Moving-Target TSP in two-orthogonal-axes

Pseudocode

Felix Greuling (666020)

05. 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
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
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. Don't remove targets which move towards the other direction so they are crossing the origin.
Add an empty Target to *Left, Right, Top and Bottom* to the end of each list

if 3 of the 4 lists are empty **then**
    Calculate the time required to intercept all remaining targets; and
    Go to the postprocessing step
end if

Main Algorithm
Let A_0 be the start state
Let A_{final} be the final state
STATE is the sorted list of states in order of nondecreasing sum of the indices of each state's targets in lists *Left, Right, Top and Bottom.* Leave out the case with 4 empty targets
Place A_0 first in the list STATE
Place A_{final} last in the list STATE
$t(A) \leftarrow \infty$ for any state $A \neq A_0$
$t(A_0) \leftarrow 0$
$current \leftarrow 0$

while current $\leq$ the number of states in $STATE$ **do**
  $A = STATE[current]$
  if there are no transitions into A **then**
    Increment *current* and jump back to the beginning of the while loop
  end if
  if for state A, all remaining targets are on one side of the origin (A consists of exactly three empty targets) **then**
  $t(\tau_{final}) \leftarrow$ time required to intercept the remaining targets (and return to the origin)
else
  Calculate the four transitions $\tau_{left}, \tau_{right}, \tau_{top}, \tau_{bottom}$ from state A using lists *Left, Right, Top, Bottom*
  if $t(A) + t(\tau_{left}) < t(A_{left})$ **then**
    $t(A_{left}) \leftarrow t(A) + t(\tau_{left})$
  end if
  if $t(A) + t(\tau_{right}) < t(A_{right})$ **then**
    $t(A_{right}) \leftarrow t(A) + t(\tau_{right})$
  end if
  if $t(A) + t(\tau_{top}) < t(A_{top})$ **then**
    $t(A_{top}) \leftarrow t(A) + t(\tau_{top})$
  end if
  if $t(A) + t(\tau_{bottom}) < t(A_{bottom})$ **then**
    $t(A_{bottom}) \leftarrow t(A) + t(\tau_{bottom})$
  end if
end if
Increment *current*
end while
$OUTPUT \leftarrow$ the reverse list of states from A_{final} back to A_0

Postprocessing
for pair of consecutive states in $OUTPUT$ **do**
    Calculate which targets are intercepted between the state pair
    Sort the intercepted targets by the interception order
end for
Output the concatenated sorted lists of targets