Algorithms for the Moving Target Travelling Salesmen Problem

[Bachelor thesis]

Felix Greuling
Universität zu Lübeck
Ratzeburger Allee 160
23562 Lübeck, Deutschland
Felix.Greuling@student.uni-luebeck.de

ABSTRACT

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

1. INTRODUCTION

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

2. DEFINITIONS

In MT-TSP we consider an amount of targets $T = \{t_1, ..., t_n\}$ and a set of velocities $V = \{v_1, ..., v_n\}$ so that each moving with a constant movement speed $\overrightarrow{v_i}$. A pursuer starts at the origin O (a defined position), moving with a velocity of v_p. His aim is to visit all targets once and finishes with returning to the origin.

Therefore, we can model this problem as the following graph $G = (T, V, O, v_p)$.

3. INSTANCES OF MOVING-TARGET TSP

It was proven that MT-TSP is NP-hard. Some instances can result in an unbounded error, whenever the pursuer choses a non-optimal tour. Therefore, the condition $v_p > |\overrightarrow{v_i}|$ must apply, to avoid these errors. This was proven by the authors in [3], since the goal is the most fast optimal tour. Thus, instead of directly calculating the tour of the pursuer, it is necessary to determine the solvability of the input. Whenever it is not possible we gain a 'No'-instance, otherwise we go ahead and calculate the tour.
This paper presents two concrete cases:

1) 1D-case: Each target's movement is limited to a single line
2) 2D-case: The movement direction of a target consists of a two-dimensional vector

Each case is investigated for the shortest and the fastest tour. Helvig , Robins and Zelikovsky showed in [3] specific algorithms for 3 concrete cases: $\mathcal{O}(n^2)$-time algorithm in 1D-case, $(1+\alpha)$-approximation algorithm when the number of moving targets is small and $\mathcal{O}(n^2)$-time algorithm whenever enough targets are stationary. However, determining the approximation for general cases is hard, because there are many influences that determine the complexity of the problem. The approximation research in [2] showed that MT-TSP cannot be approximated better than by a factor of $2^{\pi(\sqrt{n})}$ by a polynomial time algorithm unless P=NP.

Later on, we will compare new heuristics and algorithms with those that recently proposed in [1]. This paper examines Genetic Algorithm, Simulated Annealing and Ant Colony Optimization to solve MT-TSP.

3.1 One-dimensional-case

This specific case is already introduced extensive in [3]. Each movement is fixed on a single line, thus, there are two possible directions for a target and the pursuer.

3.2 Two-dimensional-case

In this case another dimension is added. We will consider different scenarios with fixed numbers of targets and corridors. Compared to the 1D-case, in which a target is fixed on a line, targets freely move within the space. Therefore, we define corridors, representing the number of vectors in which the targets move, resulting in different complexities.

This paper focusses on the 2D-case. Previous works did not construct concrete heuristics for this type of case, therefore, this paper shows new aspects in the field of MT-TSP.

3.3 Input & Output

To set up heuristics a suitable input and output are necessary. Thus, the MT-TSP can be modelled as graph problem (refering section 2), further heuristics just expect a graph G as parameter.

The Output, on the other hand, can look very different. The most obvious parameter are the length and the time needed to finish the tour. Whenever, there is no possible tour, an MT-TSP algorithm needs to return a 'No'-instance.

Furthermore, a tour should be comprehensible. Therefore, the targets are displayed in the order in that the pursuer executes the tour. For each target the initial coordinates and the time and the position whenever the pursuer intercepts the target are monitored. To fully understand and analyse the tour a visualization is a good application, but not necessary.

4. ALGORITHMS FOR MT-TSP

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

4.1 Algorithms for 1D-case

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

4.1.1 Algorithm by Helvig, Robins and Zelikovsky

As previously mentioned, the authors of [3] showed an exact $\mathcal{O}(n^2)$-time algorithm for 1D-cases, which is based on dynamic programming. The goal is to receive the fastest, optimal tour, intercepting each target.

The algorithm is based on the idea, that the pursuer cannot change its direction until it intercepts the fastest target ahead of him. Therefore, only targets, where the pursuer changes direction, need to be considered. Thus, the tour is a sequence of snapshots with turning points and a snapshot can be denoted as a state A. Each tour always starts in A_0 and finishes in A_{end}. A state $A_i = (s_k, s_f)$ consists of a target which just intercepted (s_k) and the fastest target on the other side of the origin (s_f). Ensuing a state, there are at most two transitions to other states, the fastest on the left or the right of the pursuer. The time of a transition τ is determined with $t(\tau)$.

The algorithm begins with dividing all targets into two lists, according to whether a target is on the left or right side of the origin. Afterwards, both lists are sorted in descending order of their speed. The algorithm executes at each state with one of the following steps:

- if the state has no transitions into it, we proceed to the next state in the list
- if the pursuer has intercepted all of the targets on one side, we make a transition into the final state A_{final}
- we make two transitions which correspond to sending the pursuer after either the fastest target on the left or the fastest target on the right

To denote the time used to reach a state B_i with the shortest sequence of states, the time-function t is used with $t(A_i) = min\{t(A)+t(\tau)|\tau : A \to B\}$. As usual in dynamic programming, we can traverse backwards from A_{final} to A_0, which denotes the reversed list of turning points that describes an optimal tour.

4.2 Algorithms for 2D-case

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua

4.3 Algorithms for general cases

Most of previous works refer to general cases without specific consideration of 1D- and 2D-cases. In the following subsection innovative approaches are shown, just developed in 2019, mentionned in [1].

4.3.1 Algorithm by de Moraes and de Freitas

In [1] the authors proposed to solve MT-TSP with colony-algorithms. This includes the Genetic Algorithm (GA), Simulated Annealing (SA) and Ant Colony Optimization (ACO), which try to evolve a population to an optimum.

GA evolve candidate solutions until a certain fitness or a number of iterations is reached. The operands selection, crossover and mutation take care of generating new variations within the population.

5. SUMMARY AND OUTLOOK

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

6. REFERENCES

[1] Rodrigo S de Moraes and Edison P de Freitas. Experimental analysis of heuristic solutions for the moving target traveling salesman problem applied to a moving targets monitoring system. *Expert Systems with Applications*, 2019.

[2] Mikael Hammar and Bengt J Nilsson. Approximation results for kinetic variants of tsp. In *International Colloquium on Automata, Languages, and Programming*, pages 392–401. Springer, 1999.

[3] Christopher S Helvig, Gabriel Robins, and Alex Zelikovsky. The moving-target traveling salesman problem. *Journal of Algorithms*, 49(1):153–174, 2003.