

Moving-Target TSP in two-orthogonal-axes

Pseudocode

Felix Greuling (666020)

21. Oktober 2019

Algorithmus 1 Algorithmus für zwei-orthogonale Achsen beim bewegendem Ziele in TSP

Input: Ziele T , Ursprung $z_{ursprung}$, Verfolgergeschwindigkeit v_{max}

Output: Ziele T in der Tour-Reihenfolge, inklusive Retour zum Ursprung

Sei t das Zeit-Array, welches für jedes $z_i \in Z$ die Abfangzeit angibt

Sei $current$ das Ziel, welches der Verfolger soeben eingeholt hat

Sei $OUTPUT$ die Liste an Zielen in der Abfangreihenfolge

$current \rightarrow origin$

$OUTPUT.add(current)$

for $z_i \in Z$ **do**

$t[z_i] \rightarrow \infty$

$Q.add(z_i)$

 Berechne $\alpha(z_i)$

end for

while Q is not empty **do**

if jedes verbleibende Ziel liegt auf einer der vier Seiten des Schnittpunktes **then**

for $z_i \in Q$ **do**

$t[z_i] \rightarrow$ Zeit von der aktuellen Position bis zum Einholen von z_i

end for

 Sortiere Q in aufsteigender Reihenfolge nach $t[z_i]$

 Berechne Rückkehr zum Ursprung ausgehend vom letzten Ziel aus Q

$OUTPUT.addAll(Q)$

 break

end if

 Berechne $\alpha(z_i)$, $\forall z_i \in Q$

 Update Q

$prev \rightarrow current$

$current \rightarrow Q.poll()$

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

$OUTPUT.add(current)$

 Update die Position von jedem $z_i \in Q$

$EingeholteZiele \rightarrow$ Ziele zwischen $prev$ und $current$

 Sortiere $EingeholteZiele$ in aufsteigender Reihenfolge nach $t[z_i]$

$OUTPUT.addAll(EingeholteZiele)$

end while

$OUTPUT.add(z_{ursprung})$
