Solving the Multi-Agent Path Finding with Waypoints Problem Using Subdimensional Expansion

Jeroen van Dijk

MAPFW

The Multi-Agent Path Finding (MAPF) problem is given a graph and a set of agents find a set of moves that takes all the agents from the start vertex to the end vertex. Agents can not be at the same vertex or traversing the same edge simultaneously. The optimal solution is the solution where the sum of the costs of all paths is the lowest.

In the Multi-Agent Path Finding with Waypoints (MAPFW) problem every agent also has a set of waypoints it needs to visit in any order.

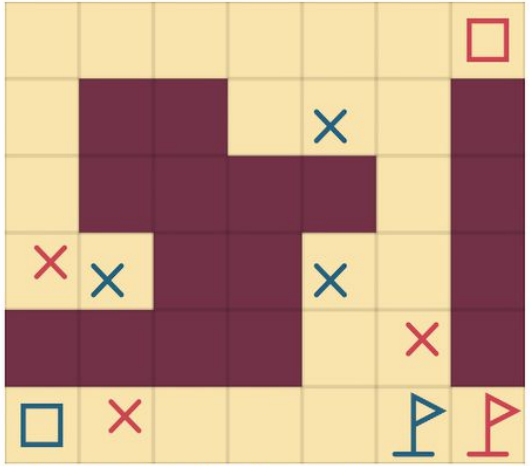


Figure 1: An example of a MAPFW problem instance. Source: mapfw.nl

Supervisors: Mathijs de Weerdt and Jesse Mulderij

M*

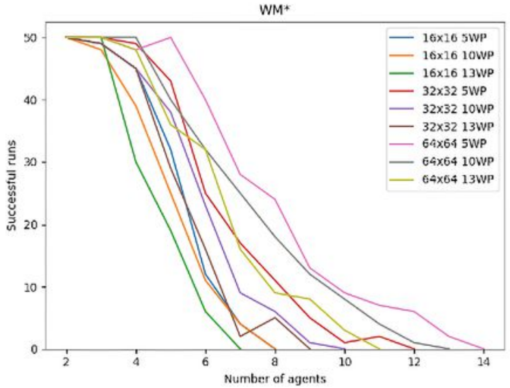
M* is an improved version of A* where agents:

- Follow their optimal policy until they collide or reach their target.
- Explore all neighbours if the optimal policy will result in a collision.

WM*

WM* is the extended M* algorithm that is designed to solve the MAPFW problem.

1. Order the waypoints using a Travelling salesman problem solver.
2. Create a policy for each waypoint and the target.
3. Extend the underlying A* planner to keep track of the visited waypoints.



inflated WM*

Inflated WM* is a variant of WM* where we multiply the heuristic with some $\epsilon > 1$.

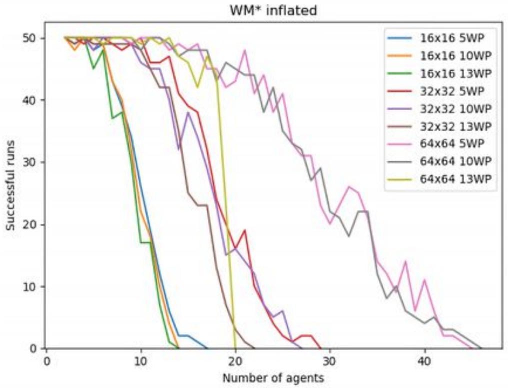


Figure 3: The results of the inflated WM algorithm on the progressive benchmarks.*

Conclusion

	WM*	Inflated WM*
Avg extra cost	<0.3%	<0.5%
Max extra cost	<2.3%	<4.5%
Runtime of individual benchmarks	Faster than optimal algorithms	Comparable to heuristic algorithms
Progressive benchmarks	Comparable to optimal algorithms	Outperforming all algorithms

Algorithm 1 WM*

Generate an optimal policy for every waypoint
Sort the waypoints as described in Section 3.2
$start.target\_indices \leftarrow [0] * n\_agents$
$start.cost \leftarrow 0$
$open \leftarrow v_s$
while $open.empty = False$ **do**
  $current \leftarrow open.pop()$ ▷ Get the cheapest configuration
  if $current = target$ & all the waypoints have been visited **then**
    [We found the solution]
    return $current.back\_ptr + current$
  for $nbr \in get\_neighbours(current)$ **do**
    $nbr.target\_indices \leftarrow current.target\_indices$
    Increment the target index for any agent that is on their target waypoint in nbr
    [ϕ returns the index of the agents in collision]
    $nbr.collisions.update(\phi(nbr, current))$
    $nbr.back\_set.append(current)$
    $backpropagate(current, nbr.collisions, open)$
    $f \leftarrow$ the cost of the move from current to nbr
    if $\phi(nbr, current) = \emptyset$ & $current.cost + f < nbr.cost$ **then**
      $nbr.cost \leftarrow current.cost + f$
      $nbr.back_ptr \leftarrow current.back\_ptr + current$
      $open.push(nbr)$
end

TU Delft