The Thirteenth International Symposium
on Combinatorial Search (SoCS 2020)

Multi-Directional Search

Dor Atzmon,[1] Jiaoyang Li,[2] Ariel Felner,[1] Eliran Nachmani,[1]
Shahaf Shperberg,[1] Nathan Sturtevant,[3] Sven Koenig[2]
[1]Ben-Gurion University
[2]University of Southern California
[3]University of Alberta
dorat@post.bgu.ac.il, jiaoyanl@usc.edu, felner@bgu.ac.il, nachamni@post.bgu.ac.il,
shperbsh@post.bgu.ac.il, nathanst@ualberta.ca, skoenig@usc.edu

1 Problem Definition

In the *Multi-Agent Meeting* (MAM) (Yan, Zhao, and Ng 2015) problem we are given a weighted graph $G = (V, E)$ and a set of k start locations $S = \{s_1, \dots, s_k\}$ ($S \subseteq V$) for k agents $A = \{a_1, \dots, a_k\}$. The cost of edge $(v, v') \in E$ is denoted by $c(v, v') \geq 0$. A *solution* to MAM is a target location $t \in V$ indicating a meeting location for the agents, plus a set of paths from each s_i to t. An *optimal* solution (meeting location) t^* has the lowest cost among all solutions, and its cost is C^*. Let $d(v, u)$ be the cost of the shortest path from v to u. We consider two cost functions: *Sum-Of-Costs* (SOC) and *Makespan* (MKSP) that are defined as follows. SOC: $C_{SOC}(t) = \sum_{a_i \in A} d(s_i, t)$, and MKSP: $C_{MKSP}(t) = \max_{a_i \in A} d(s_i, t)$.

2 Multi-Directional MM (MM*)

MM* is a multi-directional best-first search algorithm that guarantees to return an optimal MAM solution for either SOC or MKSP. A node in MM* is a pair (a_i, v) representing an agent and its location. MM* organizes nodes in a single open-list (denoted OPEN) and a single closed-list (denoted CLOSED). OPEN is initialized with k root nodes: (a_i, s_i) representing each of the k agents and its start location. Each node is associated with a g-value. Naturally, $g(a_i, s_i) = 0$. Let $N(v)$ be the neighbours of v. Expanding a node (a_i, v) is composed of two actions: **(1)** Generating a node (a_i, v') for each $v' \in N(v)$, while setting $g(a_i, v') = g(a_i, v) + c(v, v')$ and inserting it to OPEN. **(2)** Moving (a_i, v) to CLOSED.

We say that v is a *possible goal* if it was generated from all directions. Its cost, $C(v)$, depends on the cost function. Let U be the cost of the *incumbent solution*, i.e., U is the minimum $C(v)$ among all possible goals (initially $U = \infty$). U is an upper bound on C^*. MM* halts when $fmin \geq U$, where $fmin$ is the minimum f-value in OPEN.

Consider node (a_i, v) in OPEN. $f^*_{SOC}(a_i, v)$ is the cost of the minimum solution such that: **(1)** a_i is forced to pass through v. **(2)** a_i continues from v to meet the other agents at some location t. **(3)** Each of the other agents a_j travels from s_j to t. Now, $f^*_{SOC}(a_i, v) = g(a_i, v) + h^*_{SOC}(a_i, v)$ where $h^*_{SOC}(a_i, v)$ is the sum of the minimal remaining cost for a_i that will complement the path that a_i has passed (with cost $g(a_i, v)$) getting it to t (item 2), plus the cost of the other agents to get from their start locations to t (item 3): $h^*_{SOC}(a_i, v) = \min_{t \in V}[d(v, t) + \sum_{a_j \in A \setminus \{a_i\}} d(s_j, t)]$.

Let $h_{SOC}(a_i, v)$ be an admissible heuristic, i.e., $h_{SOC}(a_i, v) \leq h^*_{SOC}(a_i, v)$. For SOC, naturally,

$$f_{SOC}(a_i, v) = g(a_i, v) + h_{SOC}(a_i, v). \tag{1}$$

The MKSP case is more complicated. Since in MKSP we take the maximum among agents (not the sum), we do not know which agent will be taken by the max operation. We begin by defining $f^*_{MKSP}(a_i, v)$, which is the best solution given that a_i is forced to pass through v:

$$f^*_{MKSP}(a_i, v) = \min_{t \in V}\left[\max\left\{\begin{matrix} g(a_i, v) + d(v, t) \\ \max_{a_j \in A \setminus \{a_i\}} d(s_j, t) \end{matrix}\right\}\right]. \tag{2}$$

For a given possible meeting location t we want the maximal path of one of the agents. If it is our current agent a_i then this is given by $g(a_i, v) + d(v, t)$ (top line of the max term). If it is some other agent a_j then it is given by $d(s_j, t)$ (bottom).

Next, we need to define f_{MKSP} as a lower bound on f^*_{MKSP}. Here, we do not define h^*_{MKSP} and h_{MKSP} but define $f_{MKSP}(a_i, v)$ in terms of $h_{SOC}(a_i, v)$ as follows:

$$f_{MKSP}(a_i, v) = \max\left\{g(a_i, v), \frac{g(a_i,v) + h_{SOC}(a_i, v)}{k}\right\}. \tag{3}$$

$g(a_i, v)$ is a lower bound because a_i has already traveled a path of cost $g(a_i, v)$ and $f^*_{MKSP}(a_i, v) \geq g(a_i, v)$. Observe that $\frac{f^*_{SOC}(a_i,v)}{k} \leq f^*_{MKSP}(a_i, v)$. This is because one of the agents must at least travel $\frac{f^*_{SOC}(a_i,v)}{k}$. Since $f_{SOC}(a_i, v)$ is a lower bound on $f^*_{SOC}(a_i, v)$ then dividing it by k will yield a lower bound on $f^*_{MKSP}(a_i, v)$.

Costs of subsets f^*_{MKSP} for k agents is determined by the longest path of one of the agents. Therefore, f^*_{MKSP} as well as f_{MKSP} for any subset of these k agents are also lower bounds on f^*_{MKSP} of *all* k agents. Thus, for any subset of $k' < k$ agents, we can compute f_{MKSP} and use it as a lower bound for f^*_{MKSP} for the entire set of k agents. In our experiments, we tried all combinations of pairs of agents.

3 Heuristics for MM*

We introduce a number of heuristics that are plugged directly in $f_{SOC}(a_i, v)$ and indirectly for f_{MKSP} (Equations 1 and 3). Let $t^*(a_i, v)$ be the optimal meeting location where a_i is forced to go through v. For simplicity, we use $\hat{t^*}$ to denote $t^*(a_i, v)$ and $h(a_i, v)$ to denote $h_{SOC}(a_i, v)$. Let $S_i(v)$ be a set of all start locations in S, except for s_i which is replaced with v ($S_i(v) = S \setminus \{s_i\} \cup \{v\}$). Thus, $h^*_{SOC}(a_i, v) = \sum_{v' \in S_i(v)} d(v', \hat{t^*})$.

h_1 : **Clique Heuristic** We assume that for every pair of locations (v_1, v_2) there exists a classic admissible heuristic h, such that $h(v_1, v_2) \leq d(v_1, v_2)$. Based on the triangle inequality, for every pair of locations $v_1, v_2 \in S_i(v)$ ($v_1 \neq v_2$) we have that: $d(v_1, v_2) \leq d(v_1, \hat{t^*}) + d(v_2, \hat{t^*})$. By summing all such pairs, we get: $\sum_{v_1,v_2 \in S_i(v)} d(v_1, v_2) \leq \sum_{v_1,v_2 \in S_i(v)} d(v_1, \hat{t^*}) + d(v_2, \hat{t^*})$. As each $v' \in S_i(v)$ exists in $k-1$ pairs, we can rewrite the right side of the equation as $(k-1) \cdot \sum_{v' \in S_i(v)} d(v', \hat{t^*})$. Since $h(v_1, v_2) \leq d(v_1, v_2)$ we get the *Clique heuristic*:

$$h_1(a_i, v) = \sum_{v_1,v_2 \in S_i(v)} \frac{h(v_1, v_2)}{k-1} \leq h^*(a_i, v) \qquad (4)$$

h_2 : **Median Heuristic** For a set of numbers $B \subset \mathbb{R}$, it is provable that the median of B minimizes the sum of the absolute deviations, i.e., $\operatorname{argmin}_{r \in \mathbb{R}} \sum_{b \in B} |b - r| = median\{B\}$. Let tm_d be the median number of dimension d. This creates a potential meeting location $tm = (tm_1, tm_2)$ that minimizes the sum of absolute deviations over two dimensions. Assume that the input graph $G = (V, E)$ is a 4-connected 2D grid where every location $v \in V$ can be represented by its coordinates $\vec{v} = (v_1, v_2)$. The L_1-distance for any two locations $u, v \in V$ is defined as $||\vec{u} - \vec{v}||_1 = |u_1 - v_1| + |u_2 - v_2|$ ($= \Delta x + \Delta y$). By modeling the problem in an empty 2D L_1-space (no obstacles), we introduce the *Median heuristic*:

$$h_2(a_i, v) = \sum_{v' \in S_i(v)} |v'_1 - tm_1| + |v'_2 - tm_2| \leq h^*(a_i, v) \qquad (5)$$

h_3 : **FastMap Heuristic** *FastMap* (Cohen et al. 2018; Li et al. 2019) is a near-linear preprocessing algorithm that embeds the locations of a given edge-weighted undirected connected graph $G = (V, E)$ into a D-dimensional L_1-space $\mathbb{R}^D$. Each location $v_i \in V$ is mapped to a D-dimensional point $\vec{p_i} \in \mathbb{R}^D$. The length of the shortest path $d(v_i, v_j)$ between any two locations $v_i, v_j \in V$ is approximated by the L_1-distance $||\vec{p_i} - \vec{p_j}||_1$ between the corresponding two points $\vec{p_i}, \vec{p_j} \in \mathbb{R}^D$ in this space. See (Cohen et al. 2018) for more details of FastMap. To compute h-values for MAM, h_3 applies the Median heuristic on the generated embedding $\mathbb{R}^D$. Let $\vec{p'} \in \mathbb{R}^D$ be the corresponding point of the embedding of location v' generated by FastMap. The *FastMap heuristic* is defined as:

$$h_3(a_i, v) = \min_{\vec{t} \in \mathbb{R}^D} \{ \sum_{v' \in S_i(v)} ||\vec{p'} - \vec{t}||_1 \} \leq h^*(a_i, v) \qquad (6)$$

	SOC				MKSP			
#Agents	3	5	7	9	3	5	7	9
h_0	6.87	18.98	29.46	44.17	1.91	6.73	11.03	18.07
h_1	**0.16**	4.66	16.15	36.29	0.47	2.62	3.57	6.53
h_2	**0.16**	**0.81**	**0.85**	**1.27**	**0.46**	**2.60**	**3.52**	**6.38**
h_3	1.92	8.50	18.66	33.20	1.48	6.57	9.59	16.46

Table 1: Average time on 500x500 grids with 10% obstacles

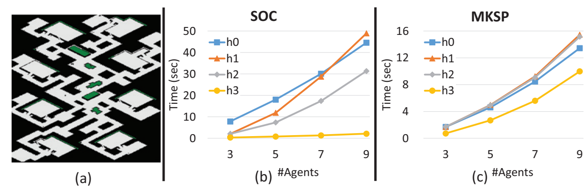

Figure 1: (a) Enigma map. (b) SOC time. (c) MKSP time.

4 Experimental Results

We compared all our new heuristics to the Dijkstra version of MM* ($h = 0$; denoted by h_0). For h_1, we used Manhattan Distance (MD) as a classic admissible heuristic between any two locations. The number of dimensions D for h_3 was always set to 10 as was suggested by Li et al. (2019).

We experimented on a 500x500 grid with 10% obstacles while varying the number of randomaly placed agents from 3 to 9. Table 1 shows the average time over 50 instances. For SOC, h_2 was the best as it is suitable for grids with small number of obstacles. h_3 incurred preprocessing time of ≈ 30 secs. For MKSP, h_2 was the best too, but here (unlike SOC), h_1 was very close to h_2. This is probably because the clique heuristic for MKSP also guides the agents to the median.

We also experimented on the Enigma map (768x768; Figure 1(a)) from the Starcraft video game (Sturtevant 2012). Figure 1(b) shows the average time of 50 instances for 3 up to 9 agents for minimizing SOC. Here, h_3 was the best. Since this map has many obstacles, h_2 and h_1 were less effective than h_3 which uses real distances. Nevertheless, h_3 required preprocessing time of 39s for this map (done once). Similarly, for MKSP (Figure 1(c)) h_3 was again the best.

In conclusion, for grids with few obstacles, h_2 is best. For domains with many obstacles, h_3 is best but requires preprocessing. h_1 is not far from both and it is applicable to all domains without the need of preprocessing.

References

Cohen, L.; Uras, T.; Jahangiri, S.; Arunasalam, A.; Koenig, S.; and Kumar, T. K. S. 2018. The FastMap algorithm for shortest path computations. In *IJCAI*, 1427–1433.

Li, J.; Felner, A.; Koenig, S.; and Kumar, T. S. 2019. Using fastmap to solve graph problems in a euclidean space. In *ICAPS*, 273–278.

Sturtevant, N. R. 2012. Benchmarks for grid-based pathfinding. *Computational Intelligence and AI in Games* 4(2):144–148.

Yan, D.; Zhao, Z.; and Ng, W. 2015. Efficient processing of optimal meeting point queries in euclidean space and road networks. *Knowledge and Information Systems* 42(2):319–351.