Max Independent Set with Min Weight for solving Multi-Agent Path Finding problems

Sailik Sengupta  Kaustav Basu
CSE 550 Final Project, Arizona State University
{ sailiks, kbasu2 }@asu.edu

Introduction
In the problem of Multi Agent (collision free centralized) Path Finding with discreet time movements (MAPF), the state space of search explodes combinatorially as the search states incorporate information for each agent. The number of actions is also multiplied by the number of agents in the problem. Moreover, this problem has been shown to be NP-Complete Ratner and Warmuth (1986).

In general, solutions for many multi-agent problems in a centralized fashion often come up with local solutions for each agent which can be used as building blocks for constructing a global solution. Each agent’s solution in this regard can be viewed as a local strategy. In our case, this strategy is a path from a particular agent’s initial state to its goal state. We follow this philosophy for reducing the MAPF problem to a new problem of finding the Maximum sized Independent Set with Minimum Weights (ISW) and (would ideally want to) find an approximate algorithm for the same. Note that any multi agent problem that can be solved bottom up in this way can be formulated to our problem.

We now first define our problem, showing that it is also NP-Complete. We then define the MAPF problem with certain relaxations, eventually mapping it to ISW (and empirically showing how our solution can provide a solution to the MAPF problem).

Approach

ISW Problem
In the ISW problem, given a graph $G = (V, E)$ and a function $f: v \rightarrow \mathbb{R}$ $\forall v \in V$, we want to find a independent set of vertices $I \subset V$ such that $\sum_{v \in I} f(v)$ is minimized.

Proof. MISMCV is NP-Complete.
Consider a restricted version of our problem where $\forall v \in V, f(v) = 1$. In this case, our problem reduces to finding the max-sized Independent Set (IS) in the graph $G$, which has already been shown to be NP-Complete. Thus, our ISW problem, which is a more general instance of IS, is NP-Complete.
Mapping Multi-Agent Path Finding (MAPF) to Independent Set with Max Weight (ISW)

In this subsection, we first describe the MAPF problem formally and the key challenges in solving it, which provides a segue into the mapping for MAPF to ISW.

Multi-Agent Path Finding (MAPF)

The MAPF problem has a $k \times k$ grid structure, a set of $n$ agents, and for agent $i$, an initial state $I_i$ and goal state $G_i$. In this setup, each agent fits in a single cell. Time steps are discretized from $\{1 \ldots T\}$ and each agent moves from one cell to one of the four adjoining cells (namely, up, down, right and left) in each time step. For our formulation, we assume that an agent $i$ cannot stop unless it is in its goal state $G_i$. The objective of our problem is to find a path from $I_i$ to $G_i$ for each agent such that two agents are not present in the same square at a single time step, i.e. have a collision. Formally for any time-step $t$, if $p_t(i)$ represents the position of agent $i$ (which is essentially an $(x, y)$ grid square co-ordinate), then $p_t(i) \neq p_t(j)$ if $i \neq j$.

Notice that one can find the shortest path from $I_i$ to $G_i$ in the 4 lattice structure (shown in Figure 1) for each agent in time polynomial in $n$ and grid dimensions. But these paths may not necessarily be collision free. Thus, search in the combined search space of agent positions and actions is prevalent in literature. We now talk about our approach.

Formulation

The key idea of our formulation is to generate $K$-diverse paths for each agent. For an agent $i$, let us represent these paths as $p^j_i$ where $i(\in \{0, \ldots, n-1\})$ denotes the agent and $j(\in \{0, \ldots K\})$ denotes the path. Each of these paths represent vertices of a graph $G = (V, E)$. Thus, $V = \{p^j_i: \forall i \in \{0, \ldots, n-1\}, j \in$
The edge set is defined as follows:

\[ E = \{ (p^i_j, p'^i_j) : \text{iff } i = i' \text{ or } p^i_j \text{ & } p'^i_j \text{ have a collision.} \} \]

The first set of edges are called intra-agent edges and the second set of edges are called the inter-agent edges. It is not hard to see that the \( K \) paths for each agent forms a \( K \) clique for each agent in \( G \).

Now, notice that each path in the graph comes with a cost or distance value, which we would seek to minimize for all agents in the original problem. We associate the cost of each path as a weight to every vertex in our graph. Thus, \( w(p^i_j) = \text{cost}(p^i_j) \).

Given this, an independent will always pick one vertex from each agent clique of size \( K \), which would essentially mean that each agent gets to select a single path it will take from its initial state to its goal state. Also note that two vertices which have an inter-agent edge, i.e. a collision, is never selected in the same independent set. Thus, the max independent set always has collision free paths (hopefully one for each agent). Also, since we want the cost of the paths to be minimized, this max independent set should have the least weighted vertices.

Depending on the value of \( K \), we might have a sub-optimal solution if the diverse paths are also not the top-\( K \) ones in terms of optimality. It is also trivial to see that when the value of \( K < \) all paths paths for each agent, the solution to our problem might not be a feasible solution to the MAPF problem. We discuss some of these instances in the Future Work section and presently talk about formulating this as an optimization problem and solving it.

### Linear Programming solver for ISW

Let \( x^i_j \) be a binary variable which is 1 if a path \( p^i_j \) is part of the solution and 0 otherwise. To make the method general, we consider each agent has an option to select one of the \( K_i \) paths (instead of \( K \) paths for each agent defined in the last section). We define the following optimization method to solve an ISW problem instance.

\[
\sum_{i \in \{0, \ldots, n\}} \sum_{j \in \{0, \ldots, K_i\}} x^i_j \\
\text{s.t. } \sum_{j \in \{0, \ldots, K_i\}} x^i_j = 1 \; \forall \; i \in \{0, \ldots, n\} \\
\quad x^i_j + x^i_{j'} \leq 1 \; \forall \; (x^i_j, x^i_{j'}) \in E & i \neq i' \\
\quad 0 \leq x^i_j \forall \; i, j \\
\quad x^i_j \leq 1 \; \forall \; i, j
\]

The first constraint ensures that, there is only one vertex added to the independent set from each agent clique. The second constrains takes care of the inter-agent edges. The last two constraints represent relaxation of the binary variables from 0 to 1. Notice that, given this we should ideally obtain decimal values for each \( x \) variable, that would represent the membership of a particular vertex to the max independent set with min weight. We now move into the empirical evaluation of this formulations, where we notice that for an extensive set of cases, the Linear Program gets \{0,1\} values for the variables \( x \).
Experimental Section

As a running example, let us consider a map with two agents, where each agent has 4 paths (as shown in Figure 2). Hence, $K_1 = K_2 = 4$. Now the input file looks as follows

```
2
3|5|2|4
3|5|2|4
5
0|2|1|0
0|0|1|0
0|1|1|1
0|2|1|2
0|3|1|3
```

Where the first line represents the number of agents $n$. The $n$ lines that follow have $K_i$ paths for the $i$-th agent with the cost of each path separated by ‘|’. In the above example, there are 2 agents and each agent has four paths with the weights being 3, 5, 2 and 4 for both the agents.

Then follows the number of inter-agent collisions. Each line after that represents a collision where the integers $i, j, i', j'$ are separated by a ‘|’. For example the line 0|2|1|0 represents the collision edge $(p_0^2, p_1^0)$, i.e. collision between the path 2 with weight 2 of agent 0 and path 0 with weight 3 of agent 1.

It is easy to see that given the above setup, our optimization produces the following output:

```
Optimal objective  5.0000000000e+00
---------------------
x-0-0 -> 1
x-0-1 -> 0
x-0-2 -> 0
x-0-3 -> 0
x-1-0 -> 0
x-1-1 -> 0
x-1-2 -> 1
x-1-3 -> 0
---------------------
```

Thus, the agent 0 should select the 0-th path with weight 3 while agent 1 selects the 2-nd
path with weight 2. This results in an total cost of 5, which is our objective value. Notice that the problem formulation just constraints the variables $x$ to be from 0 to 1 and not necessary binary. For our example, they turned out to be binary. Thus, we are probably getting integer values even without trying hard.

Surprisingly enough, this is not the only example. We did an extensive testing with on 201 more randomly generated problems. These problems can be sub-divided into 3 categories— small, medium and large (see Table 1). We ran the experiments with Gurobi on a 12-core 3.4 Ghz Intex Xeon machine with 64 GB of RAM. We saw that for all the instances of the ISW problem, we always land up getting a binary value for the $x$ instead of fractional numbers.

<table>
<thead>
<tr>
<th>Problem size</th>
<th>$n$</th>
<th>AvgK</th>
<th>Collisions</th>
<th>Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>100</td>
</tr>
<tr>
<td>Medium</td>
<td>7</td>
<td>10</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>Large</td>
<td>100</td>
<td>100</td>
<td>5000</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1: Randomly generated test instances solved using the LP approach.

With our observations, we come to the following conclusions:

1. The approach rarely encounters instances where the variables have non-integer solutions. This is similar to the scenario with the simplex algorithm.

2. The instance that produces non-integer solutions has low probability of occurrence and needs cannot be generated randomly. The instance needs to be generated in a non-trivial manner with complex analysis.

3. $P = NP$. This would be a highly improbable.

**Future Work and Discussion**

We plan to use the diverse path generation to generate paths for each agent in the MAPF problem and encode them as an instance of the ISW problem, which would then be solved by our approach. This would be an end to end solution to the MAPF problem. For instances of the MAPF problem that are scarcely populated with agents, inter-agent path collisions are less in number as the value of $K$ paths for each agent increases. As the number of agents increase (i.e., the map is densely populated), the number of inter-agent collisions becomes extremely high. Thus, unless we consider a large number of paths ($K$ is high) for each agent, the optimization problem becomes infeasible.

Now, notice that finding the inter-class collisions among paths of the various agents is non-trivial. The main reason behind this is because we need to have to account for time. An agent may occupy a grid $(x, y)$ at time $t$ if it follows path $p$ while another agent’s path $p'$ passes through $(x, y)$ at a time instance $t + x$ where $0 < x < \min(T - t, t)$. For our example instance (in Figure 2), this can be avoided since the solution paths are strictly collision avoiding, that is given any $t \in \{0, \ldots, T\}$, the location of the two agents are never on the same grid box. We plan to address this in our complete work.
References

D. Ratner and M. K. Warmuth. Finding a shortest solution for the $n \times n$ extension of the

S. G. Vadlamudi and S. Kambhampati. A combinatorial search perspective on diverse