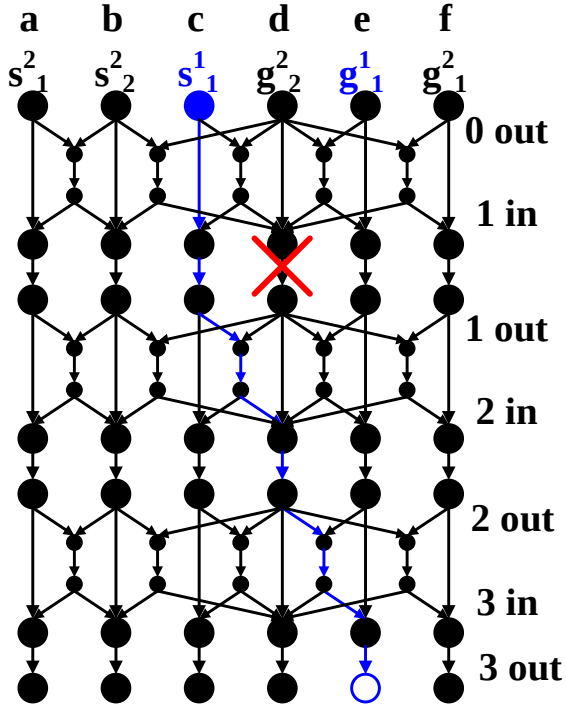
Option 1: Find New Paths for $team_1$



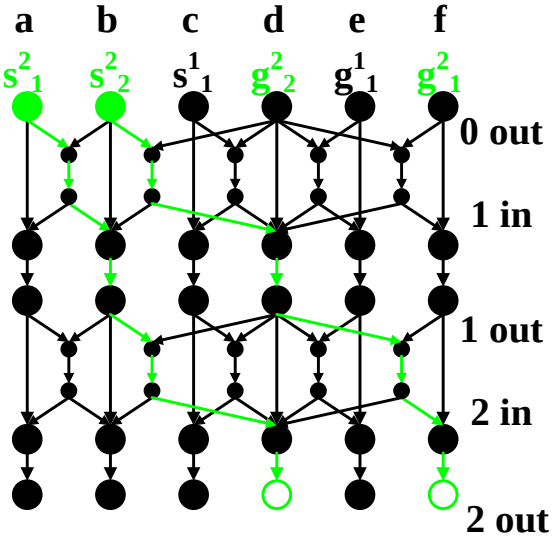
team1
{c,c,d,e}

Constraints
(team1,d,1)

team2
{a,b,d}
{b,d,f}

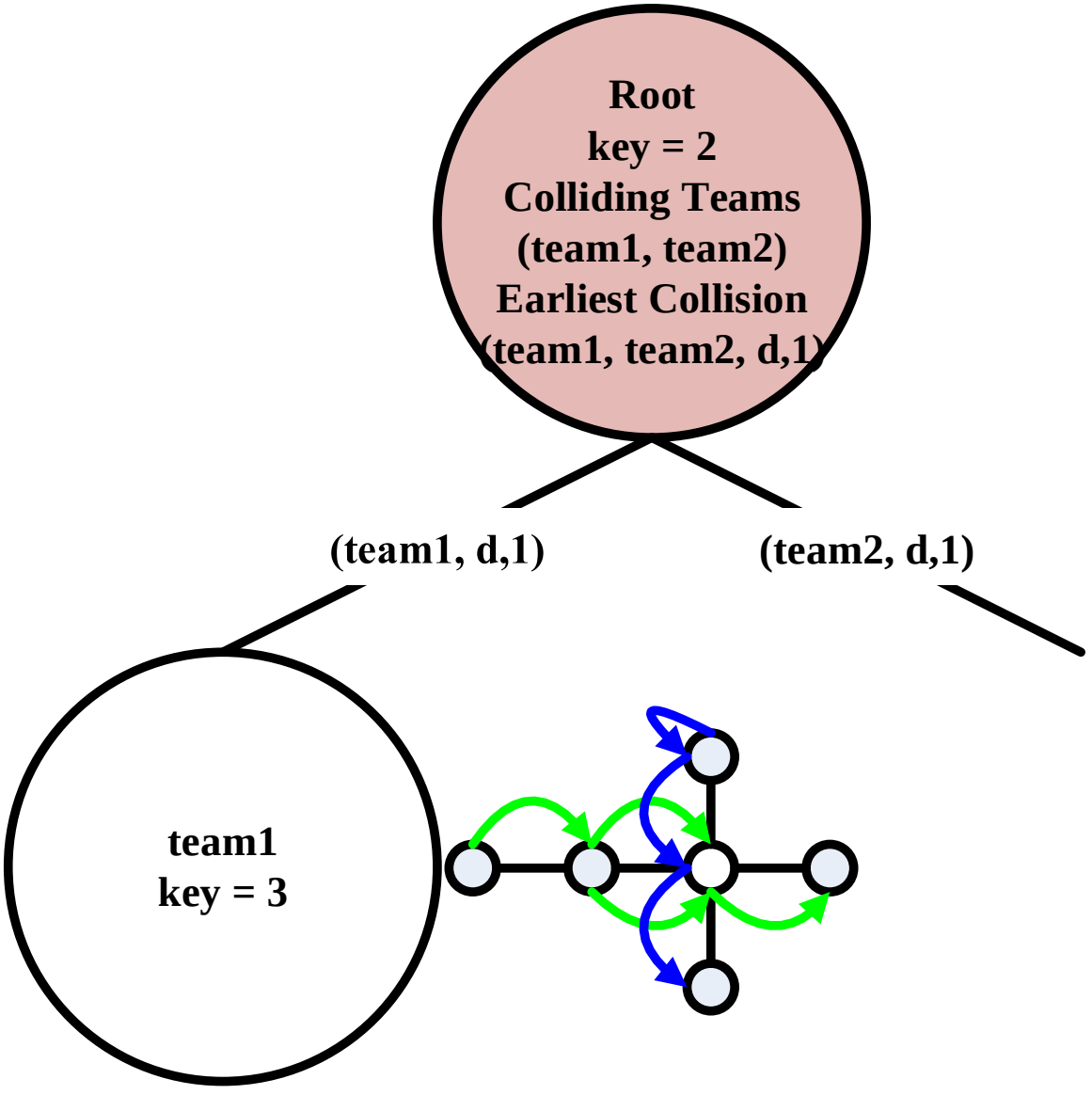


team1

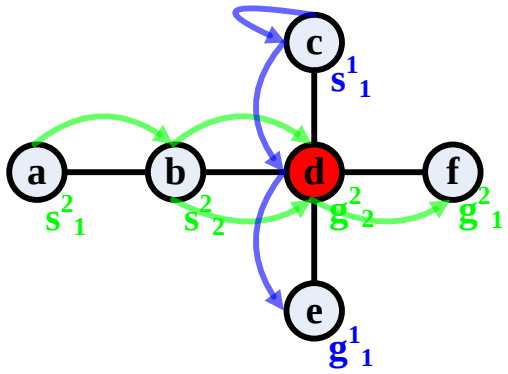


team2

Storing Paths and Key



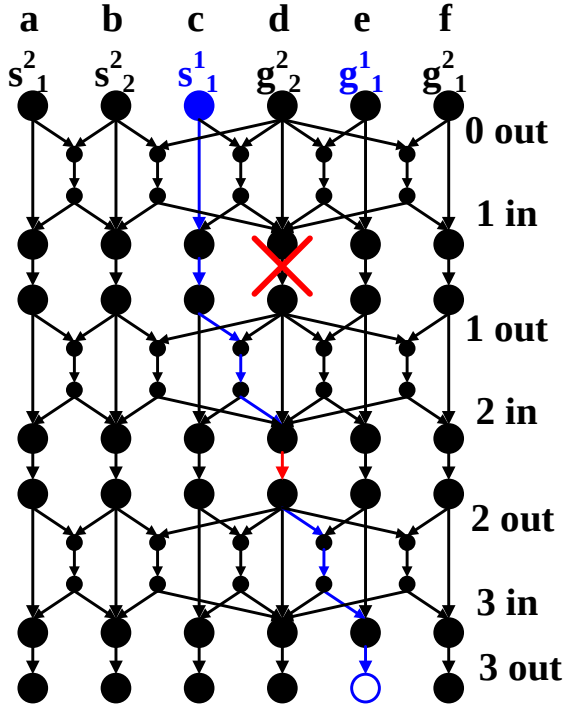
Looking for Collisions in Paths



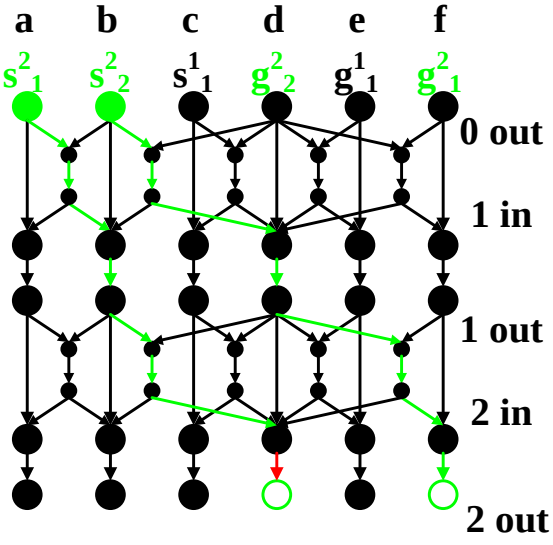
team1
{c,c,d,e}

team2
{a,b,d}
{b,d,f}

Constraints
(team1,d,1)

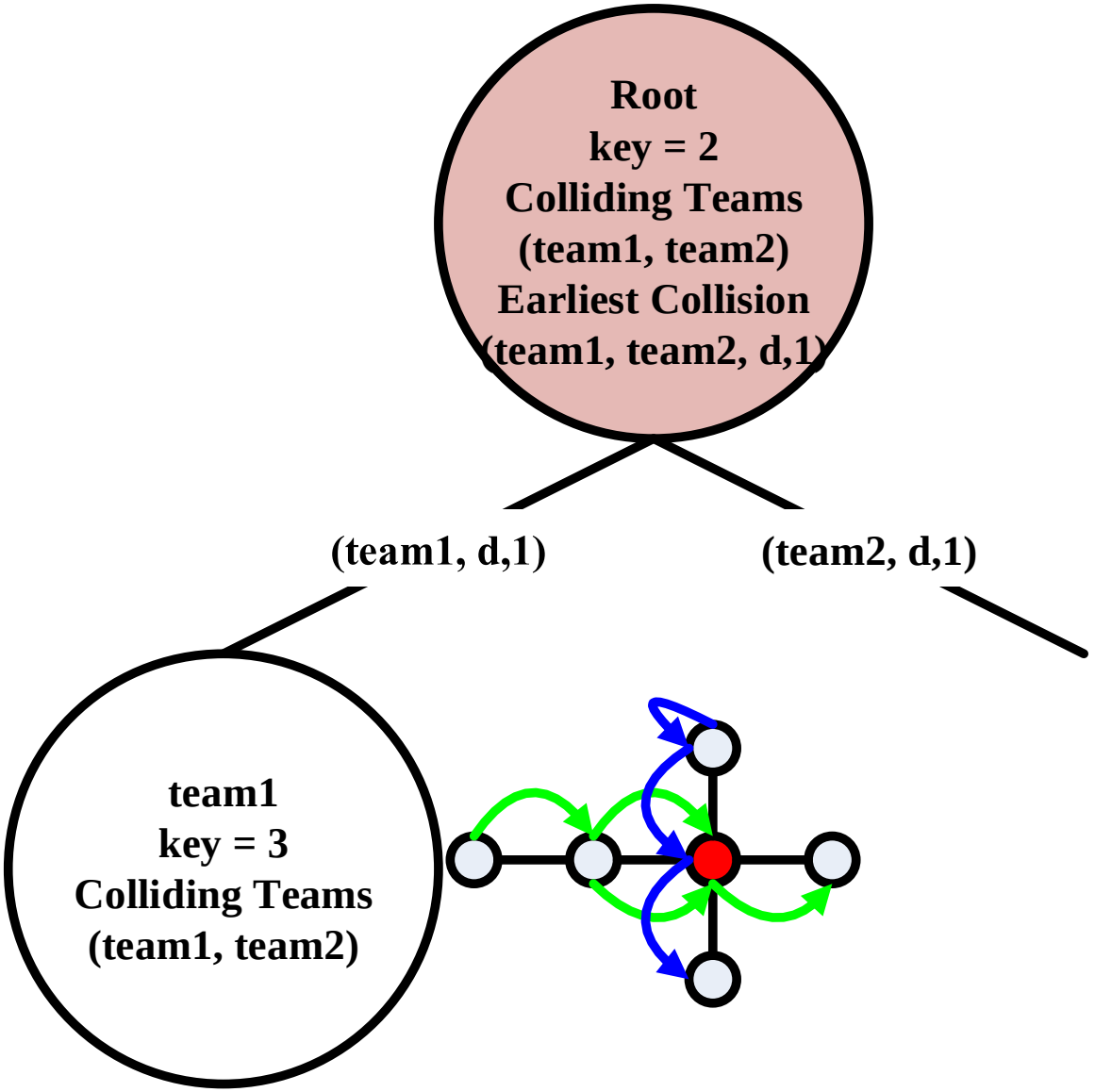


team1

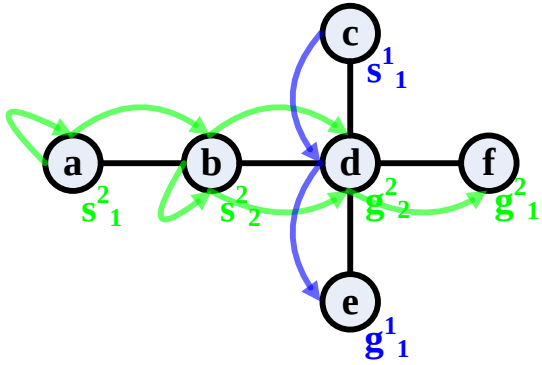


team2

Storing Colliding Teams

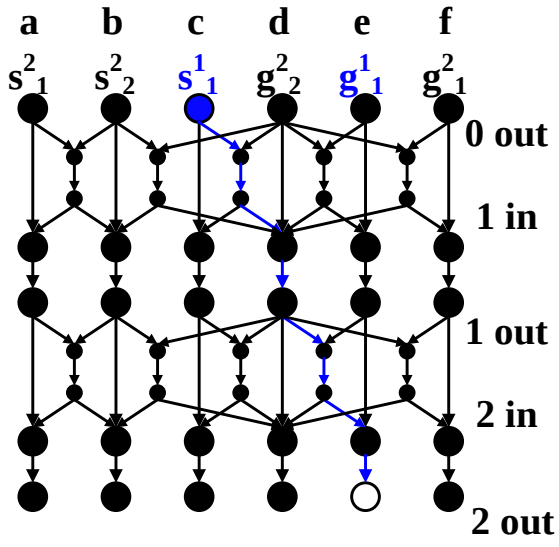


Option 2: Find New Paths for *team*$_2$

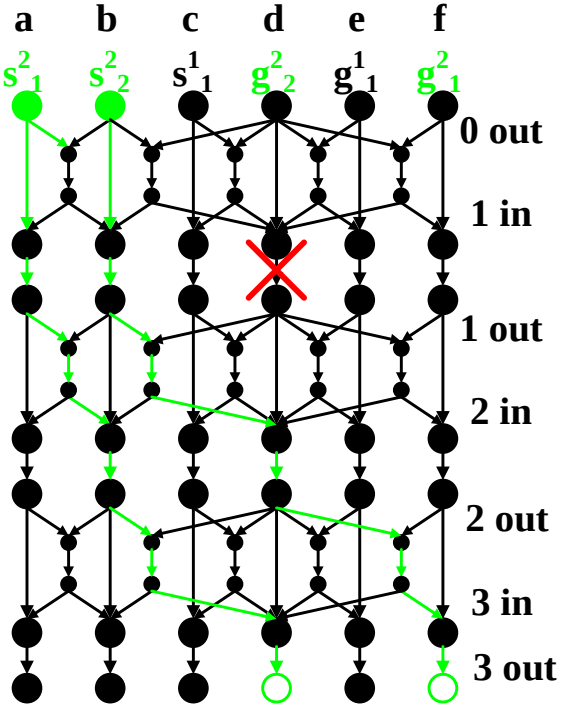


team1 **Constraints**
{c,d,e} **(team2,d,1)**

team2
{a,a,b,d}
{b,b,d,f}

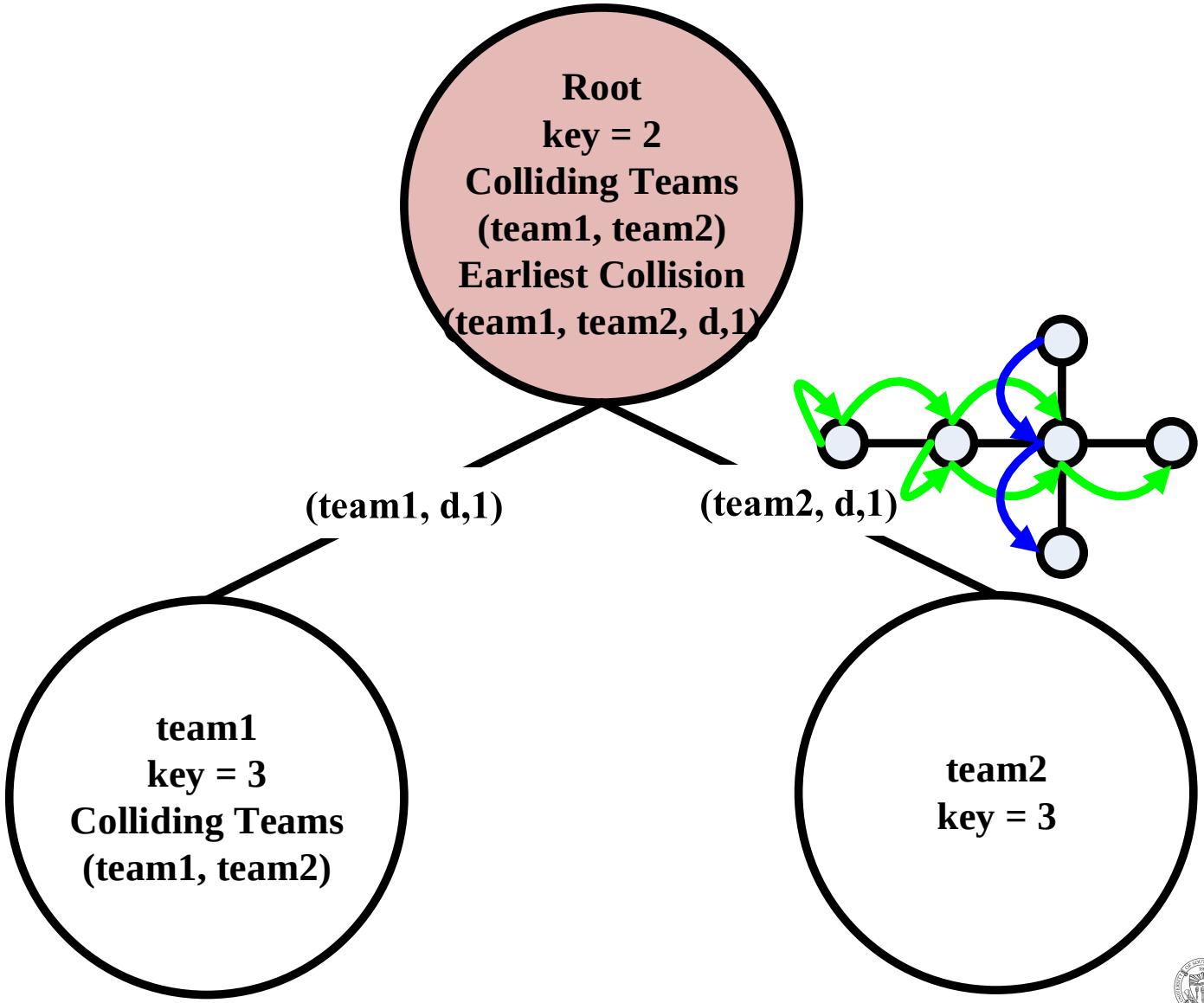


team1

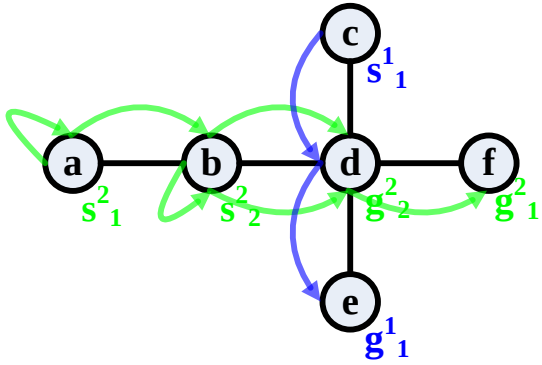


team2

Storing Paths and Key

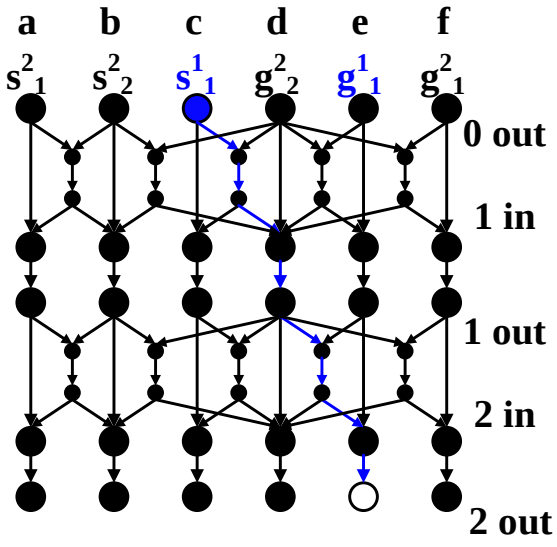


Looking for Collisions in Paths

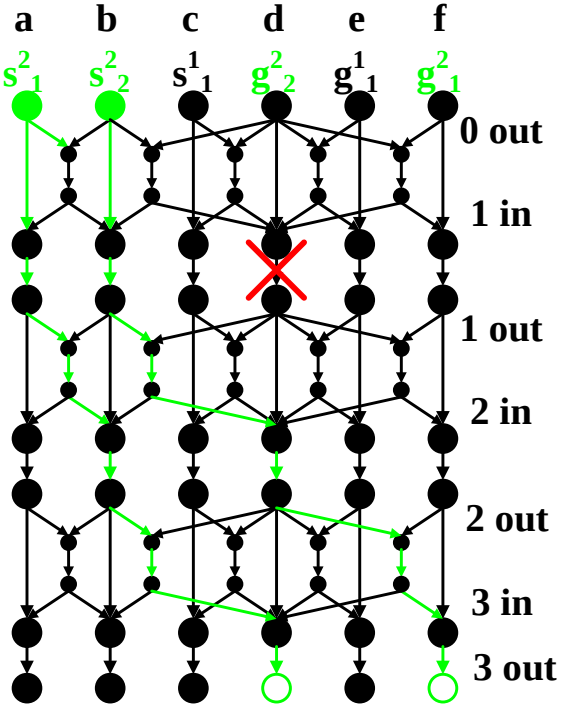


team1 **Constraints**
{c,d,e} **(team2,d,1)**

team2
{a,a,b,d}
{b,b,d,f}

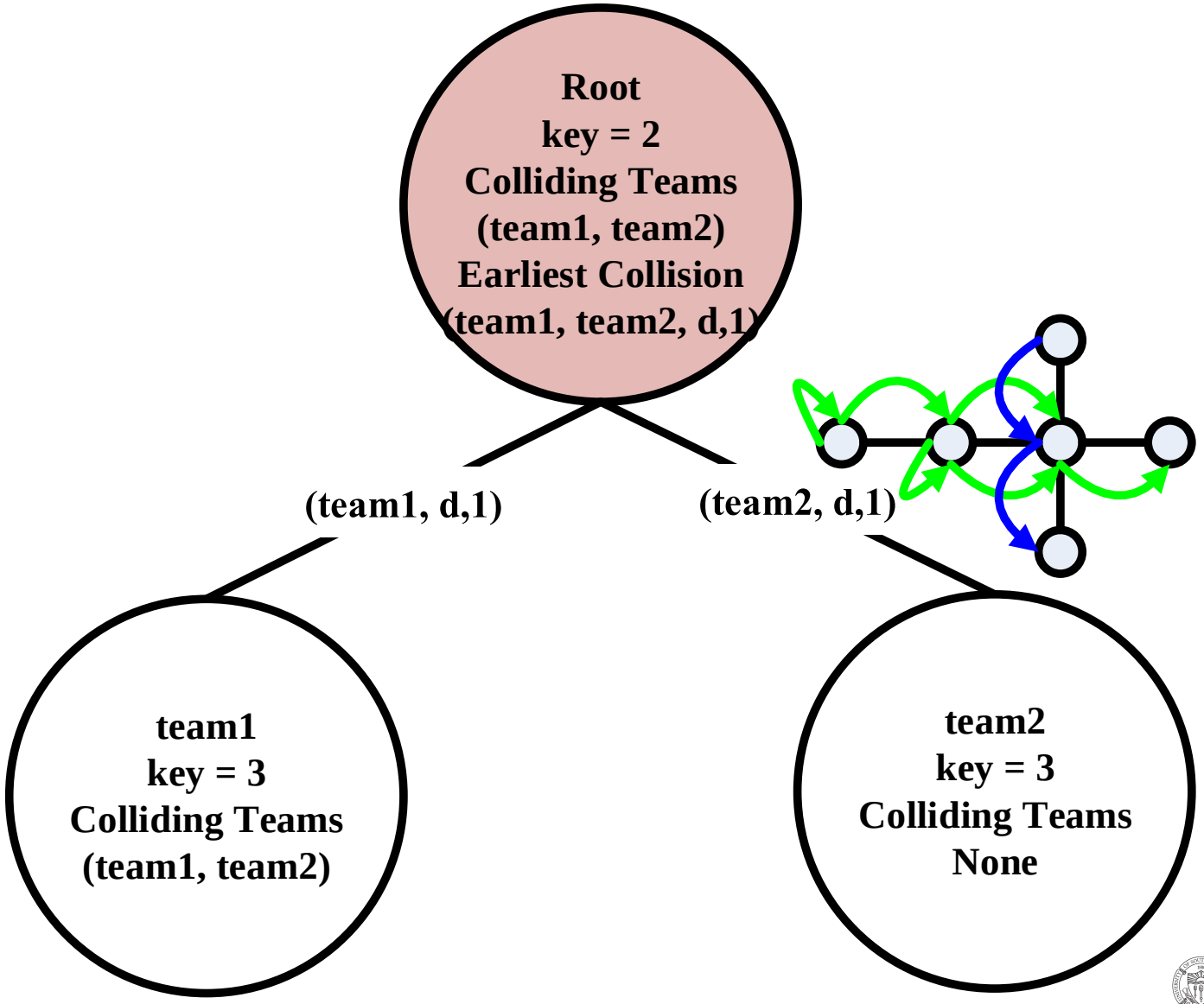


team1



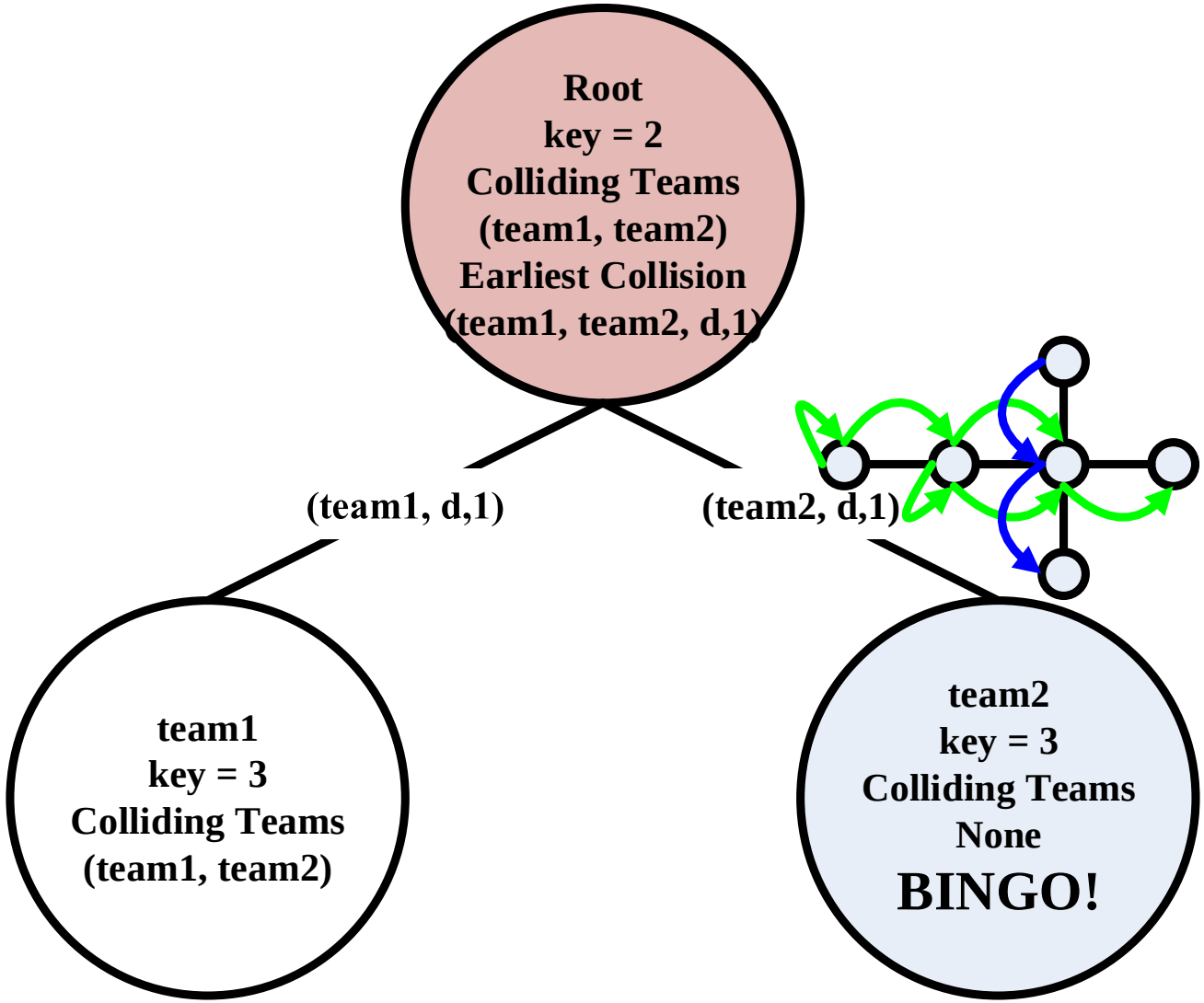
team2

Storing Colliding Teams



Poping a Tree Node

Ties are broken to favor nodes with fewest colliding pairs.



Edge Weights — Reducing Possible Collisions

Idea: Choose paths that have fewest collisions with other teams, when finding paths for a single team.

- Take into account the paths of other teams.
- Bias the search using a **min-cost max-flow algorithm** that finds a max flow with minimal total edge weights.

Edge Weights are Crucial

Setups:

- 30×30 4-neighbor grids with 10% randomly blocked cells.
- 5 agents per team.
- 5-minute time limits.

	CBM		Unweighted CBM	
agents	time	success	time	success
10	0.34	1	0.41	0.72
15	0.57	1	1.06	0.44
20	0.78	1	2.06	0.22
25	1.07	1	1.58	0.08
30	1.71	1	6.73	0.02
35	1.92	1	-	0
40	2.95	1	-	0
45	3.66	1	-	0
50	5.32	1	-	0

Guarantees

CBM is optimal and complete.

Comparisons

Setups:

- 30×30 4-neighbor grids with 10% randomly blocked cells.
- 5-minute time limits.

CBM is Faster

CBM: Specialized solver.

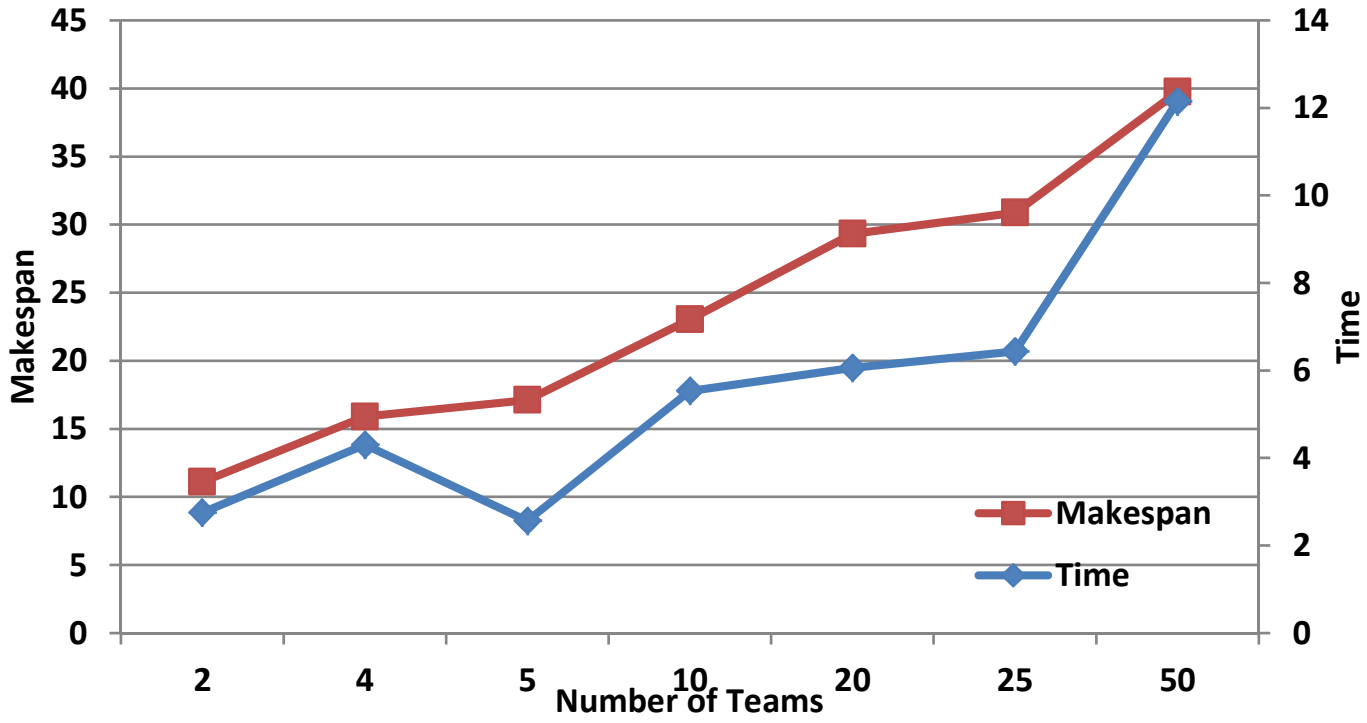
versus

ILP (Integer Linear Program): Useful tool and easy to model.

	CBM		ILP	
agents	time	success	time (over solved instances)	success
10	0.34	1	18.24	1
15	0.57	1	35.44	1
20	0.78	1	62.85	0.94
25	1.07	1	88.55	0.82
30	1.71	1	108.75	0.66
35	1.92	1	121.99	0.46
40	2.95	1	152.98	0.14
45	3.66	1	161.52	0.14
50	5.32	1	161.95	0.04

Spectrum: Anonymous$\longleftrightarrow$Non-Anonymous

Fixed 100 agents in total, 2 to 50 teams.



[2 teams, 50 agents per team] $\longleftrightarrow$ [50 teams, 2 agents per team]

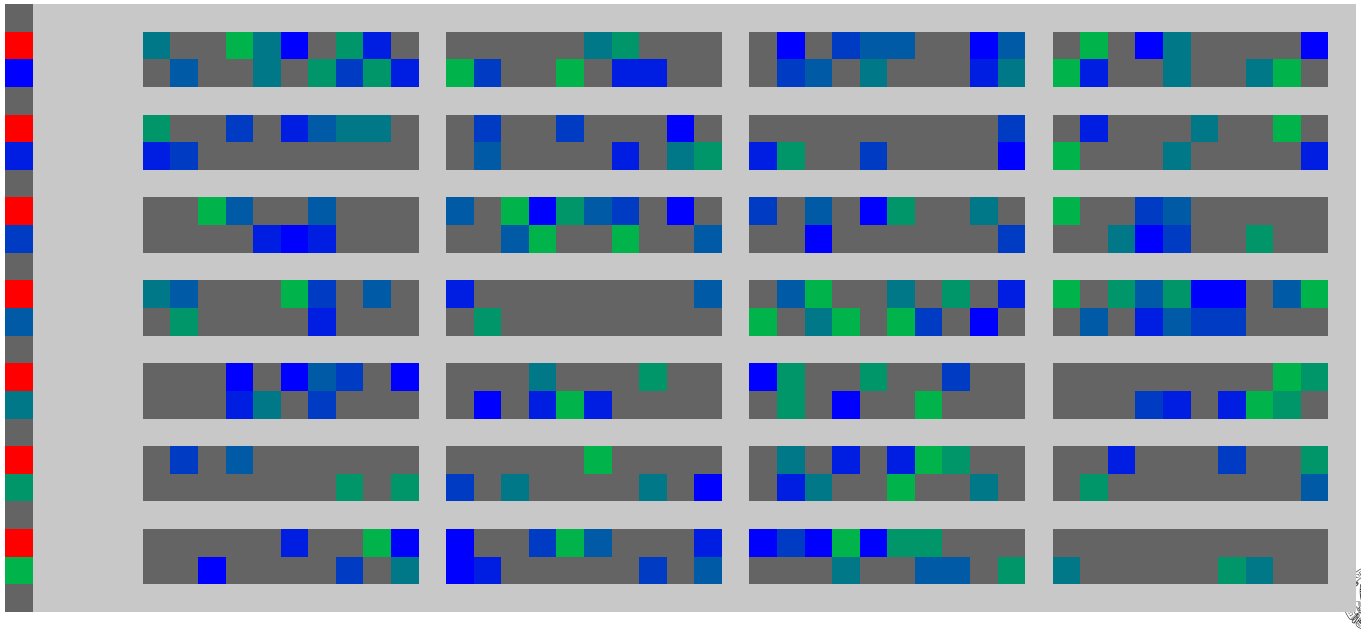
Anonymous $\longleftrightarrow$ Non-Anonymous

P $\longleftrightarrow$ **NP**-hard

USC University of Southern California

Scalibility: Simulated Warehouse System

- Each instance has 420 agents: 210 “incoming” and 210 “outgoing”.
- CBM solves 40 out of the 50 Kiva instances within a time limit of 5 minutes each.
- Average running time over solved instances is 91.61 seconds.



Takeaways

TAPF: A mix of non-anonymous MAPF and anonymous MAPF.
CBM: Guarantees optimality and completeness.