Adapting CBM to optimize the Sum of Costs

Robbin Baauw
R.W.Baauw@student.tudelft.nl

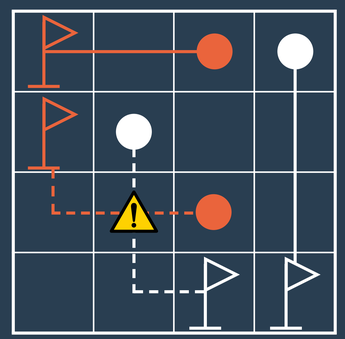
Mathijs de Weerdt
Jesse Mulderij

BACKGROUND

Multi-Agent Pathfinding with Matching

— Multiple agents in a team move through a maze towards a goal belonging to their team
— Each wait/move action has a unit cost
— Collisions are not allowed ⚠

Goal: optimizing the SoC cost metric



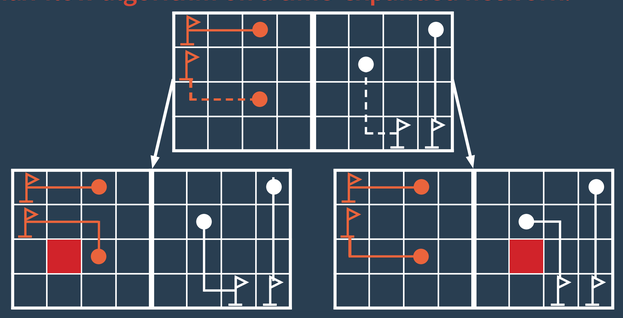
Cost metric

— **Sum of Costs (SoC):** sum of path lengths (15)
— **Makespan:** maximum path length (4)

PRIOR WORK

Conflict Based Min-Cost-Flow (CBM)

Consists of two parts:
— **High-level solver:** detect collisions between paths found by the low-level solver and add constraints in a Constraint Tree.
— **Low-level solver:** optimizes the makespan by solving a network flow problem per team using the constraints imposed by the high-level solver. This is done using a min-cost max-flow algorithm on a time-expanded network.



RESEARCH QUESTION

How can CBM be adapted to minimize the SoC?

— Can the succesive shortest path (SSP) algorithm be used to minimize the SoC?
— How does the run-time performance compare to the baseline makespan CBM?
— Can the CBM edge weight heuristic be adapted to minimize the SoC?

CBM ADAPTATIONS

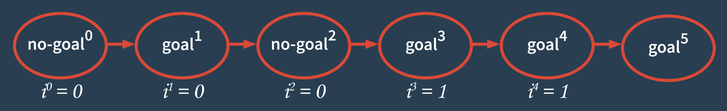
General extensions

— First find the time step for which an optimal SoC can certainly be found
— Re-use the time-expanded graph instead of rebuilding it on each usage
— Two options for solving the problem: Integer Linear Programming (ILP) and Successive Shortest Path (SSP)

ILP w/ SoC

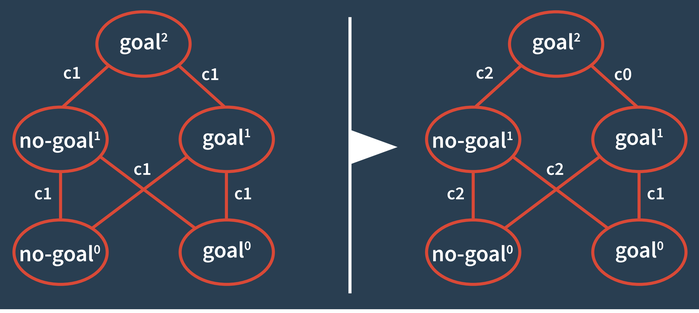
— Add binary "will-stay-on-goal-after-this-time" indicators and subtract the sum of these indicators from the objective
— **Optimality:** optimal since path length & NF cost are equal
— **Performance:** slow, as basic ILP is not tailored to this exact problem

$$cost(\mathcal{NF}) = 6 - sum(i) = 4 = SoC$$



SSP w/ SoC (CBMxSOC)

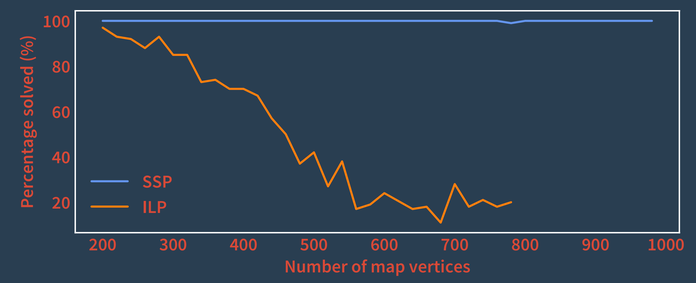
— Adjust edge costs to push an agent to a goal node as fast as possible
— **Optimality:** likely optimal, no counter-examples found in exhaustive testing
— **Performance:** decent, can be improved by actively reducing conflicts on the low-level (such as CAT / CBM do)



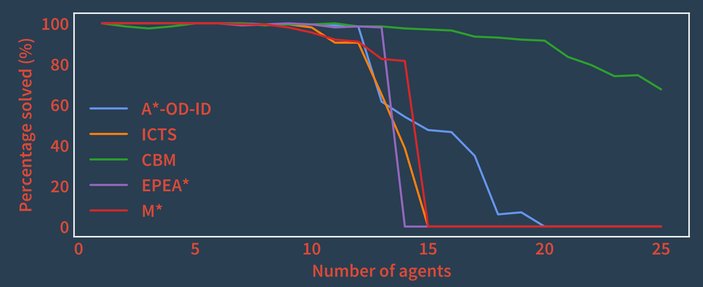
EXPERIMENTS

Experiments

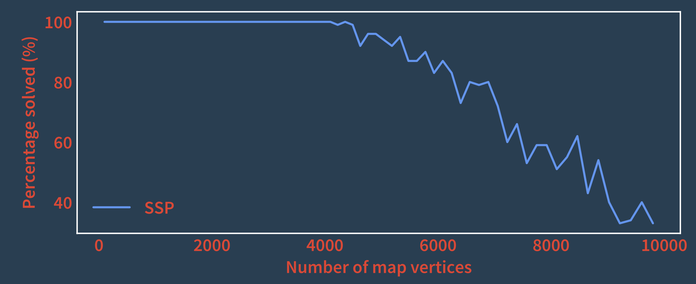
— SSP can handle bigger maps than ILP due to ILP model size limits



— CBMxSOC performs well against other algorithms due to non-exponential scaling regarding the number of agents



— On larger maps, the time-expanded network also becomes too large for the SSP solver



CONCLUSION

Conclusion

— The SSP technique is much faster than the ILP technique but it is complex to adapt to SoC optimally
— CBMxSOC performs very well with few conflicts and can be improved by adding conflict avoidance in the low-level solver
— Future work can look into MDD-SAT and disappearing agents

TU Delft