MAPF algorithms

Non optimal

Algorithm: Prioritized Planning -Explanation: Prioritized Planning assigns priority levels to agents based on their importance and solves subproblems in a prioritized order. It is complete, but the solutions may not be optimal due to the disruption caused by higher-priority agents to lower-priority agents' plans.
Efficient for solving multi-agent pathfinding problems by prioritizing agents based on their importance. Inefficient for problems with conflicting priorities or complex dependencies.

Optimal

Algorithm: ICTS

Algorithm: CBS -Explanation: Two-level algorithm, first finds a conflict-free path assignment using a high-level search tree and then resolves conflicts at the low-level. It guarantees optimality when using optimal low-level solvers. Efficient for solving multi--agent pathfinding problems with conflicting paths and bottlenecks. Inefficient for problems with a large number of agents or complex environments.

Algorithm: M-star -Explanation: A single-agent pathfinding algorithm that uses lazy search to adapt to dynamic obstacles in grid-based environments. Efficient for solving single-agent pathfinding problems in grid-based environments with an unbounded number of obstacles. Inefficient for problems with a large number of agents or complex terrain.

Algorithm: A-star OD ID -Explanation: A-star OD is a variant of A-star where 1 agent moves every time. Deeper search but lower branching factors, hopefully can explore less of the tree this way.

MAPF algorithms (cont)

Algorithm: EPE A-star (Enhanced Partial Expansion A-star) -Explanation: EPE A-star selectively expands nodes, focusing on promising areas using a heursitic on actions, to reduce computational overhead.

Independence Detection

Simple Independence Detection

1. Solve optimally each agent separately
2. While some agents conflict
   A.Merge conflicting agents to one group
   B.Solve optimally new group

Independence Detection

Try to avoid conflict with **same cost** before merge

Online Independence Detection

Solve for every group of agents separately, merge groups if necessary.
Advantages: Replan only required agents, Minimize disturbance, maintain snapshot optimality.

Planning Domain Description Language (PDDL)

Fact = a predicate, State = a conjunction of facts, The initial state: all that I know about the world
The goal: a conjunction of facts that I wish true,
Action = a method for moving between states
- Can have preconditions and effects

STRIPS - Problem is <Predicates, Actions, Initial, Goal>

Approach 1- State Space Search. Can search Forward to the goal or backward from the goal. Possible heuristics: Delete relaxation/Abstraction(Remove a predicate)

Approach 2- Partial order planning- try to achieve goals in parallel, Only make choices when conflicts occur.

Approach 3- SAT compilation.

MA-STRIPS

MA-STRIPS - Replace Actions in strips with a set of actions per agent. It has different levels:
Centralized | Centralized but allow parallel execution| Decentralized observations| Decentralized execution| Decentralized planning

Approach 1- STRIPS compilation: If agents can act in parallel, A in STRIPS will be the cartesian product of all A's.
Else, The possible actions are a unition of all actions.

In the parallel setting:

- Full knowledge of other agent' abilities
- Global heuristic estimates, as in centralized planning

In the distributed setting:

- Partial knowledge of other agents' abilities
- Local, less informed heuristic estimates

In the centralized setting:

- Agents represent a natural factoring of the problem

Can we plan in a mostly decoupled way?
Pros: - Can use regular STRIPS planner
Cons: - Does not fit a distributed setting
- Branching factor exponential w. # of agents

Private actions affect and are affected by agent's actions only.

Public actions affect or are affected by other agents.

Approach 2: Planning + CSP

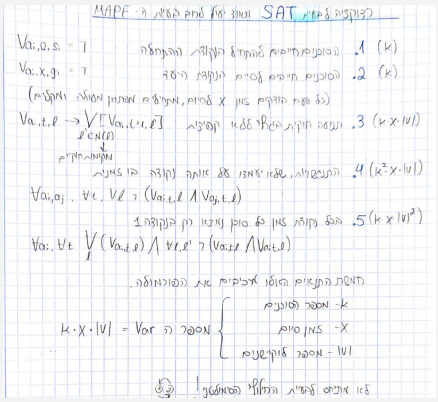
Suboptimal MAPF

Algorithm: Weighted A-Star - A-Star that has a weight factor to prioritize nodes close to the goal by increasing the weight of h in the formula.
Formula: f(n) = g(n) + w * h(n)

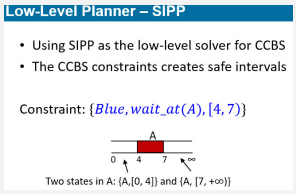
Suboptimal MAPF (cont)

Algorithm: Suboptimal CBS -First expand the routes with the least conflicts OR use single agent A-Star for low leve serach OR use BFS for high level search.

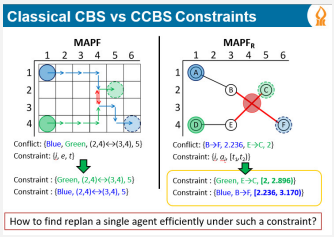
MAPF to SAT Encoding



SIPP- Low Level Planner for Continous

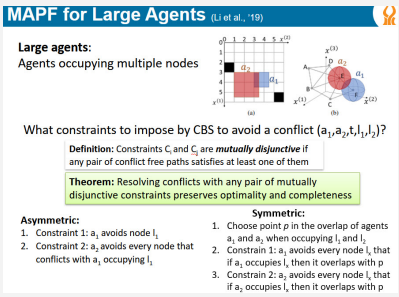


Continous Time



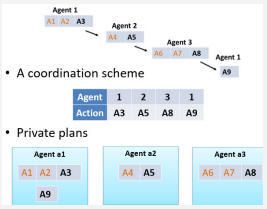
To eliminate a conflict, CBS needs to constrain the colliding agents.
And again in case of classical CBS constr-aints are imposed on locations and contain only one timestep, while in CCBS the one needs to constrain actions. And these constraints need to have time-intervals as the agent can wait for arbitrary amount of time and start to perform the action at any moment.

Large Agents

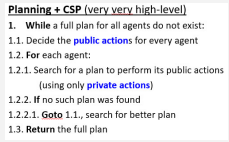


Every grid cell is a node. And agent at a node is defined here as having the top-left of the agent at that node.

MA STRIPS Plan

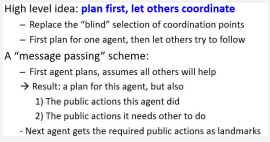


Planning + CSP



Pros -Exploits loosely coupled agents
-Adding agents may only add complexity polynomially
Cons -CSP variables have huge domain
-Regular CSP heuristic are not relevant and regular solvers can't run a planner

Approach 2+: Planning First



Pros- The choice of public actions is more informed
Cons- Which agent goes first?
How to know what the other agents can do?

Greddy Privacy Preserving Planner (GPPP):

- High level planning
  - Outcome: coordination scheme
  - Approach: Solve a relaxed planning problem
- Grounding
  - Input: a coordination scheme
  - Output: a set of private plans to follow the scheme
  - Or false, if not possible
- Notes High level planning
  - Done using a heuristic forward search (GBFS)
  - Each agent generates runs search locally using only private
    actions to identify relevant states to publish
  - Uses heuristics to guide the search
- Notes Grounding
  - Grounding is done in parallel by all agents
  - Agents can use different heuristics/p-lanners

Online MAPF

Replan Single
Plan optimally one by one for new agents
New agent avoids current agents' plans
Replan Single Grouped
Plan optimally for new agents together (MAPF)
New agents avoids current agents' plans
Replan All
Replans optimally all agents when new agents appear
+Is snapshot optimal- optimal solution assuming no new agent appear in the future
-May unnecessarily disturb agents
Possible objective functions: Sum of steps, Number of agents in graph at any time, Sum of steps over shortest path- all equal. Throughput at time x, Maximal service ration (ratio between shortest path and actual path).

Online MAPF variants

Online MAPF Variants

Definition:

Incomplete = problem may be unsolvable

Unsound = some collisions may be unavoidable

	Stay at goal	Disappear
Appears in grid	Incomplete & Unsound	Unsound
Appears in garage	Incomplete	Complete & Sound

Suboptimal ID

Online MAPF Algorithms – Suboptimal ID

- Online MAPF algorithms can not be optimal
- SubID: allow small suboptimality while re-planning
  - ✓ Improve computation time
  - ✓ Decrease number of disturbances to other agents

ITF

The Iterative Taxation Framework (ITF)

- **Path planning (PLAN).**
  - Generates a set of paths with **lowest** penalized cost for every agent.
- **Conflict detection (DETECT).**
  - Detects a set of conflicts between the plans of the agents.
- **Conflict resolution (RESOLVE).**
  - Attempts to resolve these conflicts by adding penalties.

While Condition:
PLAN
DETECT
RESOLVE

- Exhaustive ITA
  - Try to find a taxation to resolves conflicts for **all shortest paths**
- Monte Carlo ITA
  - Simulate actions, resolve conflicts of **this simulations**

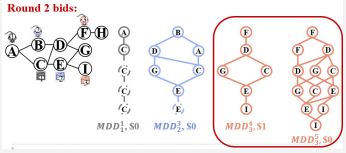
- **Winner penalization** - Winner agrees to pay penalty that the loser cannot "afford".
- **Loser penalization** Loser is motivated to alter its preference.

iBundle

iBundle for MAPF: Price Update

- At the end of each round, prices of all bundles that **unhappy** agents bid on are raised by ε (assume ε = 1)
- This will lead agent 3 to bid on higher cost (longer) paths

Round 2 bids:



Approach 3: Multi Agent Forward Search

Algorithm for agents

1.While (not done)

1.1. Handle received messages

1.2. Extract best state s from OPEN

1.3. For every action a of this agent

1.3.1. s' = generate(a,s)

1.3.2 Add s' to OPEN

1.3.3. If a is public, broadcast it

- Can be optimal! But needs halting mechanism.
- Distributed with distributed heuristics
- Sometimes, MAFS achieves super linear speedup since it exploits the structure of the problem

Action Graph

- Graph to model the interaction between actions
- Nodes correspond to actions
- An aedge (a1, a2) exists iff at least 1 of the following hold
  - ○ a1 achieves a precondition for a2 (or the other way around)
  - ○ a1 destroyed a precondition for a2 (or the other way around)
  - ○ a1 and a2 have conflicting effects
- Key: a partition of the AG induces MA-STRIPS