>_ Solving multi-agent path-finding problems with waypoints using A*

Stef Siekman

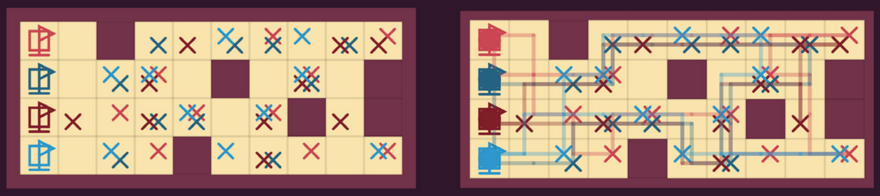
Supervisors
M. de Weerdt, J Mulderij

MAPFW

Given a graph, a set of agents must be assigned a set of moves (wait or move) that brings them from their start to their goal vertex. Agents may not not share vertices or edges at the same time.

Each agent also has a set of waypoints (vertices) they need to visit, in any order.

The optimal solution is that where the sum of the length of paths of all agents is minimal.



Example of a problem instance and an optimal solution

A* + OD + ID

This is an algorithm already proposed 10 years ago to solve MAPF peoblem, i.e. without waypoints. It is used as the basis for the extension with waypoints.

A+ (Very) famous path-finding algorithm, it uses a heuristic to 'direct' it's search, yet guarantees an optimal solution.

OD When applying regular A* to multiple agents, each expansion would be (moves * agents) in size. Operator decompositions only expands a single move per agents per expansion, limiting the size of the search tree.

ID The problem is first solved for agents individually, only when these solutions conflict are they solved cooperatively. Additionally CATs are used to avoid conflicts between groups, and before merging groups an attempt is made to find an alternative solution.

Research question

What is an effective way to extend A* with Operator Decomposition and Indepence Dectection to MAPFW?

+ How can the heuristic used by A* be extended to solve multi-waypoints problems optimally?
+ In what ways can the A* + OD + ID solver be optimised to run faster?
+ How does the best A* + OD + ID implementation compare against other extension for MAPFW?

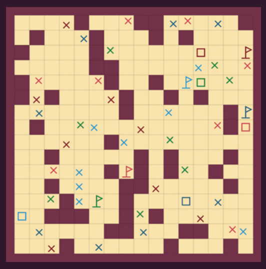
Extension for waypoints

The key concept of the extension is to alter the heuristic used by A*. As long as this heuristic is admissable, correctness of the resulting algorithm is guaranteed. However, there are many options for this heuristic. From initial testing, an optimal one (with regard to a single agent) seems to give the best resuls.

TSP To find the shortest path for each agent via all waypoints, a modified version of the traveling salesman problem is used.

The results from these calculations are then cached to void re-calculating when waypoints are visited or groups are merged.

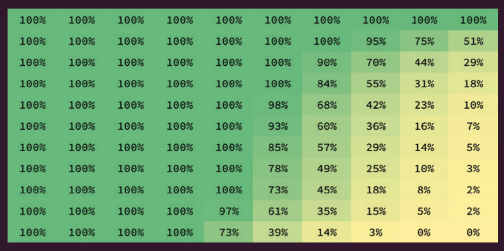
Preliminary results



Example instance

Agents ->

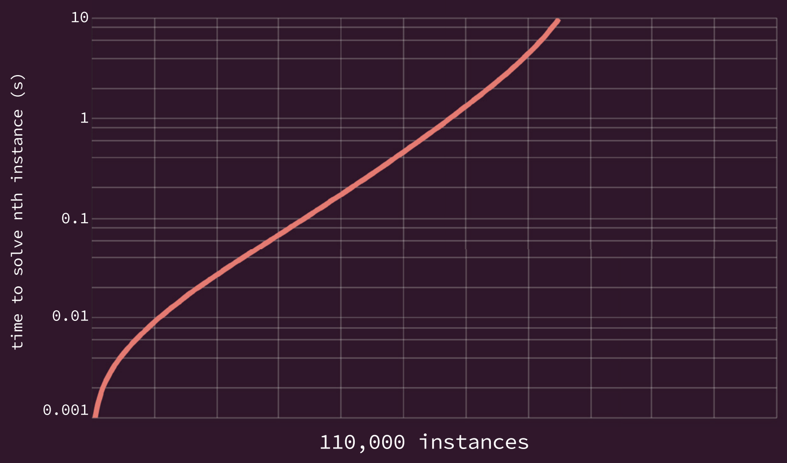
<- Waypoints



100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
100%	100%	100%	100%	100%	100%	100%	95%	75%	51%
100%	100%	100%	100%	100%	100%	90%	70%	44%	29%
100%	100%	100%	100%	100%	100%	84%	55%	31%	18%
100%	100%	100%	100%	100%	98%	68%	42%	23%	10%
100%	100%	100%	100%	100%	93%	60%	36%	16%	7%
100%	100%	100%	100%	100%	85%	57%	29%	14%	5%
100%	100%	100%	100%	100%	78%	49%	25%	10%	3%
100%	100%	100%	100%	100%	73%	45%	18%	8%	2%
100%	100%	100%	100%	97%	61%	35%	15%	5%	2%
100%	100%	100%	100%	73%	39%	14%	3%	0%	0%

% solved in 10 seconds

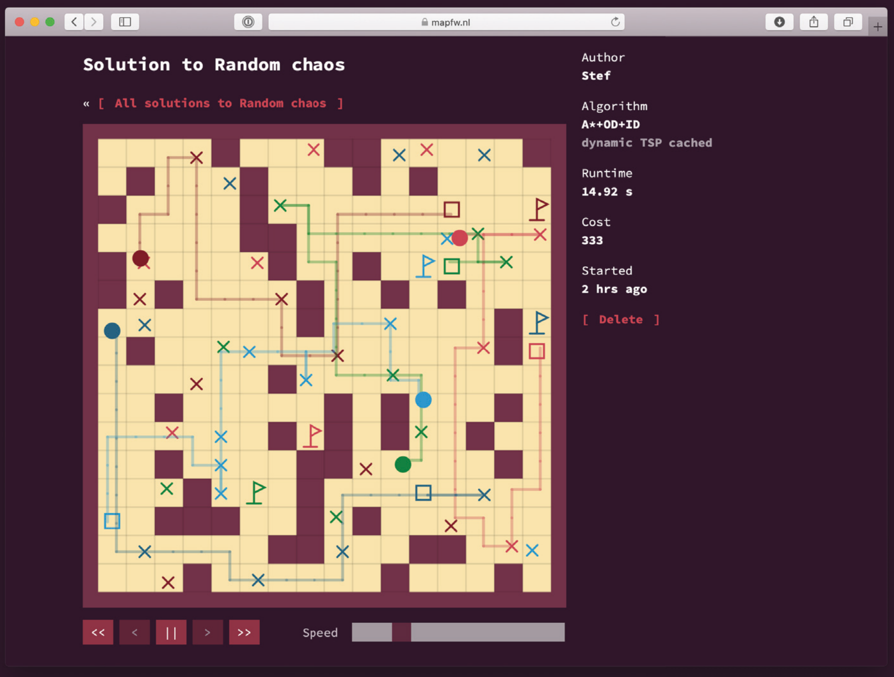
Per problem size: 1,000 16x16 random grids



Comparison

To compare the different version of algoritms, also those of other group members, a website was created to facilitate comparing runtimes and results on the same problem instances.

Want to see it for yourself? Visited **mapfw.nl**



Solution to Random chaos

« [All solutions to Random chaos]

Author
Stef

Algorithm
A*+OD+ID
dynamic TSP cached

Runtime
14.92 s

Cost
333

Started
2 hrs ago

[Delete]

Further work

+ More optimisations to the algorithm
+ Collect experimental results from different versions
+ Extend the website for better experimentation for all the algorithms