This trip scheduling algorithm is designed based on the classic dynamic programming solution to the weighted interval scheduling problem. This document shall describe the algorithm procedure, the time/space complexity, and the correctness of the algorithm.

## Notations

$L$ - the set of all loads. $l, n$ - $l$ is a list of loads. $n$ is the size of the list.
$l_i$ - the $i$-th load of $l$.
$v_i$ - the amount of profit of $l$.
$Start_i$ - the start location of the $i$-th load.
$Dest_i$ - the destination of the $i$-th load.
$Cost(a, b)$ - the fuel cost for going from one of $a, b$ to the other.
$Time(a, b)$ - the time needed for going from one of $a, b$ to the other.
$s_i$ - the start time of the $i$-th load.
$f_i$ - the finish time of the $i$-th load ($f_i = s_i + Time(Start_i, Dest_i)$).
$Pre_i$ - the index of the load in $l$ that has the latest arrival time that can be taken before $l_i$.
$Opt_i$ - the highest possible net profit of all trips formed from the first $i$ loads of $l$.
$Last_i$ - the last load taken in the trip with net profit $Opt_i$.
$BEGIN$ - the start location of the trip, given by the query.
$t_{begin}, t_{end}$ - the queried earliest beginning time and latest destination time for the trip.

## Procedure

1. From $L$, take the list $l$ of all available loads (i.e., list of all loads such that for any $l_i \in l$, $t_{begin} + Time(BEGIN, Start_i) \le S_i$, and $F_i \le t_{end}$). Denote the size of $l$ as $n$. Time complexity: $\mathcal{O}(|L|)$.

2. Sort $l$ by finish time $f_i$ in ascending order (earliest first). Time complexity: $\mathcal{O}(n \log(n))$.

3. Calculate each $Pre_i$.
   Search from $i-1$ to 1, if $f_k + Time(Dest_k, Start_i) <= s_i$, then $Pre_i = k$. Worst case time complexity $\mathcal{O}(|l|^2)$, but in most cases $\mathcal{O}(n)$

4. Fill in $Opt_i$ and $Last_i$. $0 \le i \le n$, $Opt_0 = 0$, $Last_0 =$ `null`.
   If $Pre_i = -1$ or $Last_{Pre_i} =$ `null`, let

   $$CurProfit := v_i - Cost(BEGIN,\ Start_i) - Cost(Start_i,\ Dest_i),$$

   otherwise, let

   $$CurProfit := v_i - Cost(Dest_{Pre_i},\ Start_i) - Cost(Start_i,\ Dest_i) + Opt_{Pre_i}.$$

   Then, $Opt_i = \max(CurProfit,\ Opt_{i-1})$, and $Last_i = CurProfit > Opt_{i-1}\ ?\ l_i : Last_{i-1}$.
   Time complexity: $\mathcal{O}(n)$.

Total time complexity $= \mathcal{O}(|L| + n\log(n) + n^2 + n) = \mathcal{O}(|L| + n^2)$

## Correctness

Consider the optimal trip $T = [l_{i_1}, \ldots, l_{i_k}]$ of an initial segment $l_1, \ldots, l_j$ of all loads ($k \leq j, j \leq n$). Obviously, a load $l_i$ is in $T$ if and only if

$$v_i - Cost(Dest_{Pre_i},\ Start_i) - Cost(Start_i,\ Dest_i) + Opt_{Pre_i} \geq Opt_{i-1},$$

that is, the total profit after taking the load is at least as good as the total profit of the best trip if the load cannot be taken.

In addition, step 4 correctly computes each $Opt_i$. This can be proven by induction.

*Proof.* By definition, $Opt_0 = 0$. Consider $i > 0$. In the best trip $T$ from loads $l_1$ to $l_i$, either $l_i$ is in $T$, or $l_i$ is not. If $l_i$ is in $T$, it is the last load (since $l$ is sorted), and if there is a second last load, it is $l_{Pre_i}$, and trucker will go from $Dest_{Pre_i}$ to $Start_i$, so the net profit of $T$ is

$$v_i - Cost(Dest_{Pre_i},\ Start_i) - Cost(Start_i,\ Dest_i) + Opt_{Pre_i},$$

or there is no second last load, that is, the trucker need to go from $BEGIN$ to $Start_i$, and so the net profit is

$$v_i - Cost(BEGIN,\ Start_i) - Cost(Start_i,\ Dest_i).$$

Of course, in either case, the net profit must be greater than or equal to $Opt_{i-1}$. If $l_i$ is not in $T$, then $Opt_i = Opt_{i-1}$. □