Dynamic Epistemic Logic

7. Multi-Agent PathFInding with Destination Uncertainty

Albert-Ludwigs-Universität Freiburg

Bernhard Nebel and Robert Mattmüller

July 10th, 2019

Credits

Credits:

- This presentation is based on our JAIR paper: "Implicitly Coordinated Multi-Agent Path Finding under Destination Uncertainty: Success Guarantees and Computational Complexity"

Motivation

Motivation

- All the examples of epistemic planning we have seen are rather simple.
- → Are there more complex settings, which could be dealt with in the framework?
- There are no success guarantees for the execution of joint plans in the general case.
- → Are there possibly specialized epistemic planning situations, for which postive results are possible?
- Propositional epistemic planning in general is undecidable (something, we have not seen yet).
- → Are there decidable special cases?
- ⇒ MAPF/DU – originally only used as a motivating example.

(Classical) Multi-Agent Path Finding

Multi-agent path finding

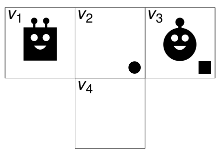
Definition (Multi-agent path finding (MAPF) problem)

Given a set of *agents* A, an undirected, simple *graph* $G = (V, E)$, an *initial state* modelled by an injective function $\alpha_0 : A \to V$, and a *goal state* modelled by another injective function α_*, can α_0 be *transformed* into α_* by *movements of single agents* without collisions?

- *Existence problem*: Does there exist a successful sequence of movements (= *plan*)?
- *Bounded existence problem*: Does there exist a plan of a given *length k* or less?
- *Plan generation problem*: Generate a plan.
- *Optimal plan generation problem*: Generate a shortest plan.

Example

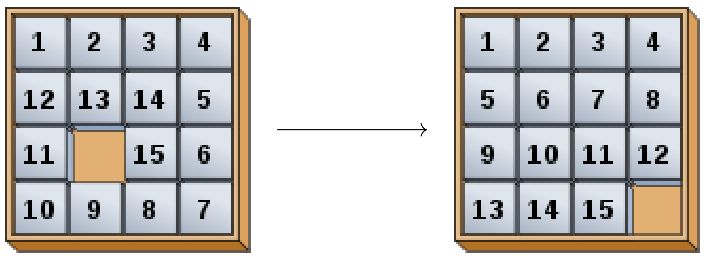
Can we find a (central) plan to move the square robot S to v_3 and the circle robot C to v_2?



$G = (V, E)$ with $V = \{v_1, v_2, v_3, v_4\}$ and $E = \Big\{\{v_1, v_2\}, \{v_2, v_3\}, \{v_2, v_4\}\Big\}$
$A = \{S, C\}$ and $\alpha_0(S) = v_1, \alpha_0(C) = v_3, \alpha_*(S) = v_3, \alpha_*(C) = v_2$

Plan: (C, v_3, v_2), (C, v_2, v_4), (S, v_1, v_2), (S, v_2, v_3),(C, v_4, v_2).

A special case: 15-puzzle



Pictures from Wikipedia article on 15-Puzzle

Lecture plan

- MAPF: variations & complexity
- Distributed MAPF (each agent plans on it own): DMAPF
- Distributed MAPF with destination uncertainty: MAPF/DU

Sequential MAPF

- *Sequential MAPF* (or pebble motion on a graph) allows only one agent to move per time step.
- An agent $a \in A$ can move in one step from $s \in V$ to $t \in V$ transforming α to α', if
  - $\alpha(a) = s$,
  - $\langle s, t \rangle \in E$,
  - there is no agent b such that $\alpha(b) = t$.
- In this case, α' is determined as follows:
  - $\alpha'(a) = t$,
  - for all agents $b \neq a : \alpha(b) = \alpha'(b)$,
- One usually wants to minimize the number of single movements (= *sum-of-cost* over all agents)

Parallel MAPF

- *Parallel MAPF* allows many agents to move in parallel, provided they do not collide.
- Two models:
  - *Parallel*: A chain of agents can move provided the first agent can move on an unoccupied vertex.
  - *Parallel with rotations*: A closed cycle can move synchronously.
- In both cases, one is usually interested in the number of parallel steps (= *make-span*).
- However, also the sum-of-cost is sometimes considered.

Anonymous MAPF

- There is a set of agents and a set of targets (of the same cardinality as the agent set).
- Each target must be reached by one agent.
- This means one first has to assign a target and then to solve the original MAPF problem.
- Interestingly, the problem as a whole is easier to solve (using flow-based techniques).

Computational Complexity of MAPF

- Existence: For arbitrary graphs with at least one empty place, the problem is polynomial ($O(|V|^3)$ using Kornhauser's algorithm). For BIBOX on bi-connected with at least two empty places also cubic, but smaller constant.
- Generation: $O(|V|^3)$, generating the same number of steps, again using Kornhauser's algorithm or BIBOX
- Bounded existence: Is definitely in NP
  - If there exists a solution, then it is polynomially bounded.
  - A solution candidate can be checked in polynomial time for satisfying the conditions of being a movement plan with k of steps or less.
- Question: Is the problem also NP-hard?

The Exact Cover By 3-Sets (X3C) Problem

Definition (Exact Cover By 3-Sets (X3C) Problem)

Given a set of elements U and a collection of subsets $C = \{s_j\}$ with $s_j \subseteq U$ and $|s_j| = 3$. Is there a sub-collection of subsets $C' \subseteq C$ such that $\bigcup_{s \in C'} s = U$ and all subsets in C' are pairwise disjoint, i.e., $s_a \cap s_b = \emptyset$ for each $s_a, s_b \in C'$ with $s_a \neq s_b$?

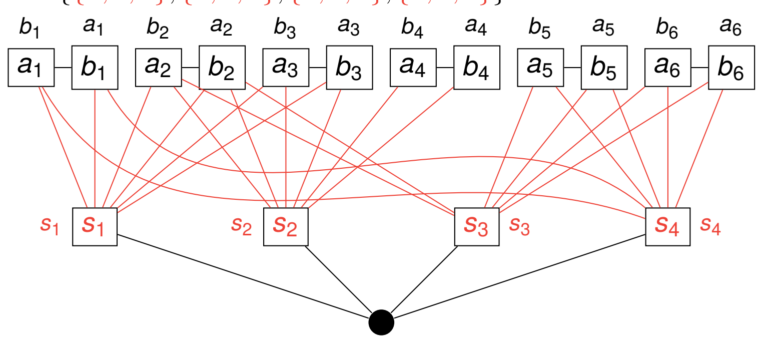
X3C is NP-complete.

Example

$U = \{1,2,3,4,5,6\}$
$C = \{\{1,2,3\},\{2,3,4\},\{2,5,6\},\{1,5,6\}\}$
$C'_1 = \{\{1,2,3\},\{2,3,4\}\}$ is not a cover.
$C'_2 = \{\{1,2,3\},\{2,3,4\},\{1,5,6\}\}$ is not an exact cover.
$C'_3 = \{\{2,3,4\},\{1,5,6\}\}$ is an exact cover.

NP-hardness of MAPF: Reduction from X3C

$C = \{\{1,2,3\},\{2,3,4\},\{2,5,6\},\{1,5,6\}\}$



Squares represent agents, circles are empty vertices, node labels denote destinations

Claim: There is an exact cover by 3-sets iff the constructed MAPF instance can be solved in at most $k = 11/3|U|$ moves.

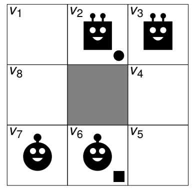
Going Beyond: DMAPF and MAPF/DU

Going beyond: DMAPF and MAPF/DU

- The common goal of all agents is still that everybody reaches its destination.
- Distributed MAPF (DMAPF):
  - All agents plan and re-plan without communicating with their peers.
  - → Models multi-robot interactions without communication
- MAPF under destination uncertainty (MAPF/DU):
  - All agents know their own destinations, but these are not common knowledge any longer.
  - For each agent, there exists a set of possible destinations, which are common knowledge.
  - → Models multi-robot interactions without communication and with goal uncertainty

DMAPF

- Note: Full observability = uniform knowledge
- General results apply: Joint execution successful if agents are optimally eager



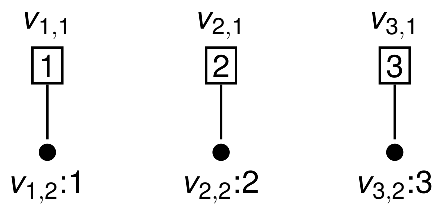
- S initially: $\langle (S, v_2, v_3), \ldots \rangle$
- C initially: $\langle (C, v_6, v_5), \ldots \rangle$
- C after (S, v_2, v_3): $\langle (C, v_6, v_7), \ldots \rangle$

Complexity of planning optimally eager

- Note that solving MAPF optimally implies NP-hardness, while in general (sub-optimal) MAPF is polynomial.
- An alternative to avoid infinite executions is to plan conservatively: When replanning, consider the already executed part as a prefix of the plan from the original configuration (history dependent policy!).
- → Because plans should be cycle-free, no agent will ever consider to revisit a previously visited configuration!
- We do not have to solve NP-hard problems every time we (re-)plan!

Problems with conservative eager agents …

- In the worst-case (if scheduled unfavorably), the agents might consider all of the exponentially many configurations



- Assuming here a schedule similar to a Gray counter (a counter that changes only one bit at a time), we visit all possible configurations.
- In addition, the rule-based polynomial planing methods may not create the right plan prefixes, except when all agents use the identical algorithm.

MAPF/DU

MAPF/DU representation & state space

- In addition to the sets of agents A, the graph $G = (V, E)$, and the assignment of agents to nodes α, we need a function to represent the *possible destinations* $\beta : A \to 2^V$.
- We assume that the set of possible destinations are pairwise disjoint (this can be relaxed, though).
- An *objective state* is given by the pair $s = \langle \alpha, \beta \rangle$ representing the common knowledge of all agents.
- A *subjective state* of agent i is given by $s^i = \langle \alpha, \beta, i, v \rangle$ with $v \in \beta(i)$ or $v = \bot$ if $\beta(i) = \emptyset$, representing the private knowledge of agent i.
- A *MAPF/DU instance* is given by $\langle A, G, s_0, \alpha_* \rangle$, where $s_0 = \langle \alpha_0, \beta_0 \rangle$.

Motivation
Multi-Agent Pathfinding
Going Beyond
MAPF/DU
Implicitly Coordinated Branching Plans
Strong plans
Stepping Stones
Execution cost
Execution guarantees
Complexity
Summary & Outlook
Literature

MAPF/DU viewed as epistemic planning

- Goal specification is complicated since all agents should reach their true destination.

→ Can be specified using a conjunction of implications: If x is i's destination, then try to achieve a configuration where i is on x.

- In order to form an implicitly coordinated plan, the last agent moving needs to know that all other agents have reached their destinations.

→ Use a public announcement when destination has been reached so that true destination becomes common knolwedge. We require that the agent does not move afterwards, but this can be varied.

→ Common goal can be stated as common knowledge goal.

Motivation
Multi-Agent Pathfinding
Going Beyond
MAPF/DU
Implicitly Coordinated Branching Plans
Strong plans
Stepping Stones
Execution cost
Execution guarantees
Complexity
Summary & Outlook
Literature

Simplifying properties of MAPF/DU

- Since there are no private announcements and no non-deterministic effects, the update operation is simple.

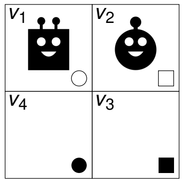
→ Simple ontic or public announcement update, never adding any worlds.

- Perspective-shifts are also simple.

→ If shifting perspective to agent i, simply assume all combinations of destinations from the agents as possible. Perhaps branch over i's possible destinations.

Motivation
Multi-Agent Pathfinding
Going Beyond
MAPF/DU
Implicitly Coordinated Branching Plans
Strong plans
Stepping Stones
Execution cost
Execution guarantees
Complexity
Summary & Outlook
Literature

MAPF/DU: Implicitly coordinated branching plans



- Square agent S wants to go to v_3 and knows that circle agent C wants to go to v_1 or v_4.
- C wants to go to v_4 and knows that S wants to go to v_2 or v_3.
- Let us assume S forms a plan in which it moves in order to empower C to reach their common goal.
- S needs *shifting its perspective* in order to plan for all possible destinations of C (*branching on destinations*).
- Planning for C, S must *forget* about its own destination.

Branching plans: Building blocks

Branching plans consist of:

- *Movement actions*: $(\langle agent\rangle, \langle sourcenode\rangle, \langle targetnode\rangle)$, i.e., a movement of an agent
- *Success announcement*: $(\langle agent\rangle, \mathcal{S})$, after that all agents know that the agent has reached its destination and it cannot move anymore
- *Perspective shift*: $[\langle agent\rangle : \ldots]$, i.e., from here on we assume to plan with the knowledge of agent $\langle agent\rangle$. This can be unconditional or conditional on $\langle agent\rangle$'s destinations.
- *Branch on all destinations*: $(?\langle dest_1\rangle\{\ldots\}, \ldots, ?\langle dest_n\rangle\{\ldots\})$, where all destinations of the current agent have to be listed. For each case we try to find a successful plan to reach the goal state.

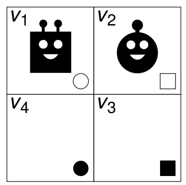
Motivation
Multi-Agent Pathfinding
Going Beyond
MAPF/DU
Implicitly Coordinated Branching Plans
Strong plans
Stepping Stones
Execution cost
Execution guarantees
Complexity
Summary & Outlook
Literature

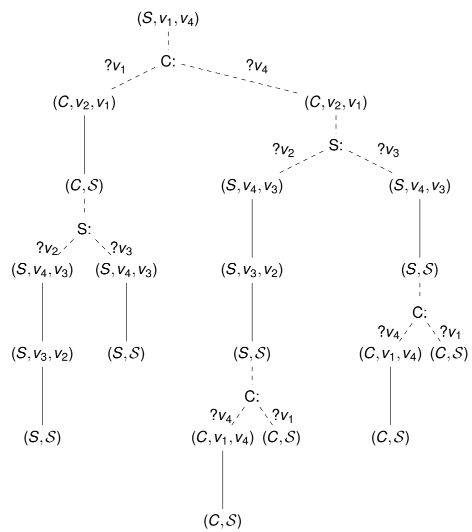
Semantics of branching plans

- Movement actions modify α in the obvious way.
- A success announcement of agent i transforms β to β' such that $\beta'(i) = \emptyset$ in order to signal that i cannot move anymore.
- A perspective shift from i to j with subsequent branching on destinations transforms the subjective state $s^i = \langle \alpha, \beta, i, v_i\rangle$ to a set of subjective states $s^{jk} = \langle \alpha, \beta, j, v_{j_k}\rangle$ with all $v_{j_k} \in \beta(j)$.
- A perspective shift from i to j without subsequent branching on destinations induces the same transformation, but enforces that the subsequent plans are the same for all states subjective states s^{jk}.

Motivation
Multi-Agent Pathfinding
Going Beyond
MAPF/DU
Implicitly Coordinated Branching Plans
Strong plans
Stepping Stones
Execution cost
Execution guarantees
Complexity
Summary & Outlook
Literature

Branching plan: Example





Motivation
Multi-Agent Pathfinding
Going Beyond
MAPF/DU
Implicitly Coordinated Branching Plans
Strong plans
Stepping Stones
Execution cost
Execution guarantees
Complexity
Summary & Outlook
Literature

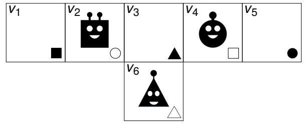
Strong plans

Similar to the notion of strong plans in non-deterministic single-agent planning, we define *i-strong plans* for an agent i to be:

- *cycle-free*, i.e., not visiting the same objective state twice;
- *always successful*, i.e. always ending up in a state such that all agents have announced success;
- *covering*, i.e., for all combinations of possible destinations of agents different from i, success can be reached.

Subjectively and objectively strong plans

- A plan is called *subjectively strong* if it is i-strong for some agent i.
- A plan is called *objectively strong* if it is i-strong for each agent i.
- An instance is *objectively* or *subjectively solvable* if there exists an objectively or subjectively strong plan, respectively.

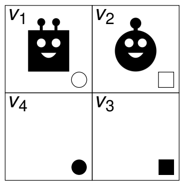


→ There does not exist a T-strong plan, but an S- and a C-strong plan.

- Difference between subjective and objective solvability concerns only the first acting agent!

Structure of strong plans: Stepping stones

- A *stepping stone* for agent i is a state in which i can move to each of its possible destinations, announcing success, and afterwards, for each possible destination, there exists an i-strong plan to solve the resulting states.
- S can create a stepping stone for C by moving from v_1 via v_4 to v_3.
- C can now move to v_1 or v_4 and announce success.
- In each case, S can move afterwards to its destination (or stay) and announce success.



Stepping Stone Theorem

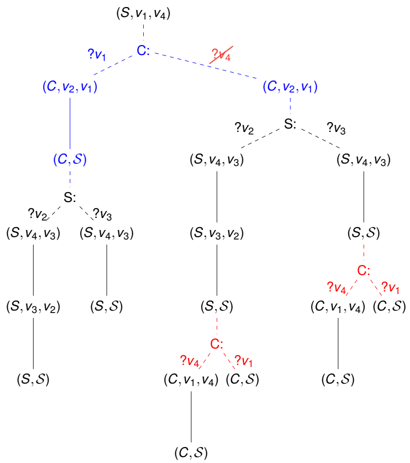
Theorem

Given an i-solvable MAPF/DU instance, there exists an i-strong branching plan such that the only branching points are those utilizing stepping stones.

Proof sketch.

Remove non-stepping stone branching points by picking one branch without success announcement. □

Proof by example



Execution cost

The *execution cost* of a branching plan is the number of atomic actions of the longest execution trace.

Theorem

Given an i-solvable MAPF/DU instance over a graph $G = (V, E)$, then there exists an i-strong branching plan with execution cost bounded by $O(|V|^4)$.

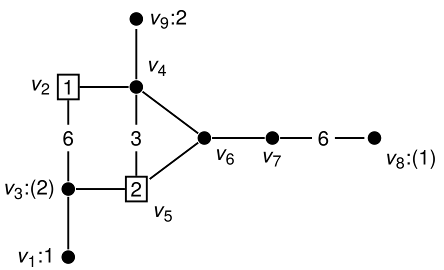
Proof sketch.

Direct consequence of the stepping stone theorem and the maximal number of movements in the MAPF problem. □

Joint execution and execution guarantees

- Joint execution is defined similarly to the fully observable case: One agent is chosen; afterwards the plan is tracked or the agent has to replan.
- In the MAPF/DU framework not all agents might have a plan initially!
- One might hope that optimally eager agents are always successful.
- In epistemic planning this was proven to be true only in the *uniform knowledge* case.
- We do not have uniform knowledge . . . and indeed, execution cycles cannot be excluded.

A counter example



A number on an edge means that there are as many nodes on a line.

- Agent 2 has a shortest eager plan moving first to v_6.
- Agent 1 has then a shortest eager plan moving first to v_4.
- Agent 2 has then a shortest eager plan moving first to v_5.
- Agent 1 has then a shortest eager plan moving first to v_2.

Conservatism

- Perhaps conservatism can help!
- Similarly to DMAPF, conservative replanning means that the already executed actions are used as a prefix in the plan to be generated.
- Differently from DMAPF, we assume that after a success announcement, the initial state is modified so that the *real destination* of the agent is known in the initial state.
- Otherwise we could not solve instances that are only subjectively solvable.

Conservative, optimally eager agents

- Conservative, eager agents are always successful, but might visit the entire state space before terminating.
- Adding optimal eagerness can help to reduce the execution length.

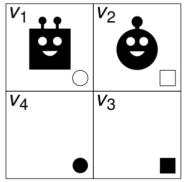
Theorem

For solvable MAPF/DU instances, joint execution and replanning by conservative, optimally eager agents is always successful and the execution length is polynomial.

Proof idea.

After the second agent starts to act, all agents have an identical perspective and for this reason produce objectively strong plans with the same execution costs, which can be shown to be bounded polynomially using the stepping stone theorem. □

Conservative replanning example



- Assume S moves first to v_4.
- Assume C re-plans. From now on, in replanning from the beginning, it has to do a perspective shift to S, because it now has to extend the partial plan starting with (S, v_4, v_1), i.e., it has to create an objectively strong plan.
- Assume that C moves now to v_1.
- From now on, also S has to make a perspective shift to C, effectively "forgetting" its own destination, i.e., it also has to create a objectively strong plan.

Computational Complexity of MAPF/DU planning

Computational Complexity of MAPF/DU planning

- MAPF planning is a polynomial problem.
- DMAPF is NP-hard, because we need optimal MAPF solutions.
- What about MAPF/DU?

→ We look at the problem of deciding whether exists a plan with execution costs k (decision problem corresponding to the optimization problem)

Computational Complexity: Complexity classes P and NP

Problems are categorized into *complexity classes* according to the requirements of computational resources:

- The class of problems decidable on *deterministic Turing machines* in *polynomial time*: P
  - Problems in P are assumed to be *efficiently solvable* (although this might not be true if the exponent is very large)
  - In practice, a reasonable definition
- The class of problems decidable on *non-deterministic Turing machines* in *polynomial time*, i.e., having a poly. length accepting computation for all positive instances: NP
- More classes are definable using other resource bounds on time and memory

Computational Complexity: PSPACE

There are problems even more difficult than NP …

Definition ((N)PSPACE)

PSPACE (NPSPACE) is the class of decision problems that can be decided on deterministic (non-deterministic) Turing machines using only *polynomially many tape cells*.

Some facts about PSPACE:

- PSPACE is *closed under complements* (… as all other deterministic classes)
- PSPACE is *identical* to NPSPACE (because non-deterministic Turing machines can be simulated on deterministic TMs using only quadratic space: Savitch's Theorem)
- NP⊆PSPACE (because in polynomial time one can “visit” only polynomial space, i.e., NP⊆NPSPACE)

this is true.

Computational Complexity: PSPACE-completeness

Definition (PSPACE-completeness)

A decision problem (or language) is PSPACE-complete if it is in PSPACE and all other problems in PSPACE can be polynomially reduced to it.

PSPACE-complete problems appear to be “harder” than *NP-complete* problems from a *practical point of view*.

An example for a PSPACE-complete problem is QBF:

Instance: A Boolean formula φ with quantification over the variables in φ:

$$\forall x_1 \exists x_2 \dots \forall x_n : \varphi$$

Question: Is the QBF formula true under all assignments described by the quantification?

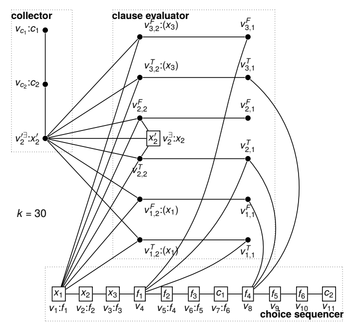
Computational complexity of MAPF/DU bounded plan existence

Theorem

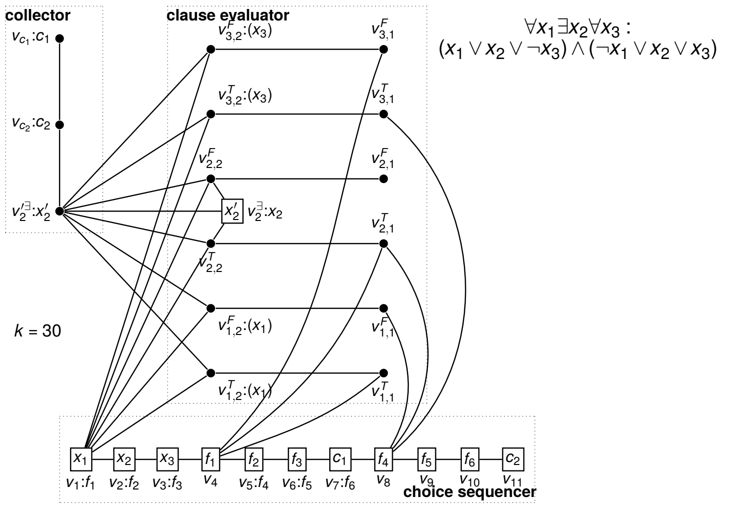
Deciding whether there exists an eager MAPF/DU i-strong or objectively strong plan with execution cost k or less is PSPACE-complete.

Proof sketch.

Since plans have polynomial depth, all execution traces can be generated non-deterministically and tested using only polynomial space, i.e., PSPACE-membership. For hardness, reduction from QBF. Example construction for $\forall x_1 \exists x_2 \forall x_3$: $(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$



The reduction enlarged



Complexity with a fixed number of agents

These results probably imply that the technique could not be used online.

For a fixed number of agents, however, the bounded planning problem is polynomial.

Theorem

For a fixed number c of agents, deciding whether there exists a MAPF/DU i-strong or objectively strong plan with execution cost of k or less can be done in time $O(n^{c^2+c})$.

That means, for two agents, it takes “only” $O(n^6)$ time – but in practice it should be faster.

An algorithm for generating an objective MAPF/DU plan for two agents

1. Determine in the *state space of all node assignments* the distance to the initial state using Dijkstra: $O(|V|^4)$ time.
2. For each of the $O(|V|^2)$ configurations check, whether it is a *potential stepping stone* for one agent, i.e., whether all potential destinations of this agent are reachable using Dijkstra on the modified graph, where the other agent blocks the way: $O(|V|^4)$ time.
3. For all $O(|V|^2)$ potential stepping stones, check whether for each of the $O(|V|)$ possible destination of the first agent, the second agent can reach its possible destinations and use Dijkstra to compute the shortest path: altogether $O(|V|^5)$ time.
4. Consider all stepping stones and minimize over the maximum plan depth. Among the minimal plans select those that are eager for the planning agent.

Summary & Outlook

Summary

- DMAPF generalizes the MAPF problem by dropping the assumption that plans are generated centrally and then communicated.
- MAPF/DU generalizes the MAPF problem further by dropping the assumptions that destinations are common knowledge.
- A solution concept for this setting are *i*-strong branching plans corresponding to implicitly coordinated policies in the area of epistemic planning.
- The backbone of such plans are stepping stones.
- Joint execution can be guaranteed to be successful and polynomially bounded if all agents are conservative and optimally eager.
- While plan existence in general is PSPACE-complete, it is polynomial for a fixed number of agents.

Outlook

→ Do the results still hold for planar graphs?
- Is MAPF/DU plan existence also PSPACE-complete?
- How would more general forms of describing the common knowledge about destinations affect the results?

→ Overlap of destinations or general Boolean combinations
- Can we get similar results for other execution semantics?

→ Concurrent executions of actions
- Can we be more aggressive in expectations about possible destinations?

→ Use forward induction, i.e., assume that actions in the past were rational.
- Are other forms of implicit coordination possible?

→ More communication? Coordination in competitive scenarios?

Literature

Literature (1)

D. Kornhauser, G. L. Miller, and P. G. Spirakis.
Coordinating pebble motion on graphs, the diameter of permutation groups, and applications.
In *25th Annual Symposium on Foundations of Computer Science (FOCS-84)*, pages 241–250, 1984.

A. Felner, R. Stern, S. E. Shimony, E. Boyarski, M. Goldenberg, G. Sharon, N. R. Sturtevant, G. Wagner, and P. Surynek.
Search-Based Optimal Solvers for the Multi-Agent Pathfinding Problem: Summary and Challenges.
In *Proceedings of the Tenth International Symposium on Combinatorial Search (SOCS-17)*, pages 29–37, 2017.

B. Nebel, T. Bolander, T. Engesser and R. Mattmüller.
Implicitly Coordinated Multi-Agent Path Finding under Destination Uncertainty.
Journal of Artificial Intelligence Research **64**: 497-527, 2019.