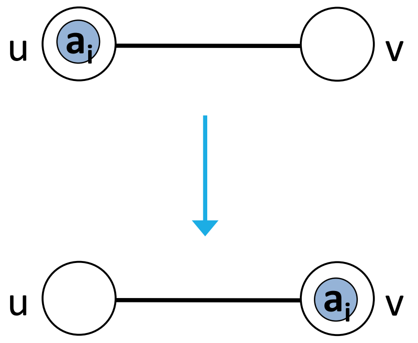


SAT-based Multi-Agent Path Finding

an overview

PAVEL SURYNEK

ČVUT - CZECH TECHNICAL UNIVERSITY

FIT - FACULTY OF INFORMATION TECHNOLOGY

PRAHA, CZECHIA

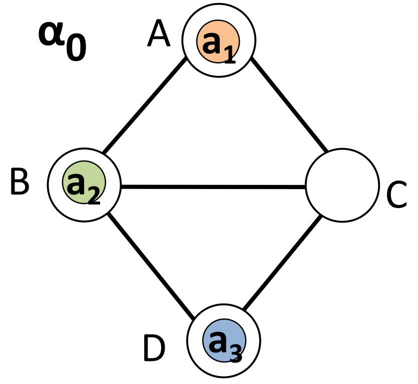
Background

in Multi Agent Path Finding

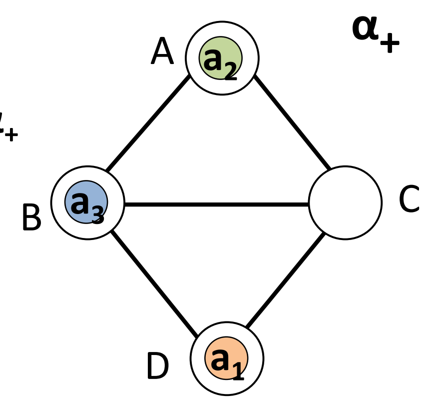
Multi-agent Path Finding (**MAPF**) [Silver, 2005]

Problem components

- **G=(V,E)**
  - agents placed in vertices
    - $A = \{a_1, a_2, \ldots, a_k\}$, **$k < |V|$**
- **at least one** vertex empty
  - **at most one** agent per vertex

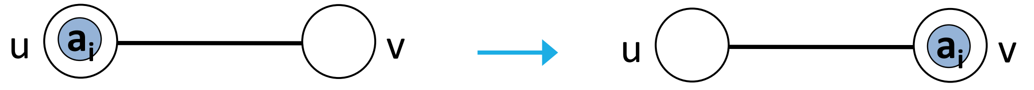


	α_0	α_1	α_2	α_3	$\alpha_4 = \alpha_+$
a_1	A	C	C	C	D
a_2	B	B	A	A	A
a_3	D	D	D	B	B



Task

- initial placement of agents
  - $\alpha_0: A \rightarrow V$
- **<u>move agents</u>** so that agents arrive to their goals
  - goal agent placement $\alpha_+: A \rightarrow V$



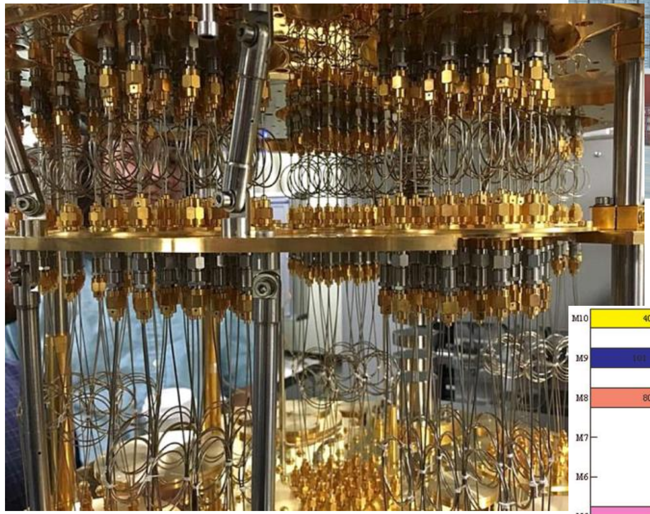
Moving agent a_i across edge $\{u,v\}$ into empty v

Motivation

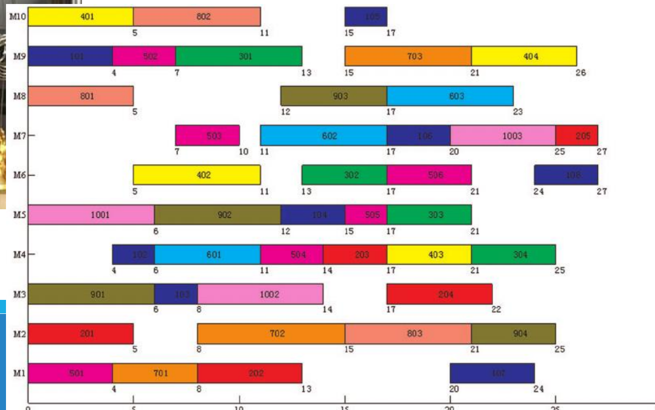
Navigation of multiple robots
- agent = robot

Container **movement planning**
- agent = container

Quantum **program compilation**
- qubit/quantum gate allocation

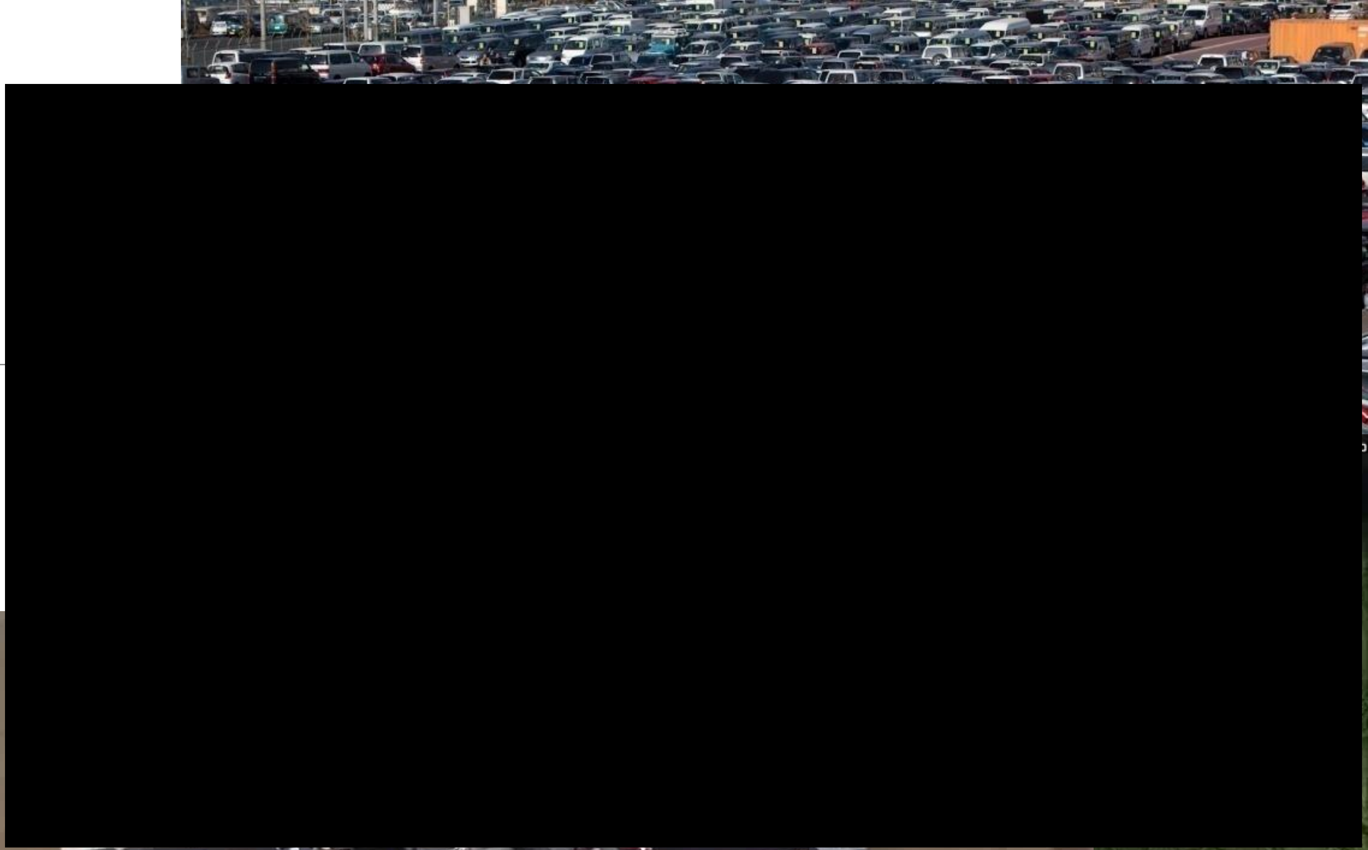


Robust **scheduling/planning**
- repair the schedule/plan by swapping of activities



Economic Impact

- **KIVA agents/Amazon**
  - warehouse relocation
  - bought by **Amazon**
    - $ 775.000.000
- **Autonomous cars**
  - **Google**, **Toyota**, **Tesla**
  - combines
    - autonomy
    - multi-agent path finding
- **Parking systems**
  - **AVERT**
  - saves
    - space, time,
      - energy, ...
- **Computer games**
  - $ multi-billion market



Optimization – Makespan/Sum-of-costs

When **time** matters (makespan **μ**)

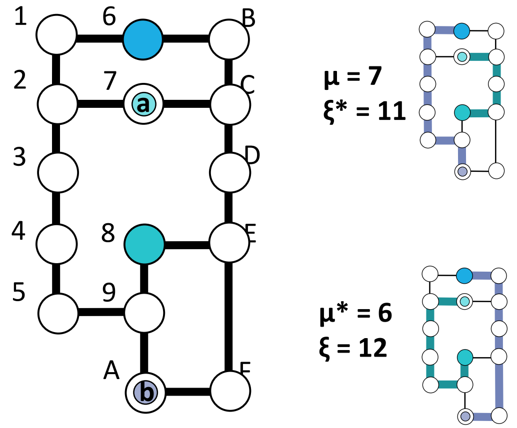
- each more requires 1 unit of time
- all agents in goals at earliest time

When **energy** matters (sum-of-costs **ξ**)

- each move consumes 1 unit of energy
- the least energy consumed in total

Makespan and sum-of-costs **optimization**

- go against each other
- both NP-hard



Optimal **makespan μ***
Sub-optimal sum-of-costs **ξ**

	α_0	α_1	α_2	α_3	α_4	α_5	$\alpha_6 = \alpha_+$
a	A	F	E	D	C	B	**6**
b	7	2	3	4	5	9	**8**

Optimal **sum-of-costs ξ***
Sub-optimal makespan **μ**

	α_0	α_1	α_2	α_3	α_4	α_5	α_6	$\alpha_7 = \alpha_+$
a	A	9	5	4	3	2	1	**6**
b	7	C	D	E	**8**	**8**	**8**	**8**

Optimization - Complexity

Minimize cumulative costs such a the number of moves, cost, fuel, …

- **unit edges** in the basic variant
  - each move or wait action costs 1
- NP-hard problem [Ratner & Warmuth, 1986; Bonnet et al., 2016]
- inapproximable (APX-hard) [Mitzow et al., 2016]

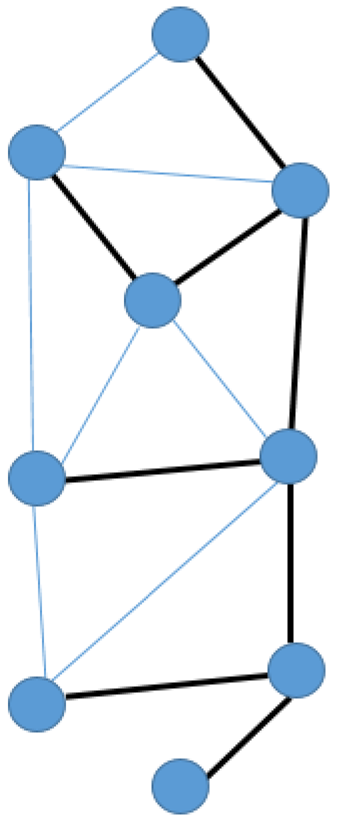
Known bounds [Kornhauser et al., 1984; Yamanaka et al., 2016]

- any MAPF instance can be solved using $O(|V|^3)$ moves
- there are instance that need $\Omega(|V|^3)$

Feasible solution (not requiring the minimum number of moves)

- can be found in polynomial time
- $O(|V|^3)$ time and $O(|V|^3)$ moves

G = (V,E)



Solving MAPF

reduction to SAT

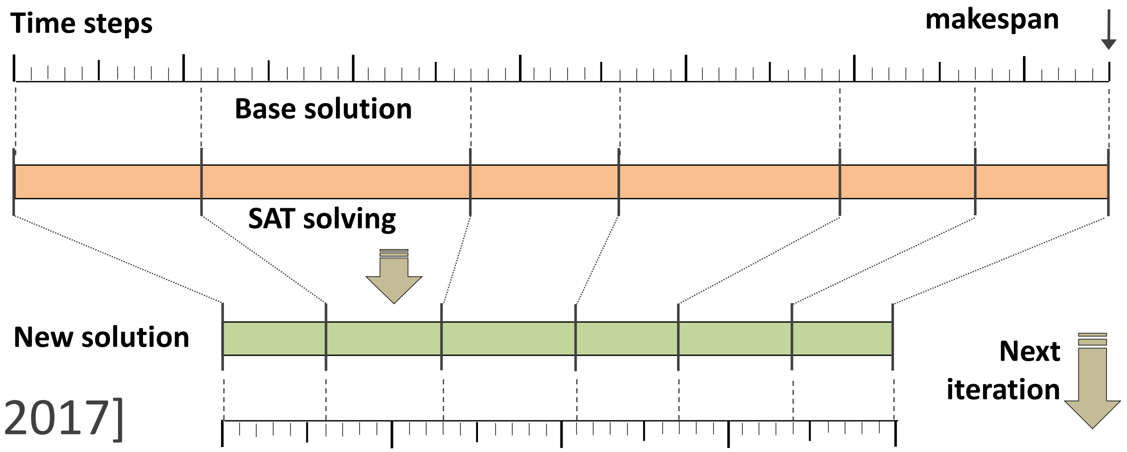
Overview of SAT-based Approaches

Improving sub-optimal solutions [2011]

- takes a solution generated by some polynomial time algorithm and improves it w.r.t. given cumulative objective (makespan, sum-of-costs, fuel, …)
  - replaces sub-sequence of moves in the current solution with an **optimal sub-sequence**

SAT-Plan inspired approach [2014]

- being confident and going directly to optimal solution
  - like replacing entire solution sequence with an optimal one
- we do not need the iterative process at all



Problem Decomposition / Independence Detection [2017]

- planning for isolated groups of agents separately

Lazy Compilation + SMT [2019]

- using incomplete propositional encodings – do not encode all MAPF rules

Propositional Satisfiability (SAT)

- **Propositional formula / satisfiability**
  - a formula $\mathcal{F}$ over 0/1 (false/true) variables
  - Is there an assignment under which $\mathcal{F}$ evaluates to 1/true?

$(x \lor \neg y) \land (\neg x \lor y)$
Satisfied for $x = 1$, $y = 1$

- **Benefits of reduction**
  - **powerful** propositional solvers
    - decades of development
    - MiniSAT, clasp, glucose, glue-MiniSAT, crypto-MiniSAT, …
    - intelligent search, learning, restarts, heuristics, …
- and most recently
  - machine learning for variable/value ordering
    - MapleSAT (winner in recent SAT competitions)

  - multi-agent path finding $\Rightarrow$ **formula** $\mathcal{F}$
    - all these advanced techniques **accessed almost for free**

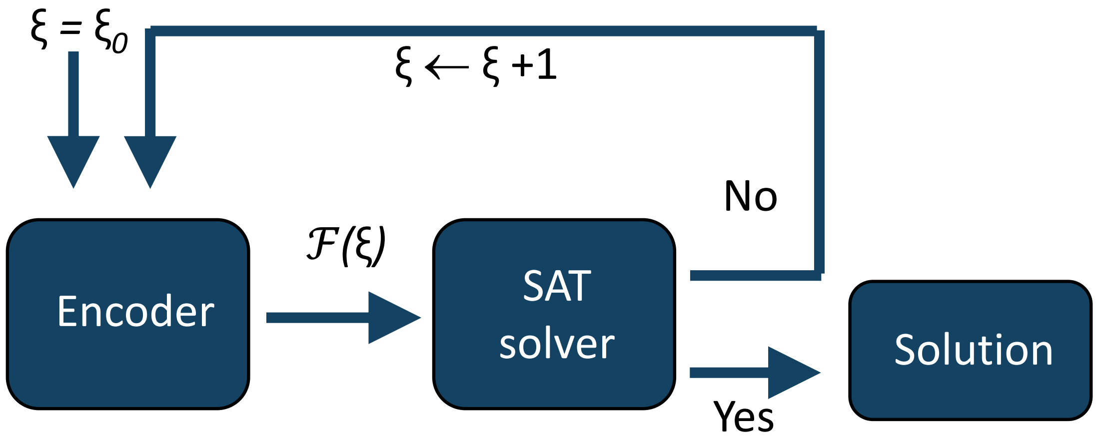
Reducing **MAPF to SAT** [Surynek, 2012]

MAPF instance → sequence of SAT instances

- $\mathcal{F}(\xi)$ satisfiable iff MAPF has a solution of cost ξ
- consult the SAT solver on $\mathcal{F}(\xi_0)$, $\mathcal{F}(\xi_0+1)$, $\mathcal{F}(\xi_0+2)$, … until a satisfiable formula is met
  - ξ_0 lower bound on the cost
- cumulative objectives in MAPF imply **monotonicity** of solvability
  - unsolvable, unsolvable, unsolvable, solvable, solvable, …

Iterative Algorithm – **MDD-SAT**

  - ξ_0 sum of lengths of shortest paths
- first satisfiable $\mathcal{F}(\xi)$ corresponds to the minimum cost
  - satisfiability of $\mathcal{F}(\xi)$ is monotonic w.r.t. ξ

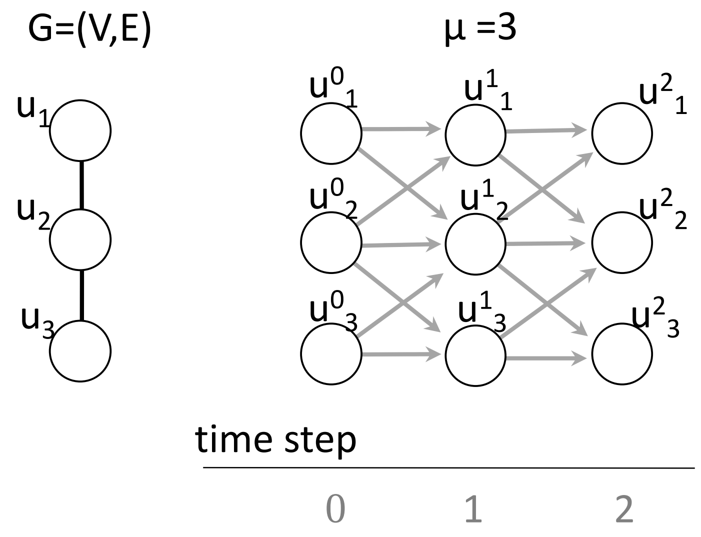


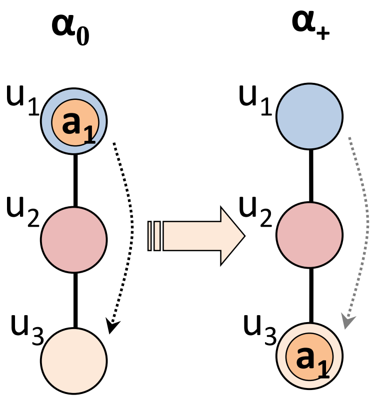
[Surynek, Felner, Stern, Boyarski, 2016]

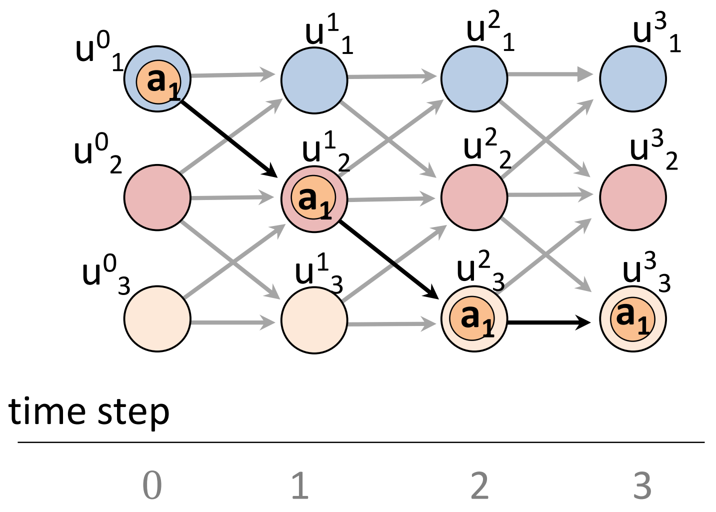
MAPF **Encoding** through Time Expansion

Time Expanded Graph (TEG)

TEG for agent a_1 = TEG_1







- positions of **all agents at all time-steps are represented** in TEGs
- introduce a **propositional** variable for **each node** and **each edge** in TEGs
  - a node variable is TRUE iff an agent occupies the vertex at the given time-step
  - an edge variable is TRUE iff an agent makes move across the edge

Example of MAPF Rule Encoding

- Propositional variables for $\mathbf{a_i \in A, v,u \in V, t} \in$ **{0,1,…, bound derived from the objective}**
  - **Node variables**
    - $\mathbf{X(a_i)_u^t}$ TRUE iff agent $\mathbf{a_i}$ occupies vertex **v** at time-step **t**
  - **Edge variables**
    - $\mathbf{E(a_i)_{u,v}^t}$ TRUE iff agent $\mathbf{a_i}$ starts traversal of edge **(u,v)** (starting in u) at time-step **t**
- Target vertex **v** of a movement of agent $\mathbf{a_i}$ across **(u,v)** must be empty at time-step **t**

$$E(a_i)_{u,v}^t \Rightarrow \bigwedge_{a_j \in A \mid a_j \neq a_i} \neg X(a_j)_v^t$$

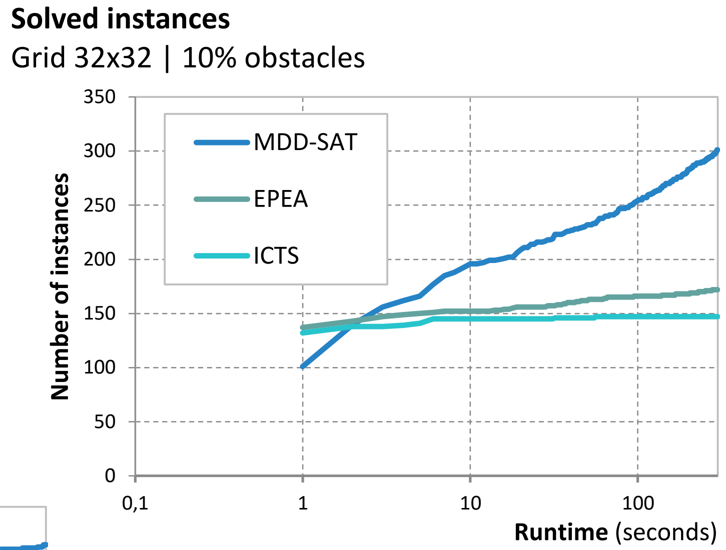
- Implication a ⇒ (¬b ∧ ¬c ∧ ¬d…) can be written as a conjunction of multiple binary clauses
  - (¬a ∨ ¬b) ∧ (¬a ∨ ¬c) ∧ (¬a ∨ ¬d) ∧ …

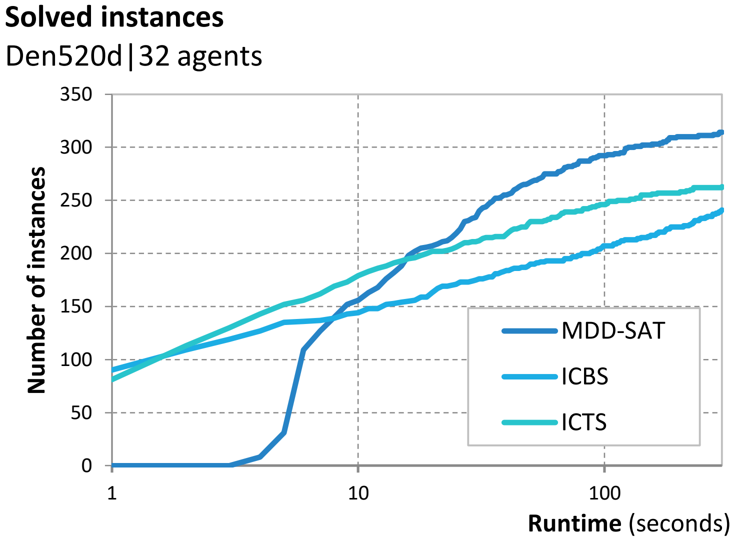
Experimental Evaluation

- **Setup**
  - small 4-connected **grids**
    - **random** initial and
      - goal arrangement
    - **dense** occupation
  - **large** game maps
    - **Dragon Age** – standard benchmark
    - **sparse** occupation

- **Comparison**
  - search-based algorithms
    - previous **state-of-the-art**
    - **ICTS** [Felner et al., 2013], **EPEA** [Sharon et al., 2014], **ICBS** [Sharon et al., 2014]

- **Results**
  - SAT-based approach
    - better in **hard setups**





Problem Decomposition [Standley, AAAI 2010]

Solve independent sub-problems **separately**

- Solving procedure of time complexity **$O(2^N)$**
  - N - the number of agents

Problem **decomposition**

- decompose into two independent sub-problems of size N/2
- solve sub-problems separately
- merge solutions of sub-problems

$$time(total) = time(decomposition) + 2 * O(2^{N/2}) + time(merging) = O(N) + O(2^{N/2+1}) << O(2^N)$$

In theory. What about practice?

Independence Detection

Dividing agents in **groups**

- $G_1, G_2, G_3, \ldots$

Plan for each group **independently**

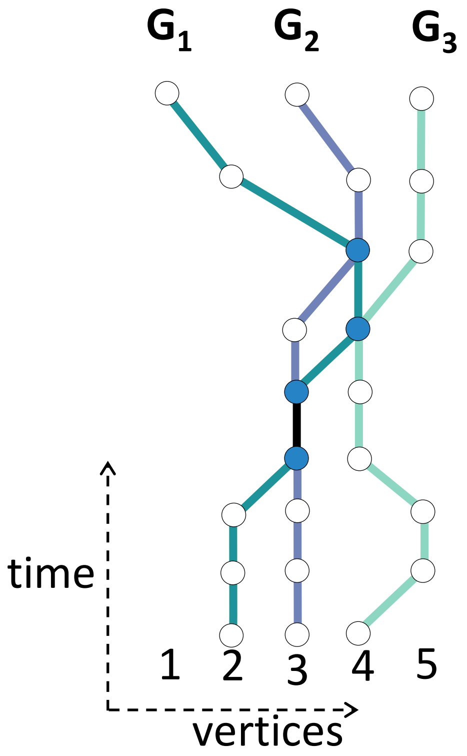
- Time $O(2^{|G_i|})$

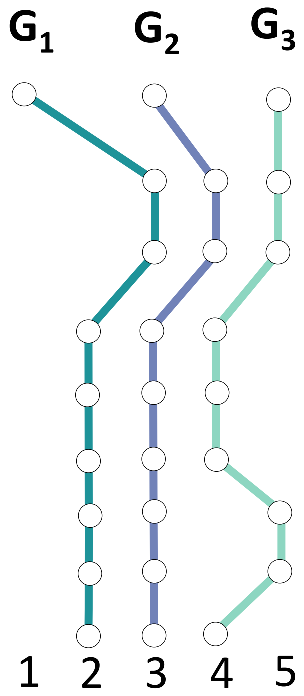
If two groups G_i and G_k **collide**

- Try to replan for G_i
  - while **avoid** all other groups
- or try to replan for G_k
  - while **avoid** all other groups
- if both fails
  - **merge** G_i and G_k

Integration into SAT-based approach

- encode **group avoidance** in formulae





Experiments – small instances

4 – connected grids

- Sizes 8x8, 16x16, 32x32
- 10% obstacles

Agents

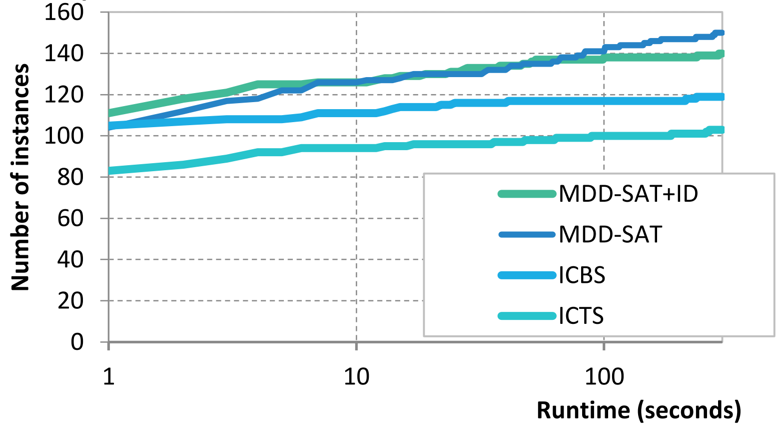
- 1..20 (8x8), 1..40 (16x16), 1..60 (32x32)

Algorithms

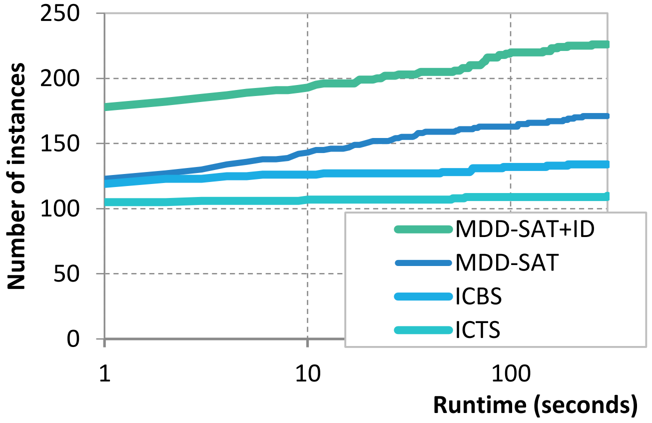
- MDD-SAT
  - original SAT-based MAPF solver (Surynek et al., ECAI 2016)
- MDD-SAT+ID
  - with independent detection
- ICTS, ICBS
  - Increasing cost tree search – search based algorithms (Sharon et al., AIJ 2013)

Solved instances

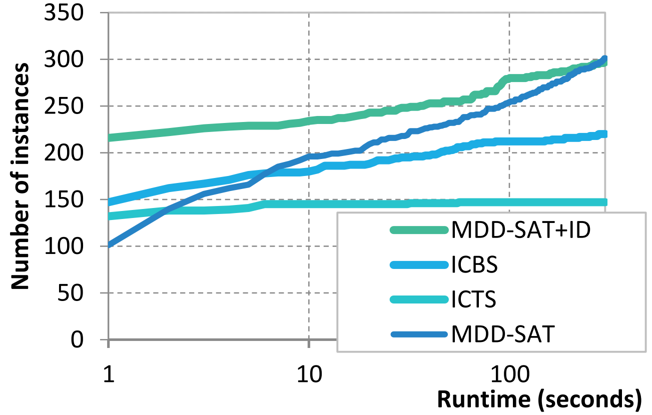
Grid 8x8|10% obstacles



Grid 16x16|10% obstacles



Grid 32x32 | 10% obstacles

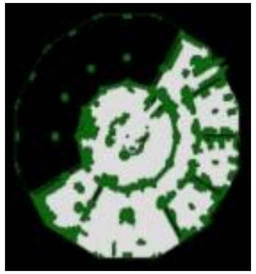


Experiments – large instances

Big 4-connected grids

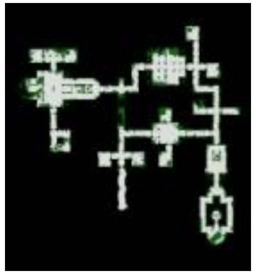
- Dragon Age game
- Size:
  - **481x530** (brc202d), **257x256** (den520d), **194x194** (ost003d)
- **32 agents**

ost003d



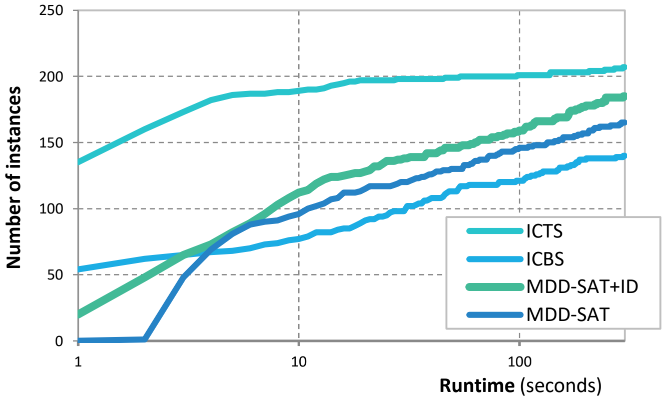
den520d

brc202d

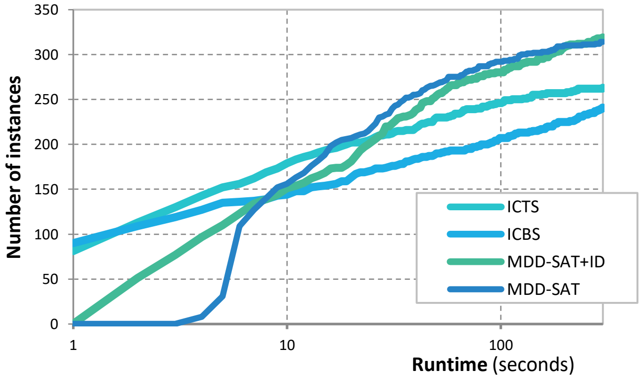


Distance from goals **1..320**

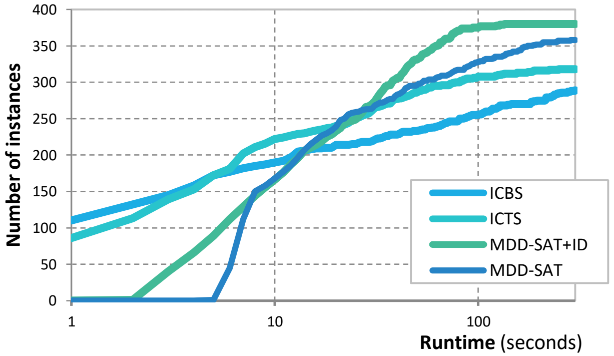
Ost003d|32 agents



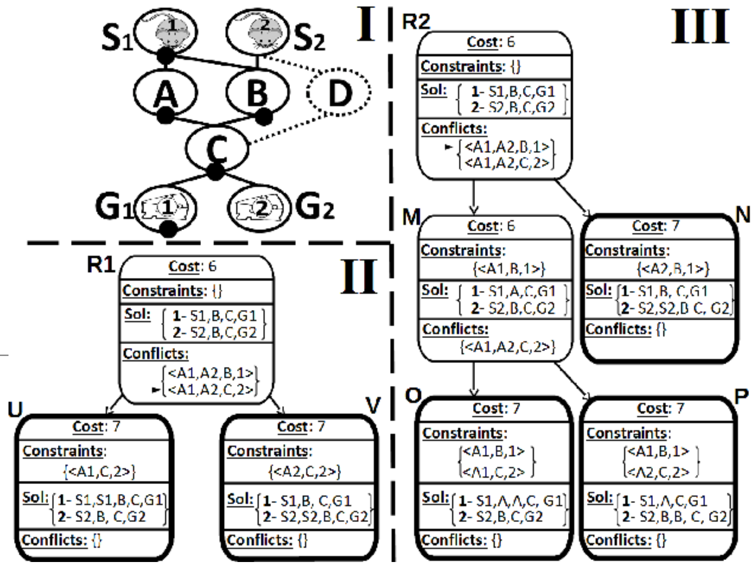
Den520d|32 agents



Brc202d|32 agents



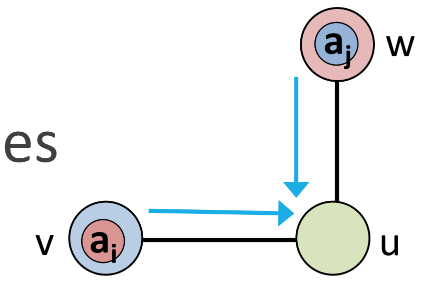
Conflict Based Search

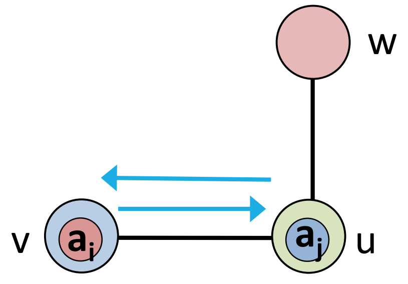


Conflict-based Search (CBS) [Sharon et al., 2013]

- A* at the high level, nodes contain incomplete solutions
- considers collisions lazily

1. searches for individual shortest paths connecting initial position $\alpha_0(a_i)$ with goal $\alpha_+(a_i)$ for each a_i
2. validates solution from the OPEN list w.r.t. MAPF rules
   - a) **vertex conflict** (a_i, a_j, u, t)
     - $\mathbf{a_i}$ and $\mathbf{a_j}$ both appear in **u** at time-step **t**
       - **add conflicts:** $\mathbf{a_i}$ cannot appear in **u** at **t** in one branch, $\mathbf{a_j}$ cannot appear in **u** at **t** in the other branch
   - b) **edge conflict** $(a_i, a_j, \{u,v,w\}, t)$
     - $\mathbf{a_i}$ traverses **(u,v)** at **t** but $\mathbf{a_j}$ appearing in **v** at **t** traverses **(v,u)** (opposite direction) which is forbidden usually
       - **add conflicts:** $\mathbf{a_i}$ cannot traverse **(u,v)** at **t** or $\mathbf{a_j}$ cannot traverse **(v,u)** at **t**

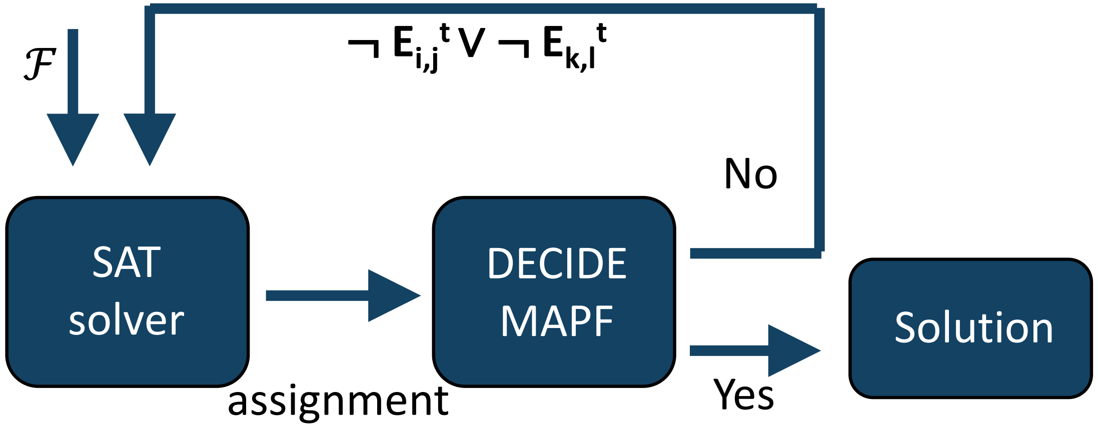


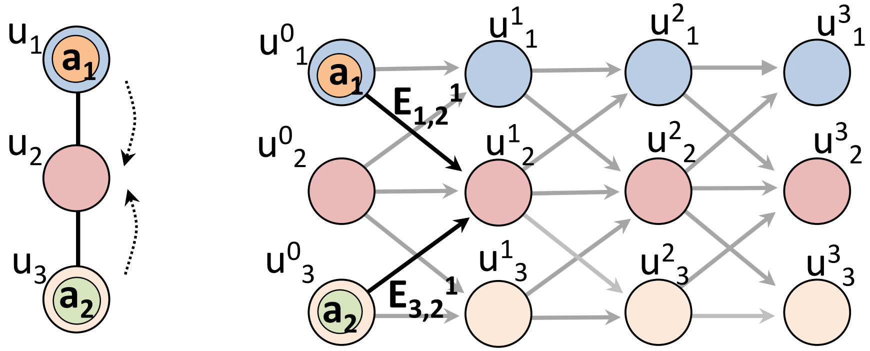


Introduce Constraints **Lazily** [Surynek, IJCAI 2019]

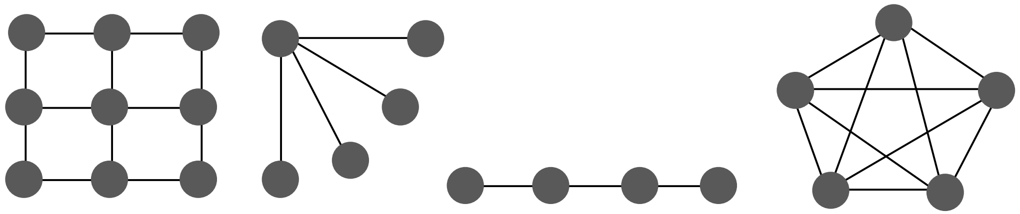
SMT – Satisfiability Modulo Theory

- **SAT Solver**
  - works on top of propositional skeleton – only decision variables (nodes $\mathbf{X(a_i)_u^t}$, edges $\mathbf{E(a_i)_{u,v}^t}$)
  - no understanding of MAPF rules
- **DECIDE$_{\mathbf{MAPF}}$**
  - complete understanding of MAPF rules
  - checks the assignment from the SAT solver
    - returns **conflict elimination constraints**





$$\neg E_{1,2}^1 \vee \neg E_{3,2}^1$$



Experiments - small graphs

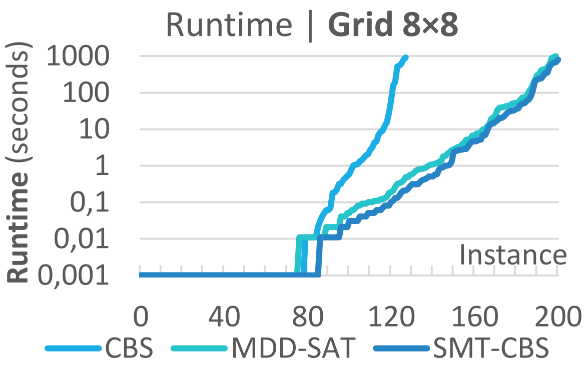
Various types of graphs

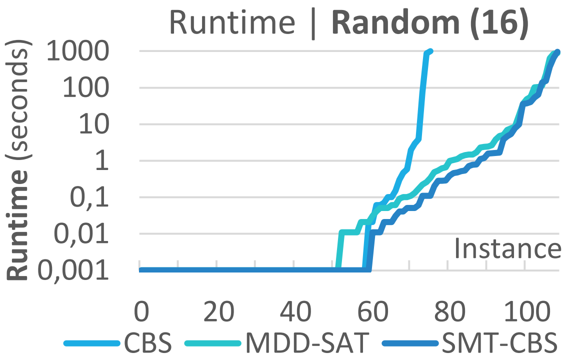
- 4-conneced grids
- Stars
- Paths
- Cliques
- Random graphs (50% edges)

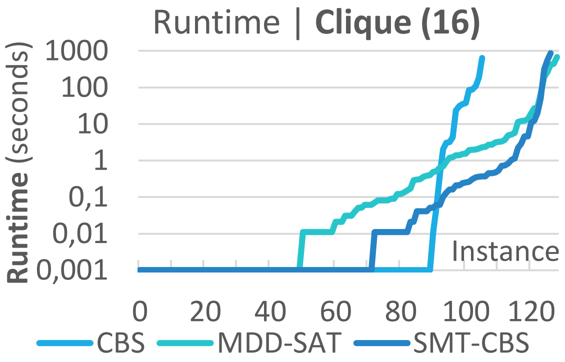
Up to 16 agents

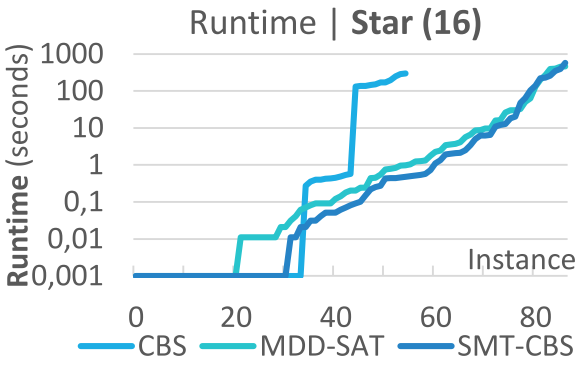
Results

- significant improvement from previous SAT-based solving
- degeneration towards complete formula in hard cases



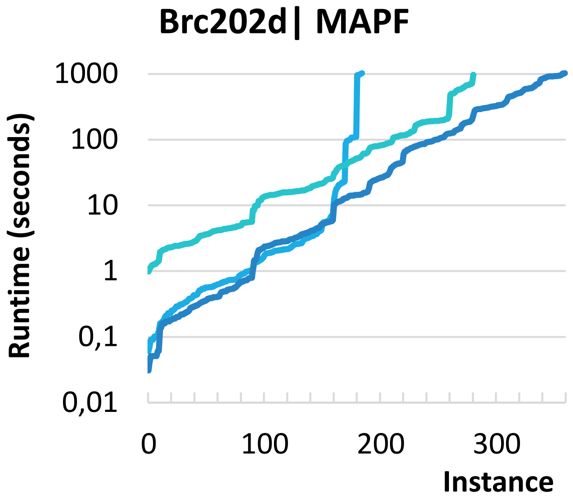


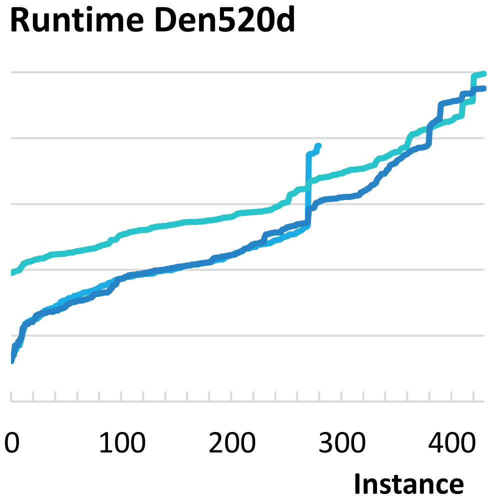




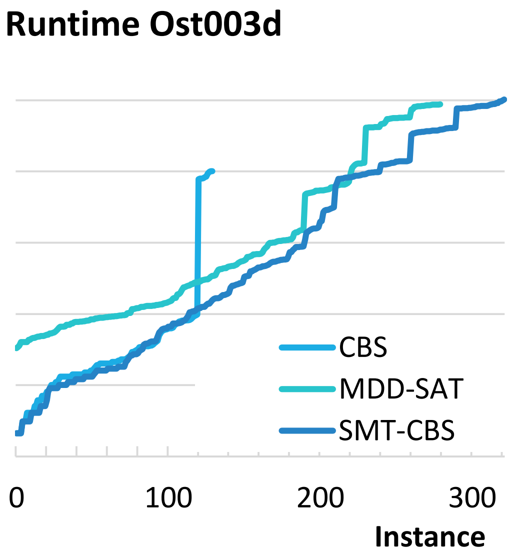
Experiments - large graphs

Big 4-connected grids
- Dragon Age game
- Size:
  - **481x530** (brc202d), **257x256** (den520d), **194x194** (ost003d)
- up to 64 agents





Results
- lazy encoding helps much more in large cases
  - better chance that agents do not collide



Conclusion

Not everything in SAT-based MAPF has been covered

- finding suboptimal solutions using SAT
- various encodings of constraints
  - Boolean circuits for calculating objectives
- log-space representation of decision variables

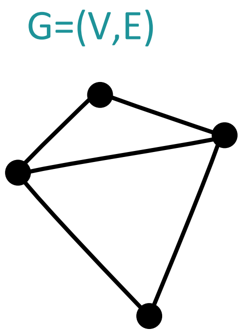
Variants of MAPF

- multiple agents per vertex
- adversarial MAPF
  - multiple teams of agents compete
- …

Further reading

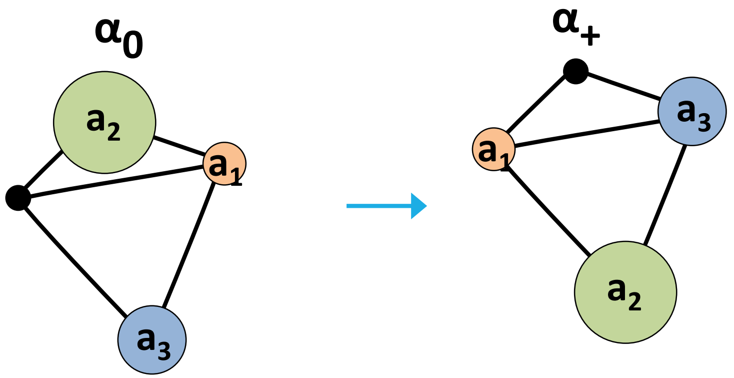
- web site: mapf.info [Koenig, 2019]
- community is growing around MAPF
  - MAPF session and workshop at IJCAI 2019

Future Work: Continuous MAPF



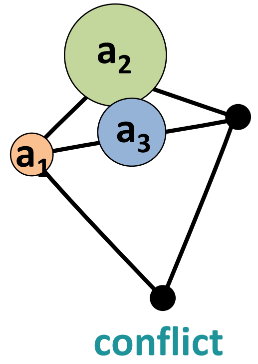
MAPF^R

- **environment** G=(V,E)
  - each vertex has a position
- **agents** $A = \{a_1, a_2, \ldots, a_k\}$
  - each circular agent has
    - constant velocity
    - diameter



Movements

- agents move along straight lines connecting vertices
- agents‘ bodies must not overlap



Methods – SAT again (SMT more precisely)

- not only lazy constraint generation but also lazy decision variable generation

Thank you