Lifelong Multi-Agent Path Finding in Large-Scale Warehouses

- A new framework Rolling-Horizon Collission Resolution (RHCR)
  1. Solves lifelong MAPF with changing new goals.
  2. Decomposes the problem into a sequence of Windowed MAPF instances.
  3. ~~The~~ A Windowed MAPF solver resolves collisions among the agents within bounded time horizon
  4. ~~And~~ It ignores collision beyond horizon.

1. Introduction

- Quality of a Multi-Agent Path Finding (MAPF) ~~algo~~ is measured by flowtime and makespan, for the agents moving from start to goal without collission.
- MAPF is NP-hard.
- when an agent reaches goal, it's assigned a new goal, hence life-long MAPF.
- Rolling Horizon Collision Resolution (RHCR)
  It decomposes the MAPF into window instances and replans path in every h-~~instances~~ timesteps for interleaving planning and execution

→ Windowed MAPF is diff as:-
  1. agents are assigned sequence of ~~gat~~ goals with a episode.
  2. collissions ~~are~~ solved only for first w-timesteps.

- By this method, the agents are continuously engaged which increases throughput and it generates pliable plans ~~which~~ to adapt to new goals.

2. Background

2.1 Popular MAPF Solvers

- CBS (Conflict Based Search)
- A complete & optimal 2-level MAPF solver
- The high level, starts with root-node containing shortest individual path for agents, which resolves collission by generating ~~two child~~ binary children & adding a constraint to sort out the collission.
- Then the low-level is called out to replan the paths.

- ECBS (Enhanced CBS)
- complete & bounded sub-optimal, which is achieved by making the solution cost as a user-specified factor away from optimal cost, by focal search rather than best-first search.

- CA* (Cooperative A*)
- incomplete & sub-optimal
- It's a simple prioritized ~~path~~ planning scheme for agent and compares the collissions ~~of~~ with higher priority fixed path.
- small runtime.

- PBS (Priority Based Search)
- ~~S~~ The higher level is similar to CBS, just that it prioritizes the binary children rather than constraints.
- The low level is same as CA*
- Incomplete & sub-optimal.

2.2 Prior MAPF works

Method 1 Break life-long MAPF problem to small problems and we already need to know the goal location.

Method-2 Decompose lifelong MAPF to small sequence of MAPF instances at every time step for all agent.
It works in online setting.

Method-3 Its similar to method-2 but its that it restrict replanning of the agents who just reached their goal., hence may lead to situation where only one agent ~~ma~~ will reach goal in each timestamp.
So, incomplete and costly, hence decrease the overall throughput.

2.3 Bounded Horizon Planning

- a.k.a. Windowed Hierarchial Cooperative A* (WHCA*)
- RHCR produces low computational cost for planning with horizons (bounded), while keeping the agent ~~bust~~ busy, hence the quality only decreases a little.

3. Problem Definition

- The input is a graph $G = (V, E)$ with a set of m agents $\{a_1, a_2, ..., a_m\}$ each with starting point.
- We have an online setting, we don't know all goals.
- We have a 'task assigner' (external), which may or may not be self-independent.
- Timesteps comprise of wait ~~of~~ or move instruction of one unit each.
- A collision will be of vertex conflict or swapping conflict.
- Our task is for collision free paths for all agents to their goals and maximize the throughput.

4. Rolling-Horizon Collision Resolution (RHCR)

- In RHCR, the user specifies time horizon 'w' and replanning period 'h', which specifies Windowed MAPF solver to replan path once every h timesteps., and $(w >= h)$.
- RHCR in every window episode, say at timestep 't', updates the start s_i and goal location sequence g_i.
- It then calculates lower bound on the number of timesteps d that agent a_i to visit other goals in g_i.

$$d = dist(s_i, g_i[0]) + \sum_{j=1}^{|g_i|-1} dist(g_i[j-1], g_i[j])$$

- To avoid ~~idele~~ ideal time, RHCR assigns new locations so that $(d \geq h)$.
- Then RHCR calls Windowed MAPF solver, and move all agents from start to goals according to the sequence, keeping in mind that there's no collision (in w timesteps)
- Finally moves for h timesteps and remove the visited goals from sequences.

4.1 A* for a Goal Location Sequence

Multi-label A* finds path for a single agent with pickup location and goal location.

- For each node we have N.label which tells how many goals have been reached
- It uses location-time A* algo to compute h-value as,

$$h\text{-value} = dist(n, g_i[l]) + \sum_{j=l+1}^{|g_i|-1} dist(g_i[j-1], g_i[j])$$

- The algo first makes root Node R, label = 0 and pushes it to queue OPEN.
- smallest f-value node P is selected
- If it reaches goal (P.Label ++) is done, otherwise children with spatio-temporal constraints are added.

4.2 Bounded Horizon MAPF solvers

- Bounded Horizon (E) CBS
  - We modify the collision detectⁿ function of CBS and (E)CBS, by only finding the collision among all paths that occur at first w timestep.
  - It has smaller-high lvl, hence faster.
- Bounded Horizon CA*
  - Similar as the CA*, in this agent has to avoid collisions with higher agents only during the first w timestamps.
  - It has fewer spatio-temporal constraints, hence faster and high success rate.
- Bounded Horizon PBS
  - We modify the high lvl as we did for BH(E)CBS and the lower lvl as we did for Bounded Horizon CA*.
  - Hence; smaller high level & faster low level.

4.3 Behavior of RHCR
- In Example 1, it exemplifies that sometimes RHCR with lower time horizons achieves higher throughput than larger time horizons.
- In example 2, sometimes small time horizons may lead to deadlock.

4.4 Avoiding deadlocks
- We design a potential function to evaluate the progress of agents and increase the time horizon if the agents don't progress.
- It estimates the no. of agents which need fewer timesteps to visit all their goal locations from timestep w on, than from timestep 0 on.
- RHCR can either focus on throughput or on completeness.

5. Empirical Results
Implemented RMCR based on CBS, ECBS, CA* and PBS.

5.1 Fulfillment Warehouse Application
- Method-3 is applicable in such inventory-pods and work-station infrastructures.
- Initial locations are set to random and task assigner chooses goals at random.
- All methods use PBS as (Windowed) MAPF solvers.
- RMCR outperforms reserving dummy path method
- Runtime is slower
- Replanning at every timestep, leads to lower throughput and not applicable for all maps.

5.2 Sorting Centre Application
- Method-3 is not applicable, as it's not well structured.
- We don't have to resolve swapping conflicts and so we focus on efficiency.
- Small values of w speed up RMCR by a factor of 6 and also yield scalability w.r.t. the no. of agent.
- Eg:- for PBS (w = ∞) → instances are 700 agents
PBS (w = 5) → intances 1000 agents.

5.3. Dynamic Bounded Horizons.
- We try deadlock avoidance
- We use larger value of p and start with smaller value of w.
- We check for Bounded Horizon RMCR by ECBS, CA* & CBS.
- We find horizon for high throughput but induces runtime overhead.

6. Conclusion

- We proposed RMCR and the transformation of ~~RA~~ MAPF to windowed MAPF.
- We empirically show success rate at warehouse maps and sorting centre maps.
- RMCR gives better throughput & run-time than method -3
- RMCR is simple, flexible & powerful.

(IMT2020133 - Darpan Singh)