Supplemental Online Material

The pseudo code of the Travel Path Distance calculation between TF_1 and TF_2

```
Start Travel_Path_Distance

Input a set of ARCs

Input TF1 and TF2

***Step 1***

For each coordinate c in ARCs

        Find c, which gives the minimum Euclidean distance between c and TF1

        Set c as SP; and the minimum distance as L1

        Find c, which gives the minimum Euclidean distance between c and TF2

        Set c as FP; and the minimum distance as L3

Next coordinate c

***Step 2***

CP = SP

Do While CP <> FP

**2.1** Set Array NP(1 to 4)

        For n = 1 to 4

        For each coordinate c in ARCs

                If the Euclidean distance between c and CP = 1 Then

                        NP(n) = c

                        n = n + 1

                End If

        Next coordinate c

        Next n

**2.2** For n = 1 to 4
```

If NP(n) <> PP Then

Find NP(n) which gives the minimum Euclidean distance between NP(n) and FP

Set n as *min1n*

Find NP(n) which gives the second minimum Euclidean distance between NP(n) and FP

Set n as *min2n*

End If

If this CP is a Turning Point Then

TP = CP

PTP = PP

End If

Next n

2.3 If *outward* >= *maxoutward* Then this move is not allowed so select a fix method

Method 1 Switch the Start and Finish Points

CP = FP; FP = SP; SP = CP; and recalculate L_2 from the beginning

Method 2 Return to the Turning Point and take a new turn

CP = TP; PP = PTP; and take the second min next point

Method 3 Increase *maxoutward*

2.4 Else *outward* < *maxoutward* Then this move is allowed so update the points

PP = CP

If move to the min next point Then CP = NP(*minn*)

Else move to the second min next point Then CP = NP(*min2n*)

End If

$L_2 = L_2 + 1$

If the Euclidean distance between CP and FP is shorter or moving toward FP Then

outward = 0

Else the Euclidean distance between CP and FP is longer or moving outward FP Then

outward = *outward* + 1

End If

End If

Loop

Step 3

If $L_1 <= L_2$ or $L_3 <= L_2$ Then

Travel_Path_Distance = $L_1 + L_2 + L_3$ (Equation (5))

Else

Travel_Path_Distance = Euclidean distance between TF_1 and TF_2

End If

End Travel_Path_Distance

The pseudo code of the Particle Swarm Optimisation (PSO) algorithm

Start PSO algorithm

Initialise input variables

Step 1. Encode and populate the first generation of Particles in the swarm

For each Particle *i* in *swarmsize*

Populate Particle *i*

If Particle *i* is invalid Then repopulate it

Next Particle

Do While the ending criteria are not met

Step 2. Evaluate the fitness value of particles and remember the best particles

For each Particle *i* in *swarmsize*

Calculate the Fitness value using Equation (1)

If the Fitness value is better than the individual $pbest_i$ value Then

Set the current Particle value as the new $pbest_i$

End If

If the Fitness value is better than the global *gbest* value Then

Set the current Particle value as the new g*best*

End If

Next Particle

Step 3. Update velocities and positions of Particles in the swarm

For each Particle *i* in *swarmsize*

For each Facility *f* in a Particle

Generate the random variable of Facility *f*

Calculate the velocity of Facility *f* of the Particle using Equation (6)

Update the position of Facility *f* of the Particle using Equation (7)

If Facility *f* of Particle *i* is within the site boundary (Equation (2))

If Facility *f* of Particle *i* does not overlap with the other facilities (Equation (3))

If Facility *f* of Particle *i* is satisfied with inter-facility distance constraint (Equation (4))

Then

Exit For; this Facility *f* of Particle *i* is valid

Else

Reset previous facilities before Facility *f*

Loop and regenerate new position of Facility *f*

End If

Next Facility

Next Particle

Step 4. Evolve the swarm until the ending criteria are met

Loop until the ending criteria are met

End PSO algorithm