

Moving-Target TSP in two-orthogonal-axes

Pseudocode

Felix Greuling (666020)

21. Oktober 2019

Algorithm 1 Exact Algorithm for two-orthogonal-axes Moving-Target TSP

Input: The initial positions and velocities of n targets, and the maximum pursuer speed

Output: A time-optimal tour intercepting all targets, and returning back to the origin

Preprocessing

Use Preprocessing to eliminate targets to optimize the tour

Partition the list of targets into the targets on the left side, the right side, the top side and the bottom side of the origin

if all remaining targets in $PrioQ$ are located on one side of the intersection,
once the pursuer reached the intersection from *current* **then**

 Calculate the time required to intercept all remaining targets (and return
 to the origin), sort them in order of nondecreasing interception times and
 add all to *OUTPUT*

 Go to the *Postprocessing* step

end if

Sort the targets on the left into list *Left* in order of nonincreasing speeds

Sort the targets on the right into list *Right* in order of nonincreasing speeds

Sort the targets on the top into list *Top* in order of nonincreasing speeds

Sort the targets on the bottom into list *Bottom* in order of nonincreasing speeds

Delete targets in *Left*, *Right*, *Top* and *Bottom* which are closer to the origin than faster targets in this list. Insert those deleted targets in the list *Eliminated*. Don't remove targets which move towards the other direction so they are crossing the origin

Rejoin *Left*, *Right*, *Top* and *Bottom* for the *Main Algorithm*

Main Algorithm

Let *time* be the time-array in which a target t_i is intercepted

Let *current* be the target the pursuer just intercepted

Let *OUTPUT* be the list of intercepted targets

current \leftarrow origin (initial position of the pursuer)

OUTPUT.add(current)

PrioQ is a priorityqueue which sorts the targets in order of nonincreasing priority

for each $t_i \in T$ **do**

$time[t_i] \leftarrow \infty$

PrioQ.add(t_i)

end for

while *PrioQ* is not empty **do**

if all remaining targets in *PrioQ* are located on one side of the intersection,
 once the pursuer reached the intersection from *current* **then**

 Calculate the time required to intercept all remaining targets (and return
 to the origin), sort them in order of nondecreasing interception times and
 add all to *OUTPUT*

 Go to the *Postprocessing* step

end if

 Calculate priority of $t_i \in PrioQ$ **TODO: Find a suitable priority measure**

 Update *PrioQ*

$prev \leftarrow current$

$current \leftarrow prioQ.poll()$

$time[current] \leftarrow time[prev] + \pi[prev \rightarrow current]$

 Update the position of each $t_i \in PrioQ$

Intercepted \leftarrow intercepted targets between *prev* and *current* (also add *current*)

 Sort *Intercepted* in order of nondecreasing interception times and add all to *OUTPUT*

end while

Postprocessing

for each $t_i \in T$ **do**

if a target $e_j \in \textit{Eliminated}$ is intercepted between t_i and t_{i+1} **then**

 Calculate the interception time of e_j and add e_j at the correct position in *OUTPUT*

 Remove e_j from *Eliminated*

end if

end for

return *OUTPUT* with the respective interception time
