Predicting the difficulty of TSP instances using MST

Lahari Sengupta
School of Computing
University of Eastern Finland
Joensuu, Finland
lahari@cs.uef.fi

Pasi Fränti
School of Computing
University of Eastern Finland
Joensuu, Finland
franti@cs.uef.fi

***Abstract*— The efforts needed to solve travelling salesman problems (TSP) obviously depend on the problem size. However, also other factors can predict the difficulty of a given problem instance. We present a measure based on the minimum spanning tree (MST). The measure counts the number of knot points, which branch the tree into multiple sub-trees. We show by experiments that the more there are knots in the tree, the more difficult the problem instance is to solve by both humans and computers.**

Keywords—Travelling salesman problem, open loop TSP, MST, human performance, instance complexity.

I. INTRODUCTION

Solving travelling salesman problems (TSP) is challenging for computers and humans both. Computers can solve efficiently only small-size instances since the problem is NP-hard. Human problem-solving skills have also raised interest in the literature [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. Solving TSP instances by humans is like solving puzzle games but it also requires visual-spatial abilities [9, 10].

TSP instances appear also in an orienteering variant called rogaining [12], and in a mobile orienteering game called Mopsi orienteering, O-Mopsi [13]. The goal of the game is to visit a number of real-world targets, mainly in a city area. Unlike in the classical orienteering, there is no pre-defined order of visiting targets and the game ends immediately after all targets have been reached. Thus, the game playing implicitly includes the open loop travelling salesman problem.

O-Mopsi game instances are of small size, consisting of 4-27 targets. From the playing records, we found that only 18% of the players followed the optimum order while visiting targets. Therefore, regardless of the small size, these game instances can be difficult for human players. However, some problem instances are significantly more difficult to solve than other instances.

From the theory of algorithm, we know that the time complexity of finding the optimum solution is exponential with the number of targets. However, results from the literature have shown that the time taken by humans to solve the problem instances grows linearly or near-linearly with the problem size [1, 2].

The size is not the only factor that affects the difficulty of the problem instance. Three O-Mopsi game instances of Fig. 1 have almost the same number of targets. Despite this, the leftmost example (Otsola) is the easiest to solve because of the targets having an almost linear structure that can be followed from north to south. The middle one (Hukanhauta 3km) is slightly more difficult, and the rightmost (Christmas Star) is the most difficult for both humans and computers.

Figure 1: Examples of three TSP problem instances with difficulty level increasing from left to right.

An open question is how to estimate the difficulty of a given TSP problem instance. The number of targets is clearly one affecting factor but what other factors are there?

In literature, several measures have been considered to estimate the difficulty of the TSP problem instances for both the closed loop and open loop cases. Many research results [3, 4, 5, 6] reported that the more of the target points lay on the convex hull, the easier the problem is to solve by humans. Contradicting results have also been reported [7, 8]. One study also claims that human prefers to solve locally at first and then reaches to a global solution by crossing avoidance method [7].

The minimum spanning tree (MST) is another closely related graph problem. Its goal is to connect all the targets using a tree structure by minimizing the total distance of the selected connections. The MST is a significantly easier problem to solve than TSP. Kruskal [14] and Prim [15] with sophisticated data structures like union-heap require only about $O(N^2)$ time to solve; compared to the exponential $O(2^N)$ time required to find

the optimum solution for TSP. Human problem-solving performance on the TSP and MST also correlates highly, by a factor of 0.66 according to [9].

The solution for an open loop TSP is also a spanning tree, although not necessarily the minimum one. Fig. 2 shows two problem instances and the corresponding MST and TSP solutions. In the case of the first instance, both solutions have the same connections. In the case of the second instance, solutions still resemble each other but there are two connections in the TSP that have been replaced by shorter ones in the MST solution. Nevertheless, the relationship between the MST and TSP is obvious.

In this paper, we propose to use the MST solution for estimating the difficulty of a TSP problem instance. The idea is first to generate the minimum spanning tree and then count the number of knot points in the tree. The *knot* is a target that connects three or more points. Our hypothesis is that the more there are knots in the MST, the more difficult is the problem instance because knots are not allowed in the solution of a TSP.

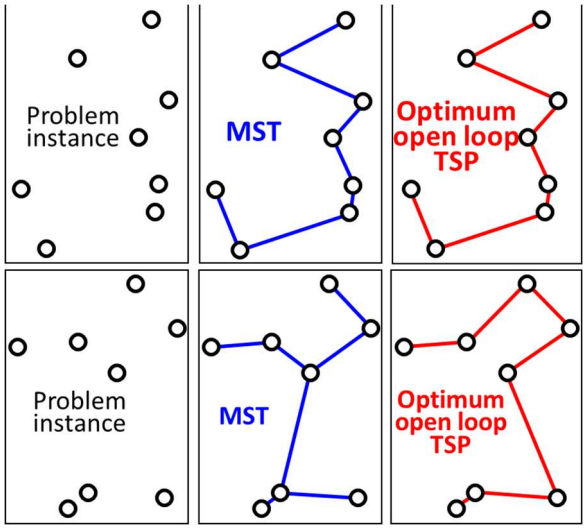

Figure 2: Examples of MST and open loop TSP solutions

The proposed measure is compared to both human and computer performance. We measure human performance in two ways. First, we study how close human can reach to the optimum when playing O-Mopsi games in the real world environment. Second, we study how fast human can find the optimum solution with computer simulations. In addition to these, we also test the existing hypotheses whether human problem solving has a linear or near-linear performance with the problem size.

Computer simulations are performed using the Concorde algorithm [16]. The Concorde algorithm was designed for the closed loop case; however, we modify it for the open loop case as explained in [17]. It runs much faster than the trivial brute force implementation. In specific, we want to find out whether its performance correlates to the problem size and our proposed MST knot measure or not.

II. Counting MST knots

The problem instance consists of N targets, which are the nodes of the graph. We define a knot as a node in the MST that has at least three links associated with it. Fig. 3 shows three examples with one, two and three knots, respectively.

Every knot divides the tree into sub-trees, called branches. Every branch makes the creation of TSP path more difficult. The number of knots varies from 0 to N/2-1. A special case is a tree without any knots. This minimum spanning tree is also the optimum solution for the open loop TSP because |MST|≤|TSP|. In general, we hypothesize that the more knots there are, the more difficult it is to solve.

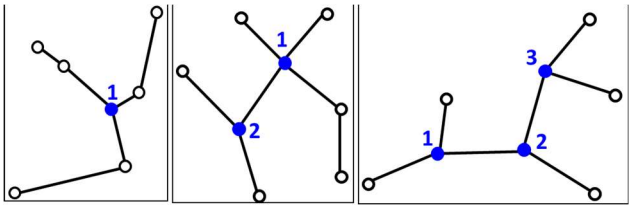

Figure 3: Examples of one, two, and three MST knots in problem sets

We measure the number of MST knots as following:

- We create MST by any algorithm
- While adding new links, we update the counts
- If a count exceeds 2, we mark the node as a knot
- Lastly, we count the total number of knots

Any algorithm such as Prim [15] and Kruskal [14] can create MST. Prim creates the tree iteratively by always adding the shortest link that connects a new target to the tree. It can be implemented in $O(N^2)$ time using the Fibonacci heap [18]. Kruskal maintains multiple trees, and at each step, adds the shortest link that merges two distinct trees.

A faster $O(N^{1.5})$ time divide-and-conquer algorithm [19] clusters the targets into SQRT(N) groups. It then solves the MST for each cluster separately using either Prim or Kruskal and then merges the sub-solutions by creating a meta-graph of the clusters. It achieves significantly faster solution at the cost of minor degradation of the accuracy of the MST.

Our problem instances are rather small and time is therefore not a bottleneck. We, therefore, use Prim's algorithm. Counting the links is trivially done during the tree construction, and the final measure is available immediately when the MST has been completed. We can use the count of the knots as such, or we can normalize it relative to the size of the problem instance (Fig. 4). In the first case, the measure takes into account the size of the problem, too.

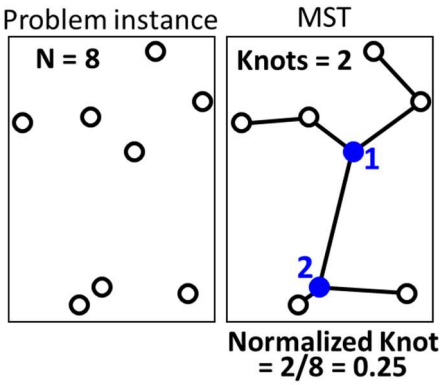

Figure 4: The number of MST knots and the normalized knot

The main benefits of the proposed measure are that it is simple to implement and that it can be calculated relatively fast. This is a significant improvement over the existing measures, of which many require to have optimum TSP solution as a reference [8, 9]. This is a huge restriction in real-time applications and limits the use of such measures only for the theoretical analysis, or to small problem instances.

III. Evaluation Methodology

To evaluate the proposed measure, we compare its results against human and computer performance. We consider the following three cases:

1. Mistakes by human
2. Time taken by human
3. Time taken by the computer algorithm

A. Human mistakes

We analyse the tours constructed by humans when playing O-Mopsi games in the real world environment. The game itself does not require players to find the optimum TSP solution but any path would work. However, since the performance is measured by the playing time, players naturally try to minimize the distance of their path, and therefore, aim at finding the optimum path.

For an easy game, finding the optimum path can be straightforward. However, with the increasing difficulty, it becomes harder to follow the optimum path during the game playing. Hence, they make mistakes and the final visiting order can differ from the optimum order. Consequently, the length of the path becomes longer.

To measure human performance we use the following three measures:

1. The number of mismatches
2. The number of mistakes
3. The gap to optimum

For the mismatch and mistake, we do not care about the length of the path for visiting the targets. Instead, we analyse the order in which targets were visited against the optimum path. The number of differences defines human performance.

Mismatch: We denote a link as a pair of two consequent targets in the path. The mismatch is defined as the number of links in the optimum tour that are missed in the player's path. An example is shown in Fig. 5 where the player's path is the same as the optimum with only two exceptions.

The problem of the mismatch is that it cumulates faults. Even a single different choice will cause another mismatch later in the path. In Fig. 5, the player reversed the optimum visiting order of the 3rd and 4th target. Therefore, the measure penalizes single mistake twice.

Figure 5: Player's tour has two mismatches in this game

Mistake: The idea is to count every mistake only once by updating the optimum tour dynamically after every choice. Otherwise, the measure works exactly as the mismatch: we count the number of links in the optimum tour that were missed by the player. After every mistake, we resolve a new optimum solution for the remaining unvisited nodes starting from the current node. Fig. 6 shows the same example as in Fig. 5 when the optimum path is updated after the mistake. It shows that the rest of the player's choices were optimally made.

Some mistakes are more significant than others are. For example, the mistake made in Fig. 6 causes only a minor increase in the path length while other mistakes can have more dramatic effects. Therefore, it can be meaningful to measure the effect of the choices rather than their count. Hence, we also consider the concept called a gap.

Gap: We measure the *gap* as the relative difference of the optimum path (TSP) to the path generated from the player's choices:

$$gap = \frac{|path| - |TSP|}{|TSP|} \quad (1)$$

We note that we do not use the real path travelled by the player. Instead, we generate artificial tour using the order of the targets chosen by the player. In this way, we eliminate the effect of navigating skills and measure merely the TSP problem-solving skill.

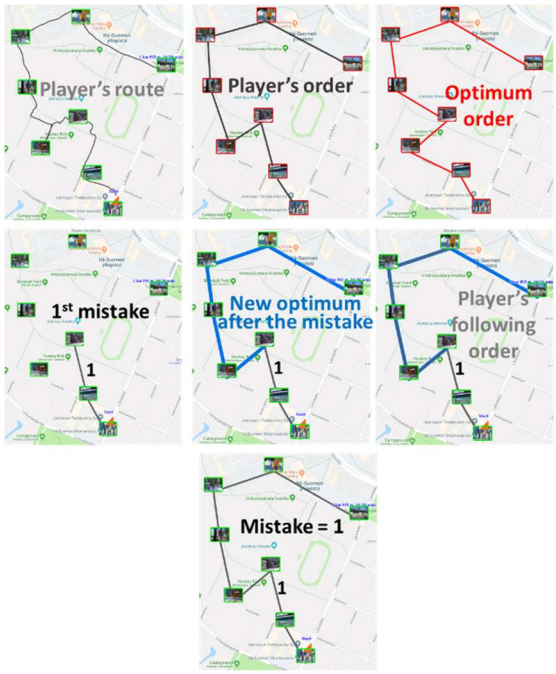

Figure 6: One mistake in this game

The distance of the path is measured using Euclidean distance between the targets. In some cases, the street network might provide a more accurate measurement. However, most games appear in parks and campus areas where using the street network would be too restrictive because players can easily make shortcuts. According to [17], Euclidean distance correlates slightly better (0.95) to the reality than routing via open street map (0.93).

Fig. 7 shows two examples of players' paths. There is only one mistake in the first example. However, the effect of that mistake is relatively large making the overall path 22% longer than the optimum path. On the contrary, even conceding two mistakes, the player managed to find a path, which is only 0.5% longer than the optimum in the second case.

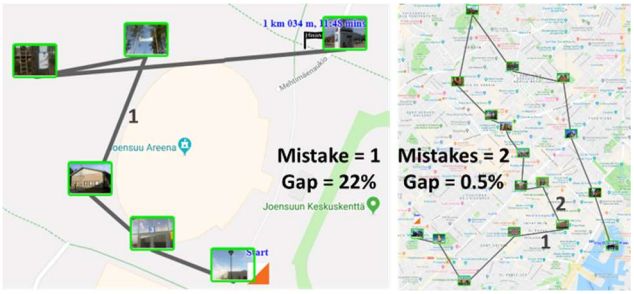

Figure 7: The number of mistakes does not always reveal the significance of the faults. The first example has only one mistake but the solution is rather poor (22% gap). The second example has two mistakes but the solution is still very close to the optimum (0.5% gap).

B. Human playing time

We evaluate human performance by measuring the duration of their playing to find the optimum solution. This can be significantly more challenging than just find a 'good' solution. For this, we presented problem instances to the players on the computer screen. Their task was to create paths using the easy-to-use web interface, which allows not only to create a new path but also to modify existing ones by mouse click and drag. We then measured the time it took for finding the optimum solution.

In the computer simulations, we showed the real-time difference between the player's current path and the optimum (gap). This guides the player towards finding the optimum solutions, which might be otherwise too hard to find for some problem instances. The optimum tours were generated by the local search algorithm in [20] as the instances were generated in real-time. Although it does not guarantee the optimality of the solution always, we later verified their optimality using the Concorde solver [16].

The problem instances can be found on the web[1], and their statistics are summarized in Table 1. The problem sizes vary from N=4 to 50, being 13 on average. The playing time varied hugely. From our analysis, we excluded cases when playing time exceeded 5 minutes.

TABLE I. DATASETS USED IN THIS STUDY

Dataset	Type	Distance	Instances	Sizes
O-Mopsi[a]	Open loop	Haversine	147	4-27
Dots[b]	Open loop	Euclidean	12125	4-50

[a] http://cs.uef.fi/o-mopsi/datasets/o-mopsi/

[b] http://cs.uef.fi/o-mopsi/datasets/dots/

C. Computer solving time

Our third test case is to measure the execution time for the computer algorithm to solve the instances. We use the Concorde algorithm, which is the fastest TSP solver for large problem instances [16]. In theory, running time should grow exponentially with the problem size. In practice, the performance depends also on other factors than the number of nodes. In specific, we expect that more difficult game instances can take considerably more time than easier instances with the same number of targets.

IV. EXPERIMENTAL RESULTS

We will test two hypotheses. The first hypothesis is based on the literature [1, 2] saying that human problem solving has a linear or near-linear performance with the problem size. The second hypothesis is that the number of MST knots can predict the difficulty of the instance.

We study three measures: problem size (N), the number of MST knots, and its normalized variants. Linear correlations of these three measures are summarized in Table II with the test scenarios presented in Section III.

Results show that all measures correlate very well with human mistakes (O-Mopsi) and Concorde time (O-Mopsi and Dots). There is also a slight correlation with the human problem-solving time (Dots), but only a weak correlation with the gap of the human solution. In other words, all measures can predict how

[1] http://cs.uef.fi/o-mopsi/datasets/

well human and computer perform on the problem instances, but not how close human players can reach the optimum solution.

We will test two hypotheses. The first hypothesis is based on the literature [1, 2] saying that human problem solving has a linear or near-linear performance with the problem size. The second hypothesis is that the number of MST knots can predict the difficulty of the instance.

We study three measures: problem size (N), the number of MST knots, and its normalized variants. Linear correlations of these three measures are summarized in Table II with the test scenarios presented in Section III.

Results show that all measures correlate very well with human mistakes (O-Mopsi) and Concorde time (O-Mopsi and Dots). There is also a slight correlation with the human problem-solving time (Dots), but only a weak correlation with the gap of the human solution. In other words, all measures can predict how well human and computer perform on the problem instances, but not how close human players can reach the optimum solution.

TABLE II. CORRELATION BETWEEN PROBLEM SIZE AND MST KNOT WITH GROUND TRUTHS

	O-Mopsi			Dots	
	Human mistake	**Human gap**	**Concorde time**	**Human time**	**Concorde time**
Problem size (N)	0.46	0.06	0.75	0.38	0.61
MST Knots	0.54	0.16	0.68	0.35	0.56
Normalized Knots	0.44	0.11	0.48	0.13	0.22

A. *Human mistakes*

Fig. 8 shows that with increasing problem size humans make more errors and take more time to solve the problem. Additionally, these figures show that human performance degrades linearly with the problem size, which confirms the conclusion made in [1] and [2].

The number of MST knots provides additional insight into the difficulty of the problem. Even it has stronger correlation (0.54) with human mistakes. Therefore, when the human mistake is the measure of human performance, both problem size and MST knots can predict the difficulty of instances.

B. *Human playing time*

Reference [1] and [2] both reported that the time needed by a human to solve a TSP is linearly proportional to the problem size. Our results with the Dots instances confirm that there is indeed linear correlation (0.38) with the average playing time. The regression line follows the function time = 1.4*N – 3.2.

However, there is a huge variation in the results, see Fig. 8 (middle). Again, in this case, MST knots can predict the difficulty of the instances as it has a linear correlation (0.35) to human time. Therefore, if the human time of finding an optimum solution is the measure of human performance, problem size and MST knots can define the difficulty of instances.

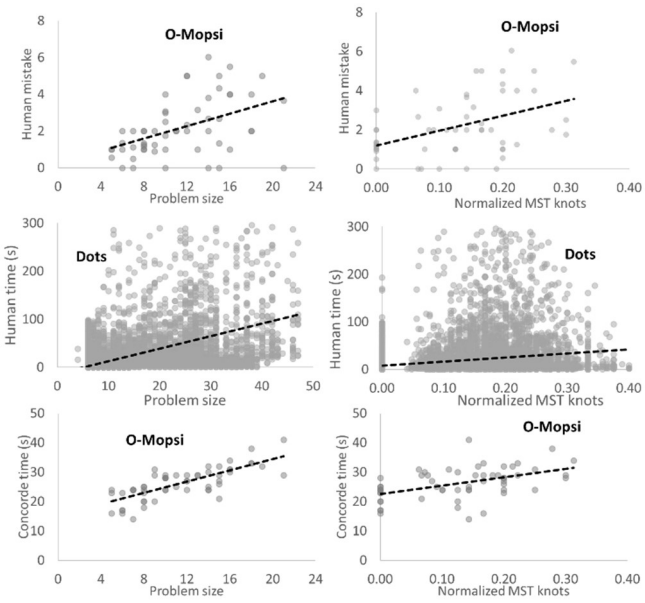

Figure 8: Human and algorithm performance with varying problem sizes and MST knots

In [1], human performance was also measured as the gap between the human solution and the optimum solution. It was reported that the gap grows very slowly with the problem size. Our results show that the gap has almost no correlation with the problem size and with the MST knots. With the problem sizes from N=10 to 15, the gap is almost constant. A linear correlation between MST knots and the gap can be observed only for the problem sizes between N=15 to 20, see Fig. 9.

C. *Computer solving time*

Concorde is an exact algorithm, and therefore, the problem size should correlate well with the processing times. This is indeed the case both with O-Mopsi (0.75) and Dots (0.61) problem instances, see Table II. The variation is much less than when compared against human performance, see Fig. 8 (bottom). The performance of the computer algorithm is, therefore, more predictable. The MST knots has also strong correlation both with O-Mopsi (0.68) and Dots (0.56) problem instances.

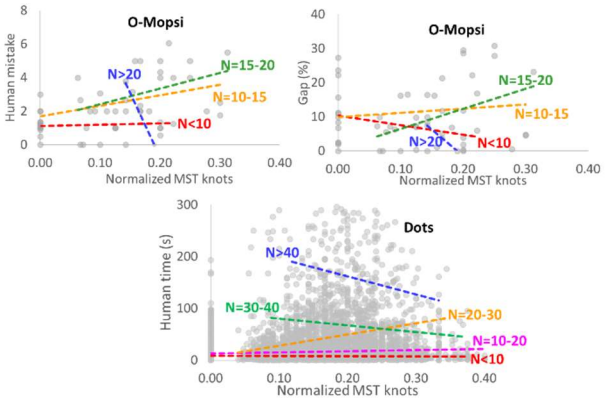

Figure 9: Human performance with varying MST knots for several ranges of problem sizes

D. Summary of results

The results in Table II showed that human mistake, human playing time, and algorithm time all correlate well with the problem size and the number of knots of the MST. The problem size alone predicts computer performance very well. In the case of human performance, none of the two parameters can predict all the behavior.

Skills of human is also a significant contributor. The O-Mopsi games are played in the real world where navigational skills of the player affect results. In the case of large instances (N>20), human performance also improves with an increasing number of knots. This might be because more skillful players played the larger games.

To sum up, in addition to the problem size, the number of knots in MST provides an important clue about the difficulty of the problem. However, the player skills should also be taken into account when interpreting the results.

V. CONCLUSIONS

We have shown that counting MST knots can predict the difficulty of TSP problem instances. It correlates very well with the time both humans and computer algorithm take to find the optimum solution, and the number of mistakes humans make while playing O-Mopsi games in the real world.

The results also showed that human mistakes and computer solving time have a linear relationship with the small problem sizes. However, when human performance is measured by the gap of the human and optimum solutions, neither the problem size nor the MST knots can predict the difficulty of the problem instances.

REFERENCES

[1] S. M. Graham, A. Joshi, and Z. Pizlo, "The traveling salesman problem: A hierarchical model," *Memory and Cognition*, 28 (7), 1191-1204, 2000.

[2] M. J. Dry, M. D. Lee, D. Vickers, and P. Hughes, "Human performance on visually presented traveling salesperson problems with varying numbers of nodes," *Journal of Problem Solving*, 1, 20-32, 2006.

[3] J. N. Macgregor and T. C. Ormerod, "Human performance on the traveling salesman problem," *Perception and Psychophysics* 58: 527, 1996.

[4] J. N. Macgregor, T. C. Ormerod, and E. P. Chronicle, "Spatial and contextual factors in human performance on the travelling salesperson problem," *Perception*, 28, 1417-1427, 1999.

[5] J. N. Macgregor, T. C. Ormerod, E. P. Chronicle, "A model of human performance on the traveling salesperson problem," *Memory and Cognition*, 28(7), 1183-1190, 2000.

[6] J. N. Macgregor, E. P. Chronicle, T. C. Ormerod, " Convex hull or crossing avoidance? Solution heuristics in the travelling salesperson problem," *Memory & Cognition*, 32(4), 260-270, 2004.

[7] D. Vickers, M. D. Lee, M. Dry, and P. Hughes, "The roles of the convex hull and the number of potential intersections in performance on visually presented traveling salesperson problems," *Memory and Cognition*, 31 (7), 1094-1104, 2003.

[8] M. J. Dry, and E. L. Fontaine, "Fast and Efficient Discrimination of Traveling Salesperson Problem Stimulus Difficulty," *The Journal of Problem Solving*: 7 (1), Article 9, 2014.

[9] D. Vickers, T. Mayo, M, Heitmann, M. D. Lee, and P. Hughes, "Intelligence and individual differences on three types of visually presented optimisation problems," *Personality and Individual Differences*, 36, 1059-1071, 2004.

[10] M. J. Dry, K. Preiss, and J. Wagemans, "Clustering, Randomness, and Regularity: Spatial Distributions and Human Performance on the Traveling Salesperson Problem and Minimum Spanning Tree Problem," *The Journal of Problem Solving*, 4 (1), Article 2, 2012.

[11] Y. Haxhimusa, W. G. Kropatsch, Z. Pizlo, A. Ion, A. Lehrbaum "Approximating TSP Solution by MST Based Graph Pyramid," *Graph-Based Representations in Pattern Recognition*. GbRPR 2007. Lecture Notes in Computer Science, vol 4538, 2007.

[12] G. N. Phillips and R. Phillips, "Rogaining: cross-country navigation," *Outdoor Recreation in Australia*, Perth, W.A, 1982.

[13] P. Fränti, R. Mariescu-Istodor, and L. Sengupta, "O-Mopsi: Mobile Orienteering Game for Sightseeing, Exercising, and Education," *ACM Trans. Multimedia Comput. Commun. Appl.* 13 (4), 56:1-12, 2017.

[14] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the Americal Mathematical Society*, 7, 48-50, 1956.

[15] R. C. Prim, "Shortest connection networks and some generalizations," *The Bell System Technical Journal*, 36 (6), 1957.

[16] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, "*The traveling salesman problem: a computational study*," Princeton university press, 2011.

[17] L. Sengupta, R. Mariescu-Istodor, and P. Fränti, "Planning Your Route: Where to Start?," *Computational Brain & Behavior*, 1 (3-4), 252-265, 2018.

[18] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, "*Introduction to Algorithms*," second ed., The MIT Press, 2001.

[19] C. Zhong, M.I. Malinen, D. Miao and P. Fränti, "A fast minimum spanning tree algorithm based on K-means," *Information Sciences*, 295, 1-17, 2015.

[20] L. Sengupta, R. Mariescu-Istodor, and P. Fränti, "Which local search operator works best for open loop Euclidean TSP?", *manuscript* (submitted).