**Input**(n, e, l, d, t)

Procedure **Start\_path(a):**

*greedy* = {}

**for** i ← 0 … (n – 1) **do**

*greedy* ← *greedy* + (*a[i]*)

**sorted**(*greedy*)

*visited* = {}

**for** i ← 0 … (n – 1) **do**

*visited* ← *visited* + FALSE

*path* = []

for i ← 0 … (n – 1) do

for j ← 0 … (n – 1) do

if *greedy[i]* = *a[j]* and *visited[j]* = FALSE do

*path* ← *path* + j

*visited[j]* = TRUE

Procedure **Generating\_starting\_path:**

*startpath1* = start\_path(*e*)

*startpath2* = start\_path(*l*)

*tabu* = queue{(*1, startpath1*), (*1, startpath2*)}

Procedure **Calculate(path)**:

*bool check* = TRUE

*cur\_pos* = 0

*total\_time* = 0

*travel\_time* = 0

**for** i ← 0 … (n – 1) **do**

*total\_time* = *total\_time* + *t*[*cur\_pos*][*path[i]*]

travel\_time = travel\_time + t[cur\_pos][path[i]]

**if** *total\_time* ≤ *l*[*path[i]*] **then**

*total\_time* = max(*total\_time*, *e*[*path[i]*])

*total\_time* = *total\_time* + *d*[*path[i]]*

*cur\_pos* = *path[i]*

**else**

*check* = FALSE

**endif**

**if** *check* = TRUE **then**

**return** *travel\_time*

**else**

**return** ∞

**endif**

Procedure **Tabu\_Search(***runtime***):**

*list\_satisfy\_path* = {*startpath1, startpath2*}

*value\_path1* = Calculate(*startpath1*)

*value\_path2* = Calculate(*startpath2*)

cur\_MIN = ∞

*optimal\_path* = ∅

*depth* = 1 **if** *n* > 200 **else** min(200 / *n*, 3)

*vertex\_check* = *n* / 4 × *depth*

while *runtime* ≤ **runtime** do

*element* = **pop**(*tabu*)

**if** *element[0]* ≤ *depth* **then**

*cur\_best\_element* = {}

*found\_better* = FALSE

*try\_path* = *element[1]*

**for** i ← 0 … (n – 1) **do**

**for** j ← i … min(i + *vertex\_check*, *n*) **do**

**swap**(*try\_path[i]*, *try\_path[j]*)

*travel\_time* = Calculate(*try\_path*)

**if** *travel\_time* < *cur\_Min* **then**

*found\_better* = TRUE

*cur\_best\_element* = {}

*cur\_best\_element* ← *cur\_best\_element* + (0, *try\_path*)

*cur\_MIN* = *travel\_time*

*optimal\_path* = *try\_path*

**else if** *travel\_time* = *cur\_MIN* **then**

*cur\_best\_element* ← *cur\_best\_element* + (0, *try\_path*)

**else**

*cur\_best\_element* ← *cur\_best\_element* + (*element[0]* + 1, *try\_path*)

**endif**

**swap**(*try\_path[i]*, *try\_path[j]*)

**if** *found\_better* = TRUE and *cur\_best\_element* ≠ ∅**then**

*tabu* = {}

**for** all element in *cur\_best\_element* **do**

*tabu* ← *tabu* + *element*

**endif**

Procedure **Print():**

*cur\_OPTIMAL\_path* = NULL

**for** all element in *list\_satisfy\_path* **do**

*travel\_time* = Calculate(*try\_path*)

**if** *element[0]* = 0 and *travel\_time* ≤ *cur\_MIN* **then**

*cur\_MIN* = *travel\_time*

*cur\_OPTIMAL\_path* = *element[1]*

**endif**

**print**(*cur\_OPTIMAL\_path*)