Disjoint Splitting for MAPF with CBS

CBS is a two-level tree search, which in standard case has non-disjoint splitting to subproblems, which can create duplication. So, we discuss disjoint splitting.

1. Introduction

- MAPF is defined by $G = (V, E)$ with $\{a_1, \ldots, a_k\}$ agents.
- Each agent with start & goal vertex s_i, g_i and we have unit timesteps.
- Vertex conflict $\langle a_i, a_j, v, t \rangle$ & edge conflict $\langle a_i, a_j, u, v, t \rangle$
- to find a solution (conflict free path) and also minimize the summation of path cost.
- Usually non-disjoint subproblems are created by negative constraint.
- We introduce +ive constraint, forcing an agent to be at a given place at particular timestep, hence split to disjoint sub-problem.

2. CBS

- a two level, low level → best first search for shortest path for each agent.
- at high level, best first search on binary constraint tree (CT)
- each CT node N has constraints, paths and cost.
- here the negative constraint $\overline{\langle a_i, v, t \rangle}$ prohibits for that moment.
- Improved CBS ~~—~~ (ICBS) and CBSH are optimal variants. Multi-valued decision Diagram to check cardinal conflict (when splitting the cost of each child $\geq$ cost of N)
- MDD$_i$ for a_i is a directed acyclic graph, consisting of shortest paths of a_i which satisfy

- All nodes at depth t correspond to possible vertices at t.
- It's a singleton if only that node at depth, and cardinal conflict if both conflicting agents have singleton at the conflict.
- Enhanced CBS (ECBS) uses focal search instead of best-first search, which maintains a focal list.

3. Inefficiency of CBS splitting.
- CBS splitting is complete but not disjoint.

4. CBS with disjoint splitting.
- Positive constraint $\langle a_i, v, t \rangle$ forces an agent a_i to be at v at t, while other agent is prohibited.
- For disjoint splitting a conflict $\langle a_i, a_j, v, t \rangle$ we choose one of the to split on a_k ($k \in \{i,j\}$) with children (a_k, v, t) & (a_k, v, t).
- Basically pruning happens, so there's a smaller CT.
- **Proposition 1**: The sets of candidate plans in child CT nodes ~~A, B~~ after disjoint splitting contain no duplicates & their union contains all conflict-free candidate plans in N.

4.1 Low-level search
- Usually we use A* search, but for disjoint splitting we view start & all positive constraints as landmarks.

4.2 High level search
- CT branch in disjoint splitting, not symmetric.
- We select acc to
  (i) Singletons : MDD$_k$ with most singletons
  (ii) Width : MDD$_k$ has fewest nodes at a depth t.

4.3 Related Work

- Disjoint 3 split into :-
  - (i) C_1 : $\{\langle a_i, v, t\rangle, \langle a_j, v, t\rangle\}$
  - (ii) C_2 : $\{\langle a_j, v, t\rangle, \langle a_i, v, t\rangle\}$
  - (iii) C_3 : $\{\langle a_i, v, t\rangle, \langle a_j, v, t\rangle\}$
- which in disjoint splitting, we merge the 2nd and 3rd child.

5. Empirical Evaluation

- We compare CBSH with non-disjoint, Disjoint, Random, Singletons and Width splitting.
- ECBS with non-disjoint & Random splits.
- Cardinal rectangle conflict & corridor conflict
  - ↓ all shortest path conflict with each other.
  - ↓ like edge conflict
- Grids with & without randomly blocked cells
  - the CBSH is better than non-disjoint & Disjoint 3.
  - Width is superior in success rate of splittings.
- Large game maps
  - (CBSH with Disjoint 3) < (CBSH non-disjoint)
  - (CBSH Disjoint) > (CBSH non-disjoint)
- Warehouse maps
  - Disjoint splitting is highly beneficial for both success rate & runtime.
  - Singletons the fastest.
  - For ECBS, disjoint splitting is also beneficial.

6. Conclusions

- The split of CBS is not disjoint, which creates duplicate of effort.
- We introduce disjoint splitting with position constraint.
- Empirically disjoint is atleast at par with CBS.

[Darpan Singh - IMT2020133]