PAVEMENT MAINTENANCE

SCHEDULING OPTIMIZATION

Submitted by:

陳志榮 (M10605509)

Submitted to:

PROF. YO-MING HSIEH
Department of Construction and Civil Engineering

Spring Semester
July 4, 2018

Table Contents

1. Problem Definition

1.1 Pavement deterioration and defects

Pavement plays an important role in transportation infrastructures and certainly requires periodic maintenance and repair treatments (Shahnazari et al. 2012) to prevent undue distress or to restore performance. These distresses may be classified and is not limited to: rutting, fatigue, longitudinal crack, transverse crack, block crack, patch or pothole, raveling, and bleeding (Gharaibeh, Zou, and Saliminejad 2010). However, pavement maintenance and its impacts do not receive sufficient attention in many cases, and are either ignored or treated as low priority (Zhang and Mohsen 2018).

1.2 PCI definition

Pavement Construction Index (PCI) is one of the common indices to quantify pavement condition through visual surveys, which was developed by the U.S. Army Corps of Engineers (ASTM 2003). The PCI values ranges from 0 to 100, with 100 representing the perfect rate for a newly constructed pavement and 0 for the pavement having the worst condition possible. In this study, a value of 70 and above is considered to be at the lowest distress severity level making it in the least priority in the road rehabilitation scheduling.

1.3 Data acquisition

In Taiwan, there are 1069 road sections surveyed from the database of the Taiwanese government infrastructure projects. This number leads to a puzzling problem on which roads should be given priority on a limited allocated budget by the Taiwanese government. The 67.6% of these roads are considered to be in condition based on CPI of above 70 points. The roads with a CPI between 40 to 70 points consist the 22.1% of the road section and remaining 10.3% are the roads, which have the severest condition.

1.4 Scheduling optimization methods

There are optimization methods conducted by researchers to solve scheduling networks problems. (Dekker 1996) mentioned that optimization methods employed include linear and nonlinear programming, dynamic programming, Markov decision methods, decision analysis techniques, search techniques and heuristic approaches. A study conducted by (Panda and Swamy 2018), where they used the Artificial Bee Colony, a heuristic approach, for optimizing the pavement resurfacing maintenance activities.

1.5 Research objectives

This study aims to provide a scheduling tool for the Taiwanese government officials who are responsible for the road maintenance projects. The scheduling tool prioritizes the road sections that will provide the optimum benefit that will be acquired after the maintenance operations on the road pavements. The primary variables for the scheduling prioritization are the road section area and the PCI values.

2. Algorithmic Design

2.1 Setting the objective function

The fitness value is a function which is dependent on the PCI and the corresponding road section area. It is calculated as benefit and expressed as follows:

$$Benefit = \sum_{i=1}^{n} (100 - 0.8PCI) \times Area_i \times k \qquad (1)$$

where:
k = 0, when PCI > 70
k = 1, when $0 \le$ PCI ≤ 70
n = total number of roads sections

Equation (1) is the objective function for Genetic Algorithm (GA), which will provide the fitness values for the scheduling problem.

2.2 Penalty function

A penalty shall be introduce when the cost is greater than the budget and expressed in equation (2). The range values for alpha (α), beta (β), and gamma (γ) are listed in *Table 2.1*.

$$Penalty = (Cost - Budget)^{\alpha} \times \beta \times generation^{\gamma} \qquad (2)$$

2.3 Combination of GA and PSO

Particle Swarm Optimization (PSO) algorithm is used to optimize the parameters in setting the objective function of GA, which are: crossover rate (C_r), mutation rate (M_r), alpha (α), beta (β), and gamma (γ). PSO searches for the optimum combination of the parameters within the range values as listed in *Table 2.1*.

Table 2.1 - GA parameters

Parameter	Code variable	Range Values
Crossover rate	*Cr*	[0,1]
Mutation rate	*Mr*	[0,1]
Alpha	*alpha*	[0.3,1.3]
Beta	*beta*	[0,1]
Gamma	*gamma*	[0.3,1.3]

2.4 Fine tuning of GA parameters

The optimized values computed by PSO is used as the fixed parameters in GA. Therefore, the user will gain default values for all the parameters to be used in GA from the results of PSO from the constraint values listed in *Table 2.1*.

Table 2.2 – PSO default parameters

Parameter	Code variable	Input Value
Swarm size	*swarm_size*	15
Number of iterations	*iteration*	50
Acceleration constants	*c1, c2*	2.05
Velocity penalty	*velocity_penalty*	0.70
Search interval	*search_interval*	5
God choose rate	*god_choose_rate*	0.02

2.5 Algorithm flowchart

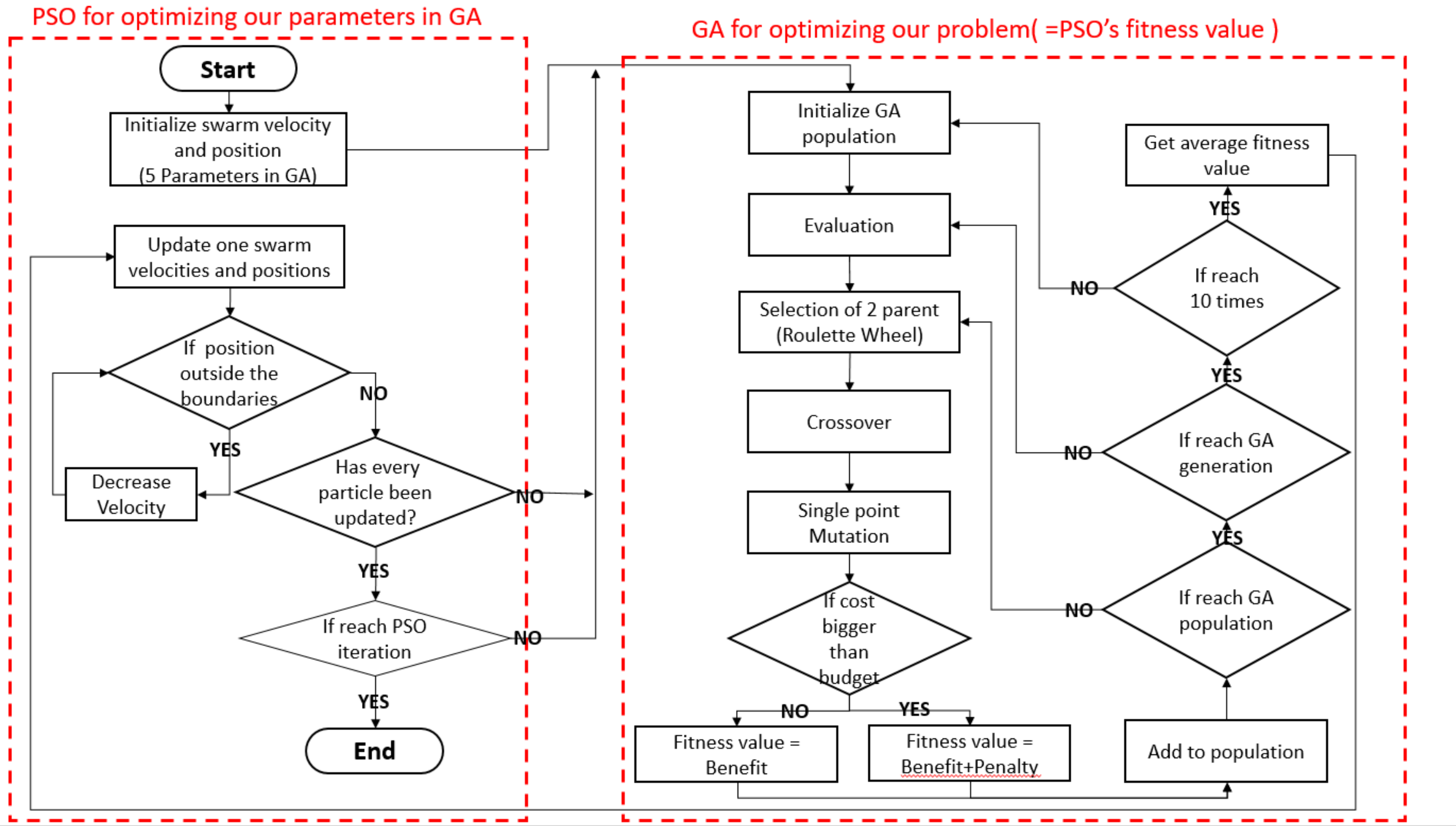


Figure 1 - Flowchart for PSO-GA optimization algorithm

3. Optimization Results

3.1 PSO optimization results for parameters of GA

After 50 iterations in PSO, the parameter values of GA are recorded in Table 3.3 as well as

1. Table 3.3 – Optimized GA parameters

Parameter	Code variable	Optimum Values
Crossover rate	Cr	0.06006
Mutation rate	Mr	0.31459
Alpha	alpha	0.88428
Beta	beta	0.71797
Gamma	gamma	0.61730

The simulation for PSO lasted for approximately 5 hours to run 35 iterations and reach an acceptable fitness value. The convergence history of PSO is shown in

Figure *2*. The optimized parameters was also verified through the convergence each parameters as shown in Figure 3. The convergence in each plot in Figure 3 confirms the optimum values listed in Table 3.3.

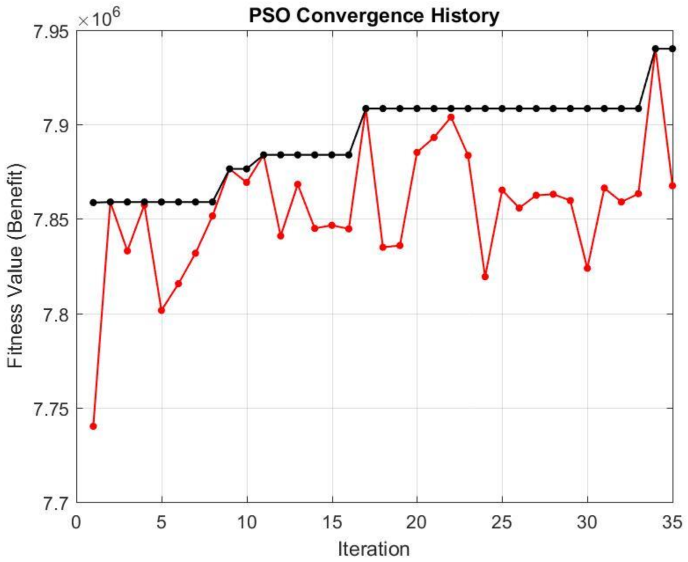
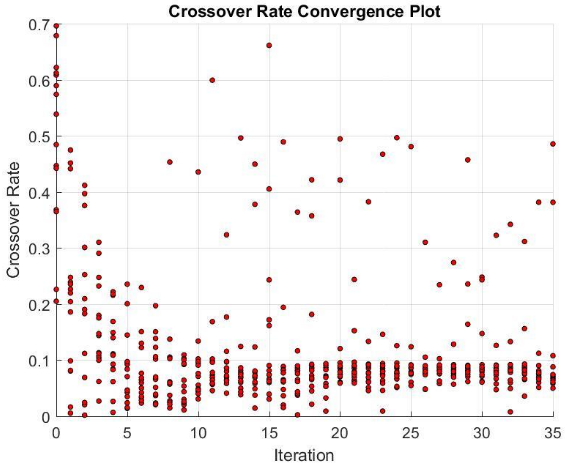


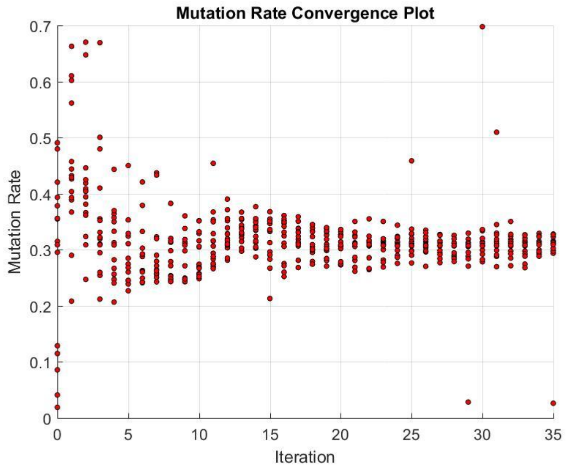
Figure 2 - PSO Convergence History

Another noticeable observation is for the values of alpha as illustrated in 3c. The alpha values that would provide a non-zero fitness value are those approximately greater than 0.8.

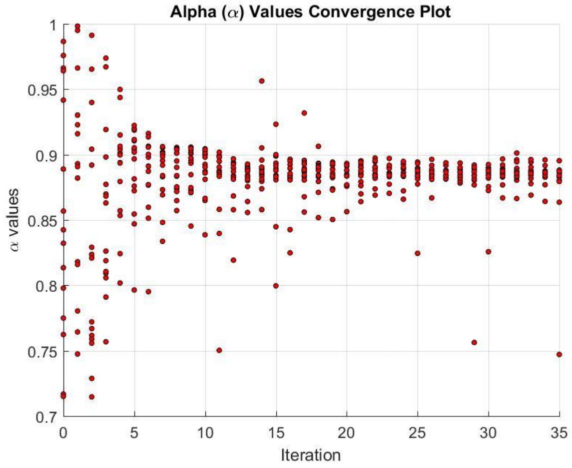
Figure 3 – Parameters Convergence Plots



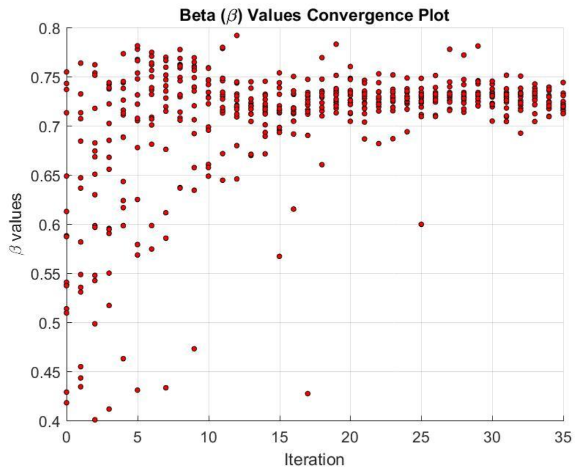
(a)



(b)

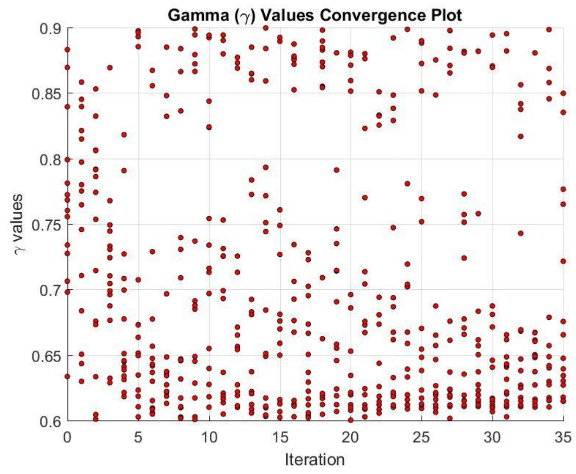


(c)



Beta (β) Values Convergence Plot
β values
Iteration
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0
5
10
15
20
25
30
35

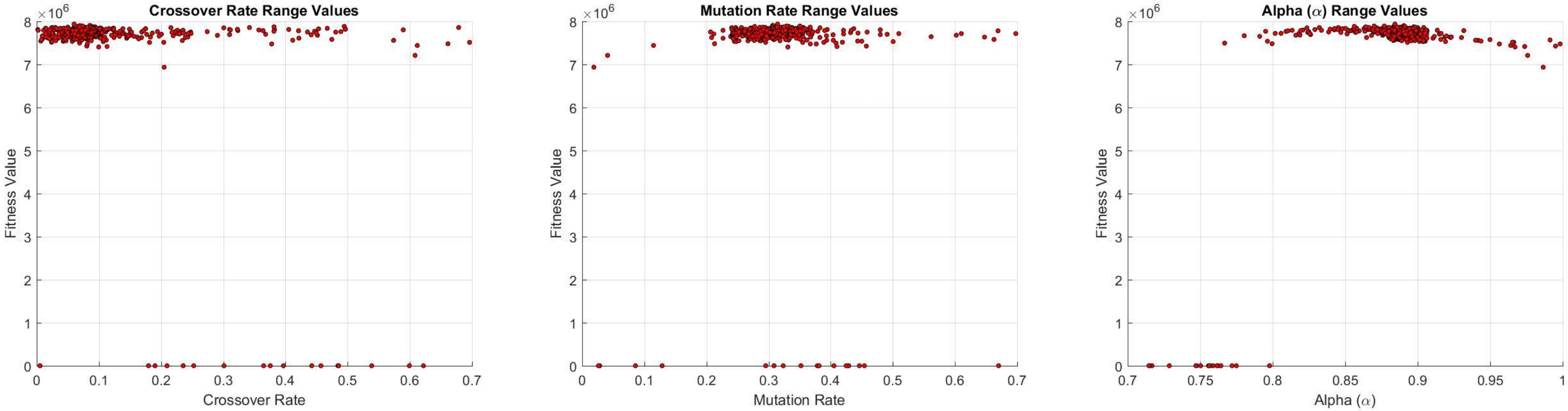
(d)



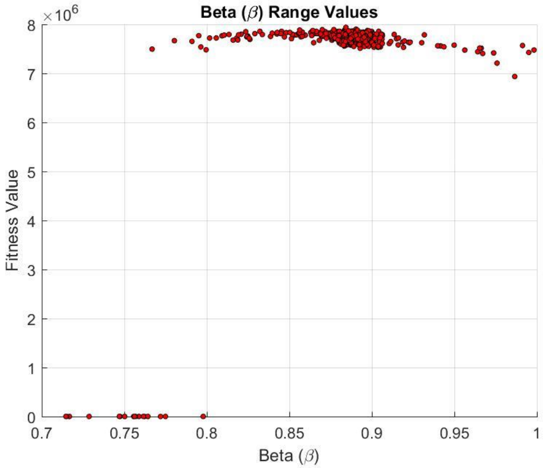
Gamma (γ) Values Convergence Plot
γ values
Iteration
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0
5
10
15
20
25
30
35

(e)

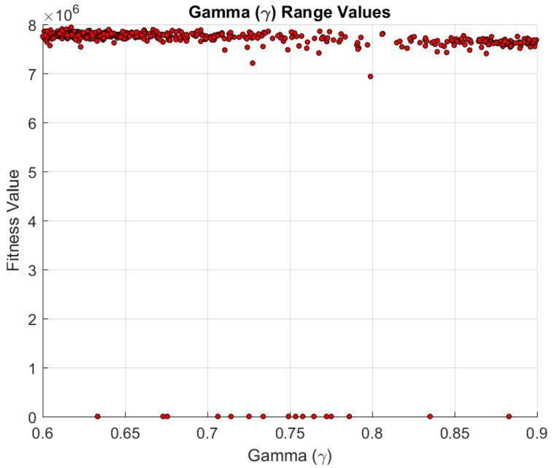
Figure 4 – Parameters vs Fitness Value Plots



(a) *(b)* *(c)*


Beta (β) Range Values
×10^6
Fitness Value
Beta (β)
0.7
0.75
0.8
0.85
0.9
0.95
1
0
1
2
3
4
5
6
7
8

(d)

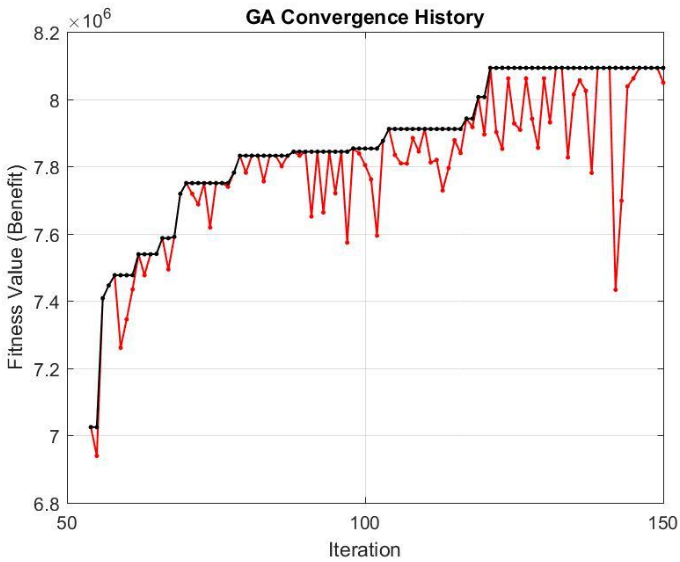

Gamma (γ) Range Values
×10^6
Fitness Value
Gamma (γ)
0.6
0.65
0.7
0.75
0.8
0.85
0.9
0
1
2
3
4
5
6
7
8

(e)

3.2 GA results

2. Table 3.4 - Optimum Scheduling Solution

Priority Rank	Road ID Number
1st	PC000741
2nd	SC000751
3rd	LJ000970
4th	LK000013
5th	SL001749
6th	PC004622
7th	CH000374
8th	CH000191
9th	SC000752
10th	PC000219
11th	SC001086
12th	BL001220
13th	TW000083
14th	SC001259
15th	CH000193
Total Cost:	**49,898,180**
Benefit:	**8,092,910**



3.

Figure 5 - GA Convergence History

3.3 Validations

To validate the performance of the proposed algorithm, the branch and bound method is used a baseline model for comparison of results. Mann-Whitney test was used to compare the performance of the results of the two algorithms. The fitness values for each algorithm are ranked accordingly as shown in Table .

4. Table 3.3 - Mann-Whitney test ranking of fitness values

BBM	8381891	8381891	8381891	8381891	8381891	8381891
Rank	9.5	9.5	9.5	9.5	9.5	9.5
PSO-GA	7883750	8158840	8104930	8040800	8024240	8066870
Rank	6	5	4	3	1	2

To check whether there is a statistical difference between the two algorithms, the null hypothesis H_0 was defined to be:

$H_0 =$ GA and BBM are no different

$H_1 =$ GA and BBM perform differently.

With an α value at 0.05, the U values yield to:

$$U_1 = 36$$
$$U_2 = 0$$
$$U = 0$$

$$P(U \leq 0 \mid H_0 is\ true) = 0.01 < 0.05;$$

The last equation above pointed out to reject the null hypothesis, H_0 and conclude that the performance of BBM is statistically better than GA.

5. Table 3.4 - Error rate compare with enumeration result

	Fitness value mean	**Error rate**
PSO-GA	8046571	4.00%
BBH (Global best solution)	8381891	0.00%

4. Parallel result

4.1 The input of the parallelized version (Same as the serial version)

Table 4.1 - Input of my program

Parameter	Code variable	Values
Input file name	*argv[1]*	data/testfile.csv
Output file name	*argv[2]*	testfile.json
Crossover rate	*argv[3]*	0.06006
Mutation rate	*argv[4]*	0.31459
Alpha	*argv[5]*	0.88428
Beta	*argv[6]*	0.71797
Gamma	*argv[7]*	0.61730
Budget	*argv[8]*	50,000,000

Table 4.2- Example on Makefile

Example
test: main.exe *./main.exe data/testfile.csv testfile.json 0.06006 0.31459 0.88428 0.71797 0.61730 50000000*
test1: main.exe *./main.exe data/01.csv 01.json 0.06006 0.31459 0.88428 0.71797 0.61730 15000000*

4.2 Performance

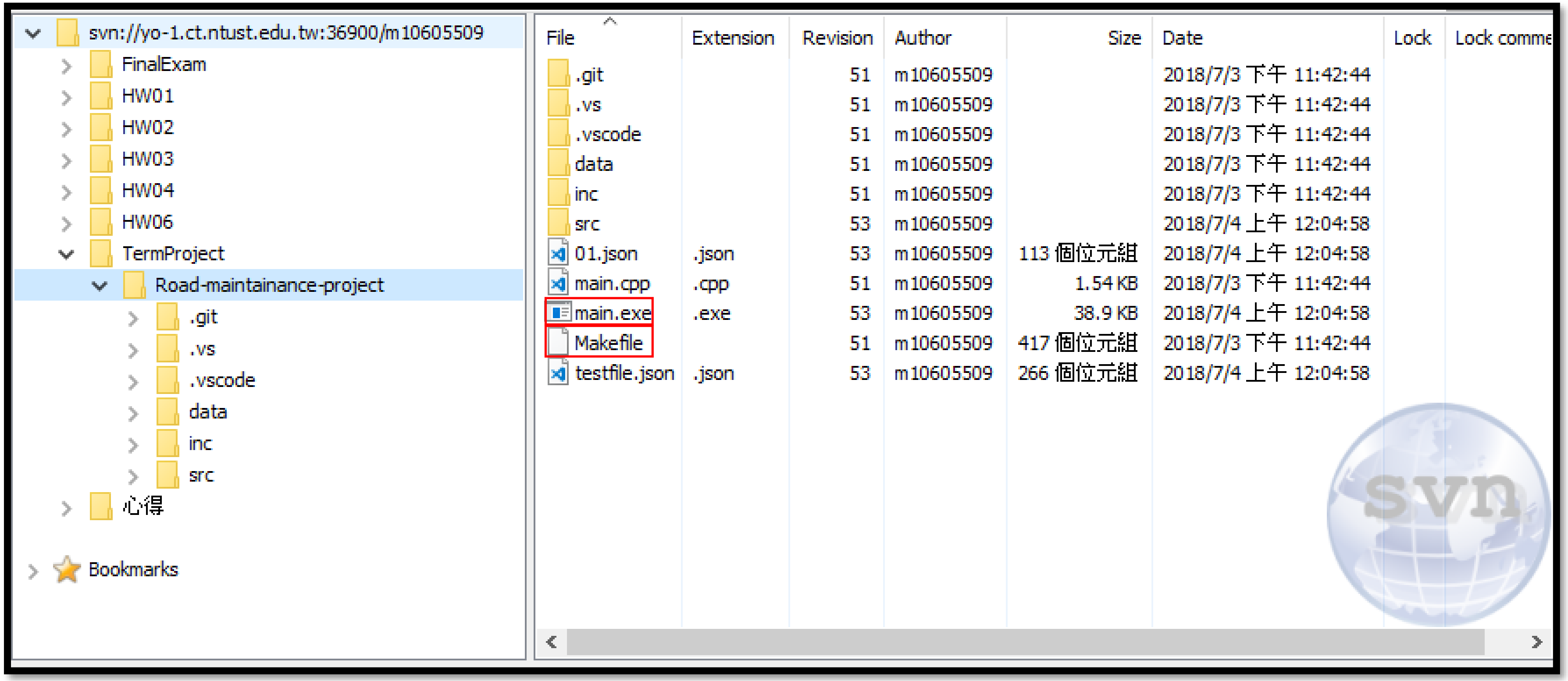
I accelerated the "Computefitness()" function because other functions slowed down after acceleration.

Table 4.3- Performance of the two difference data size file

	Data size	**Initialization**	**Rproduction**	**Cossover**	**Mutation**	**Compute fitness value**	**Output file**	**Total**	**speedup**
testfile.json (serial)	1069	0.0677542	0.249567	0.147143	0.300167	5.29926	0.000116	6.06406	100%
testfile.json (parallelized)	1069	0.066411	0.371364	0.373453	0.580377	2.07745	0.0000778	3.4692	175%
01.json (serial)	39	0.0105537	0.250217	0.0222528	0.0333437	0.558731	0.000117	0.875265	100%
01. json (parallelized)	39	0.0131984	0.381665	0.0396791	0.0637365	0.27081	0.000156	0.769314	114%

4.3 Implementation

Figure 1 - file path



4.4 Result validations between two version

If random seed is same, the program between serial and parallel version can get the totally same result. In this case, I use the first number as the random seed, so we can get the same result ..

```
    {
        "NUMBER":["CH000207,PC000741,LJ000970,LK001052,CH000374,CH000191,SC000752,RF000610,BL000493,SC001316,SC001086,CH000230,YG000280,BL001220,BL000238,CH000226,SC001259,JH000025,SG000362"],
        "COST":4.99962e+07,
        "BENEFIT":8.10721e+06
    }
    {
        "NUMBER":["CH000207,PC000741,LJ000970,LK001052,CH000374,CH000191,SC000752,RF000610,BL000493,SC001316,SC001086,CH000230,YG000280,BL001220,BL000238,CH000226,SC001259,JH000025,SG000362"],
        "COST":4.99962e+07,
        "BENEFIT":8.10721e+06
    }
```