# AI HW2 - Our Heuristics

Omri Himelbrand & Peter Kogan

December 12, 2020

## 1 Closest savable people heuristic

This is our main heuristic.

The heuristic is used together with a utility function, which is defined per type of game, so it will fit all types of games, the heuristic returns a tuple of approximated scores for each agent.

The approximated scores are computed as followed:

- We go over all of the nodes with people to save at this state.
- For each such node, compute the distance for each agent to this node, using Dijkstra.
- Denote the distance for Agent1 as $d_1$ and Agent2 as $d_2$.
- $d_1 \leftarrow 2 \cdot d_1, \quad d_2 \leftarrow 2 \cdot d_2$, this is done since at each time step only one agent can move.
- Add 1 to the distance of the agent that it is not his turn at this state.
- The agent with the lower modified distance is then tested
- If the current time of the agent plus the modified distance is greater than the deadline then we add 0 to the approximated score.
- Else we add $PeopleAt(node) - d \cdot 0.001$ to the approximated score of the agent, this is done to break ties in case that we have more than one node with the same approximated score, basically it will prefer the shorter path to break the tie.
- After the iteration over all nodes with people to save we subtract the current time multiplied by 0.0001 from both the approximated scores, this is also done for tie breaking.
- return the approximated scores computed above

## 2 The utility functions

As mentioned above, we use the same heuristic for all games, but we use a different utility function for each type of game:

**Zero-Sum**: Just return the difference between the score of A and B, i.e. $Score_A - Score_B$. Note that this is done on the sum of the actual score, g, with the approximated score from the heuristic.

**Semi-cooperative**: this is done on the sum of g and h, and the value of the score of the current agent is then tested, if we have a tie, we break it by checking the other agent's score, choosing the maximal value.

**Fully-cooperative**: this is done on the sum of g and h, we just sum both scores together.

## 3 Example scenarios

We gave 2 example graphs, that for different types of games work differently.
**graph1.json** - need to use cutoff of 5, with the first agent starting at node 1, and the second at node 5, with deadline of 8.
**graph2.json** - need to use cutoff of 10, with the first agent starting at node 1, and the second at node 5, with deadline of 7.

Other values might work too, but these were tested and operate differently for different games.