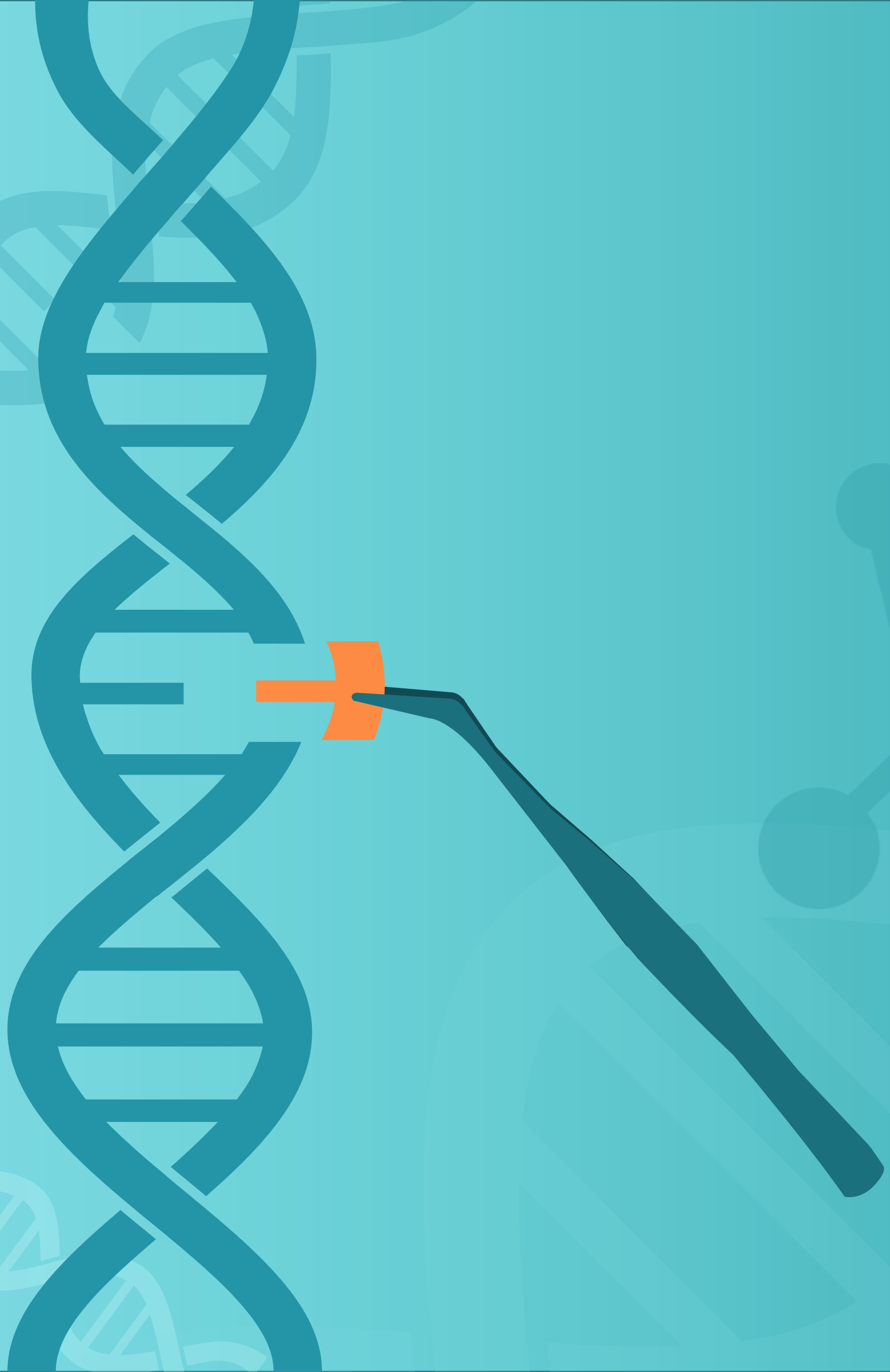


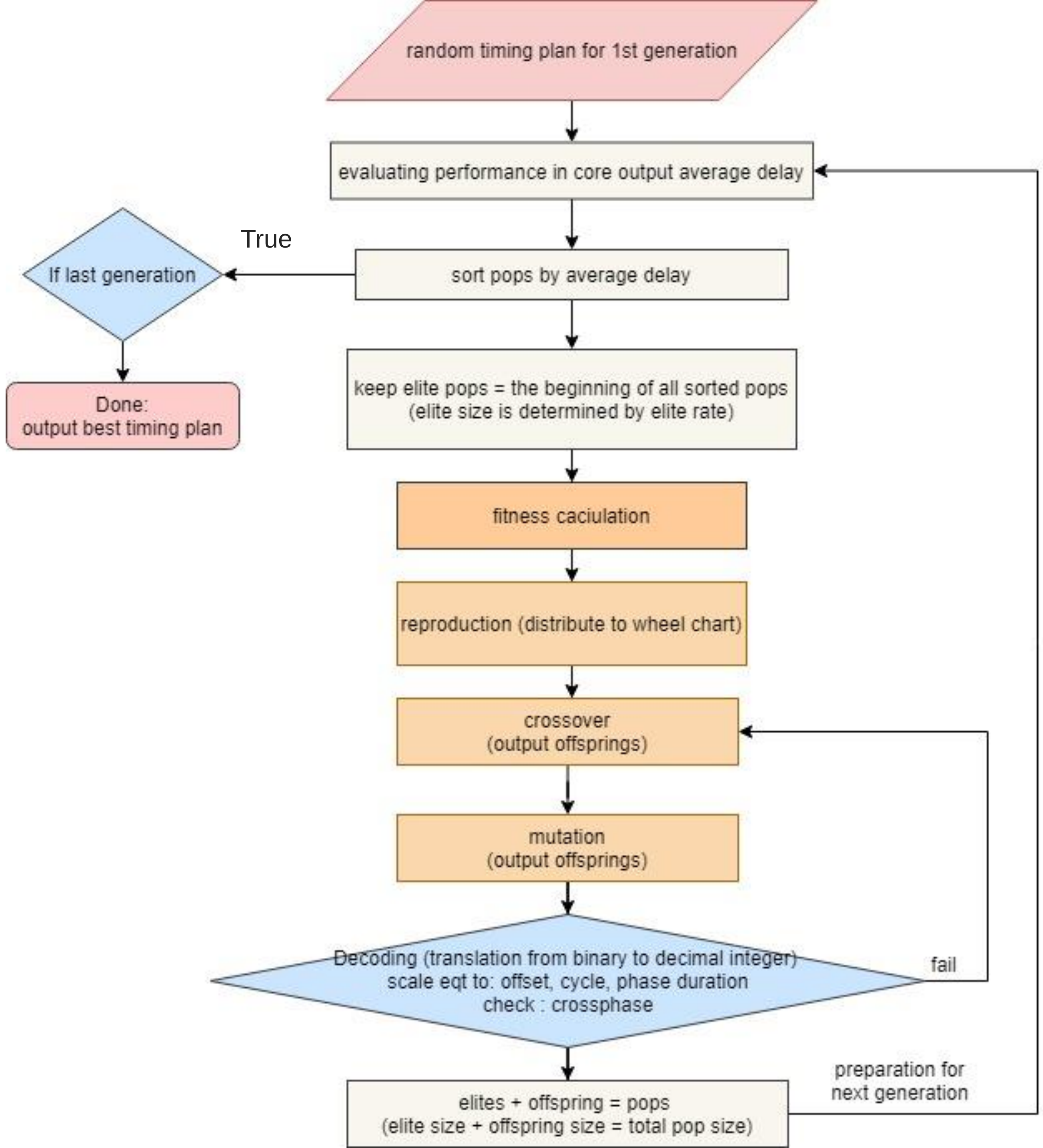
Genetic algorithms

Contents

GA Logic of NGA

Work flow of the GA

Flow chart of NGA



Initiating:

Input parameters

Total Simulation time

The expansion of timing plan, used in calculated number of cycle needed in VGVC, VGFC

Elite rate

Crossover rate x population size = offspring size
1- crossover rate = elite rate
Population size* elite rate = elite size
is used to determine size of elite in order to keep track on best result in each generation

Mutation rate

A occurrence percentage for mutation, bit flips

Power factor

P is for the amplifier on the fitness ratio effect

Variables

Population size

Testing population number for each of the generations

Number of generation

Number of the loop of GA

Stop criteria

A stopping criteria (tolerance) for a insignificant optimization after certain amount of generation

strategy

1.Simple GA element by element and randomly select
2.Whole gene (combined all binary chromosome)

Initiating: Initial phases Information data structure per each intersection

Example

Fixed by value for now

If false

Phase IDs	Initial phase duration	Boolean of this phase green changeable	Min green	intergreen	Crossphase IDs (vector of phase id)	Min crossphase
Phase green 1	Int	Bool	Int	Int	`vector<int>`	Int
Phase green 2	Int	Bool	Int	Int	`vector<int>`	Int
Phase green 3	Int	Bool	Int	Int	`vector<int>`	Int
Phase green 4	Int	Bool	Int	Int	`vector<int>`	Int

Initiating:

Initial intersection Information data structure for all population Example

Intersection IDs	Type of plan (per intersection)	Initial offset	Boolean of changeable of this offset	Initial cycle time	Boolean of changeable of this cycle	Max cycle time	Min cycle time	Phase duration inf collectinos	Intersection group (closely packed, related intersec)
Intersection 1	Fixed	Int	Bool	Int	Bool	Bool	Int	vector<phase_inf>	vector<IDs>
Intersection 2	VGVC	Int	Bool	Int	Bool	Bool	Int	vector<phase_inf>	vector<IDs>
Intersection 3	VGFC	Int	Bool	Int	Bool	Bool	Int	vector<phase_inf>	vector<IDs>
Intersection 4	Fixed	Int	Bool	Int	Bool	Bool	Int	vector<phase_inf>	vector<IDs>

Initiating:

phases genes data structure in 1 intersection
Example

Phase IDs	Decimal integer phases duration	Binary phases duration
Phase green 1	vector<int>	vector<bin>
Phase green 2	vector<int>	vector<bin>
Phase green 3	vector<int>	vector<bin>
Phase green 4	vector<int>	vector<bin>

Binary phases duration for 1st cycle	Binary phases duration for 2nd cycle	Binary phases duration for 3rd cycle
bin	bin	bin

...

Initiating:

1 intersection gene data structure in 1 population Example

intersection IDs	Decimal integer offset	Binary phases offset	Decimal integer cycle time	Decimal integer cycle time	Collections of phase gene
Intersection1	int	bin	Vector <int>	Vector <bin>	vector<phase_gene>
Intersection 2	int	bin	Vector <int>	Vector <bin>	vector<phase_gene>
Intersection 3	int	bin	Vector <int>	Vector <bin>	vector<phase_gene>
Intersection 4	int	bin	Vector <int>	Vector <bin>	vector<phase_gene>

Phase gene structure from previous slides

Int cycle time for 1st cycle	Int cycle time for 2nd cycle	Int cycle time for 3rd cycle	…
int	int	int	

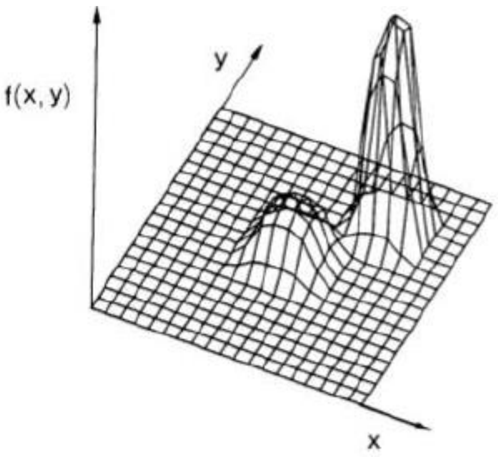
Case1 : normal GA (or called net GA)

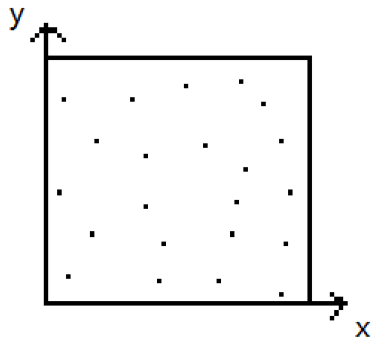
NGA logic

NGA 1st generation

1. Randomly generate timing plan

In first generation, randomly generate timing plan up to the amount of input population size.
One among populations is the initial setting of their own offset, cycle, phases, the left populations is in uniform distribution randomly generated within the range of limitation on inputted information data structure.





Cycle : uniform random between range ~ (min cycle , max cycle)
Offset : uniform random between range ~ (0 , max cycle)
Phases: uniform random range ~ (random cycle – min cycle)* (uniform random ratio) + min green

2. Evaluating performance

Pass populations to core to compute, output average delay

3. Sorting populations

According to their average delay, sorting the populations in ascending order.

4. Keep elite

Regarding to crossover rate, 1- crossover rate = elite rate
elite rate * population size = elite size
Populations after sorting, the first few individual populations according to elite size is selected out in each generation, keep refreshing in order to store the best results gene

NGA 1st generation

5. fitness ratio calculation

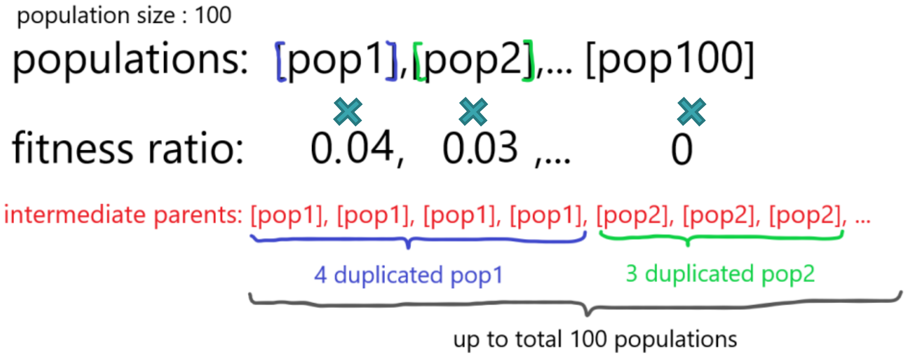
According the equation, fitness ratio is obtained in ascending order.

$$\mathrm{FIT}_i = \frac{A_i}{\sum A_i}$$

$$A_i = \exp\left[\left(\frac{\mathrm{TD}_{\max} - \mathrm{TD}_i}{\mathrm{TD}_{\max} - \mathrm{TD}_{\min}}\right) \cdot p\right]$$

6. Reproduction of the intermediate parents

The designated fitness ratio multiply by the population size representing the duplication number of that selected population kept in the intermediate parents populations. Same size with the population size.



7. Selections of single run crossover

a.) randomly selected the 2 parents populations from pervious intermediate populations
b.) avoid the repeatedly selected, so each population in intermediate parents population wheel chart would only been selected once.
c.) until all intermediate parents population wheel chart have been undergoing crossover outputting offsprings

NB.: Offspring size = population size = intermediate parent size

NGA 1st generation (Cont)

8. Crossover

a.) pick up a pair of population from previous selection in intermediate parents (same in 7. **Selections of single run crossover**)
b.) randomly selected the targeted chromosome (including offset, cycle length, phase green; those could be reselected and not guarantee every element is selected to undergo crossover
c.) randomly located single crossover point on each chromosome

Example:

4(a) 4(b)

Pop1: [offset, Cycle time, green1, green2, …]

Pop2: [offset, cycle time, green1, green2 …]

Pop3: [offset, cycle time, green1, green2 …]

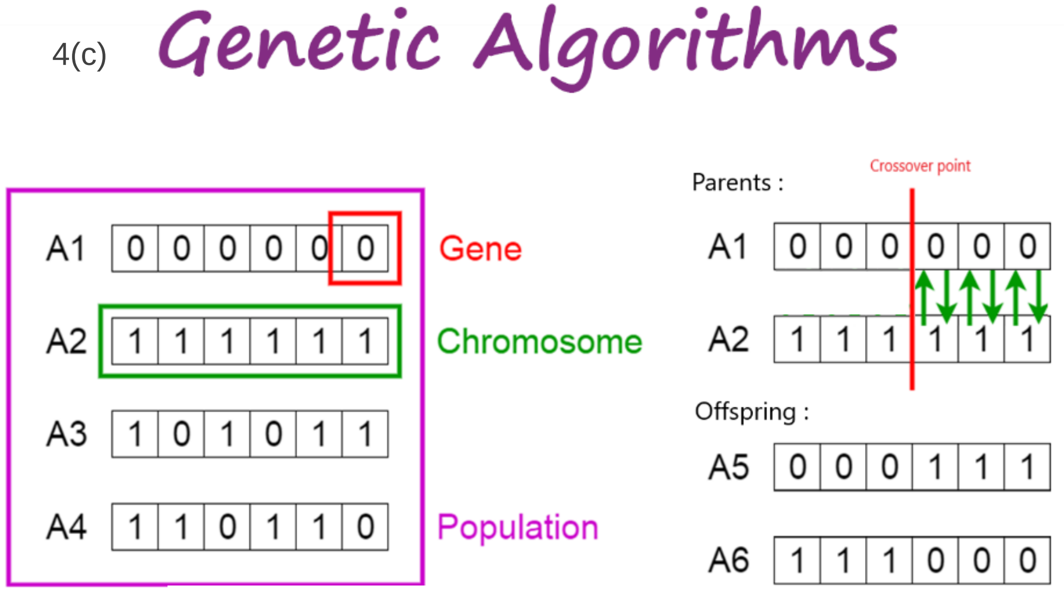
4(a) 4(b)

Pop4: [offset, Cycle time, green1, green2, …]

……

Pop100: [offset, cycle time, green1, green2 …]

4(c)



single crossover point :

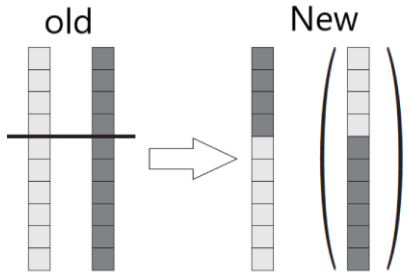


Fig.: Single-point Crossover (SPX).

NGA 1st generation (Cont)

9. mutation

The gene bit is read one by one inside a population
Whereas the random generation of value (range 0 ~1) in each gene bit is smaller than the occurrence of the mutation rate, the mutation is activated on the selected gene and then flip the bit

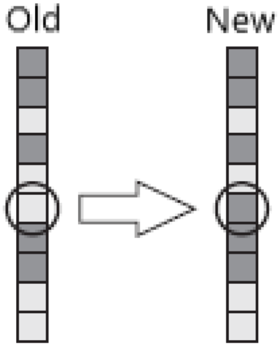


Fig.: Single-gene mutation.

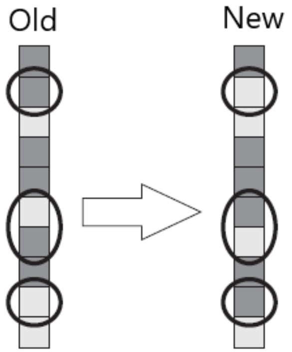


Fig. : Multi-gene mutation

10. Decoding (translation binary to decimal integer)

Offset, cycle and phases have their own scale equations

For offset, just direct bit wise translation,

(bin) (dec int)

eg. 00001001 -> 9

For Cycle,

$$C = CI \cdot \left(\frac{C\max - C\min}{127} \right) + C\min$$

For phases

$$PD_{ji} = (C - C^{j}\min) \left(\frac{PDI_{ji}}{\sum_{i} PDI_{ji}} \right) + G\min$$

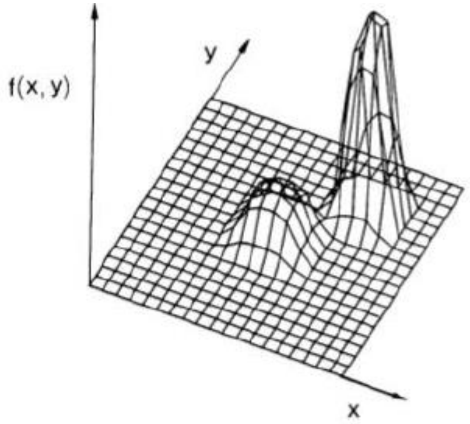
NGA 1st generation (Cont)

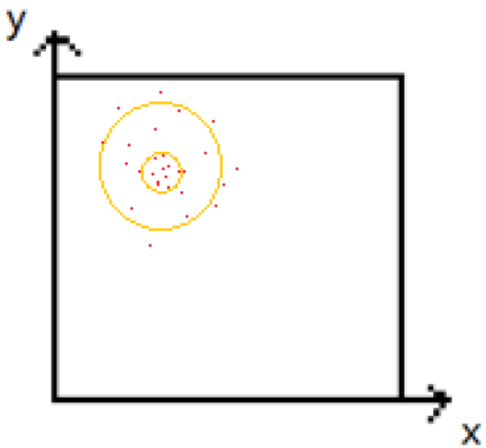
10. Decoding (translation binary to decimal integer) problems on phases translation

Bits representation capacity {0 – 256) | {0 – 256) Bits representation capacity

A	B
P1	P2

Total combination : 256^2 Total bits representation capacity





If consider available green time 30sec

Then the decoding of the chromosome:

$$\text{P1 g-time} = 30 \times \frac{A}{A+B}$$

50/50 split 41927 -> ~ 64%

NGA 1st generation (Cont)

11. Checking crossphase requirement

If current Phase duration >= min_crossphase - previous_phase_durations - previous_intergreens is true, then pass.

If failed, it goes back into the 8.crossover process again

12. Checking history for overlapping population

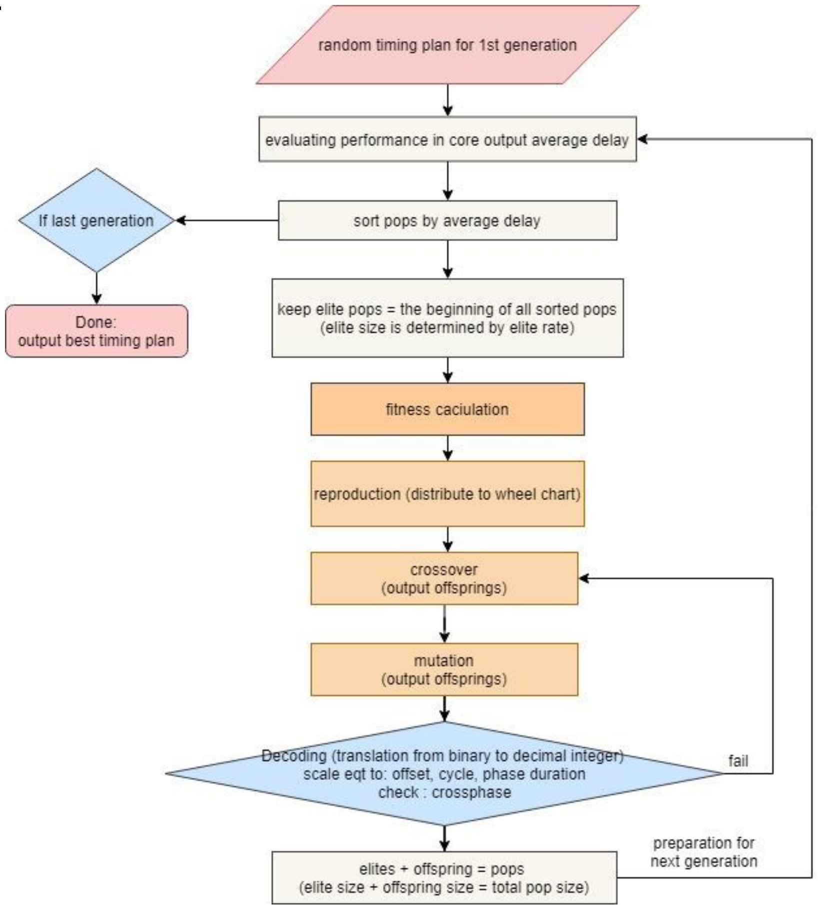
The calculated population cases would be stored up in history for checking the uniqueness of new generated offspring, avoid redundancy

13. Keep on prepare for next generation

Elite + offsprings = population for next generation
(elite size) + (offspring size) = population for next generation cropped out the exceeding population = population size

Then goes into process **2. Evaluating performance**

The unique newly generated offspring would be assign to computation task for next generation, until it reach the max iteration number stopping criteria and the max generation number



NGA

Explain and Examples

NGA **Example1**

Example: NGA; no_generation200; Populcation_size200; Elite_rate0.2; crossover_multiplier1.3; Mutation_rate0.005; Power_factor4; All cycle changeable; offset changeable; Include the initial_seed

All cycle, offset, phases changes independently

Ave delay 67.6884 is the initial seed when all intersection cycle time are 97

```
 //--- --- --- --- --- --- --- 197th generation --- --- --- --- --- --- --- //
< 67.2244 67.6884 69.5532 69.6636 71.2252 72.0381 74.6091 77.3692 79.1436 80.3884 82.3171 83.6019 84.0547 84.848 85.1686 86.0768
87.6933 87.9781 88.8752 89.3211 90.9454 92.9837 93.2856 93.4202 94.224 95.6884 96.367 96.4513 96.6194 96.6512 96.8234 97.2306 97.
499 99.7474 100.231 101.279 102.337 103.144 103.361 103.673 104.537 105.943 106.56 106.686 107.957 108.219 108.37 108.394 109.044
 109.207 110.526 110.56 110.682 110.806 110.998 112.196 112.59 113.461 113.886 114.579 >

 //--- --- --- --- --- --- --- 198th generation --- --- --- --- --- --- --- //
< 67.2244 67.6884 69.5532 69.6636 71.2252 72.0381 74.6091 77.3692 79.1436 80.3884 82.3171 83.6019 84.0547 84.848 85.1686 86.0768
87.6933 87.9781 88.8752 89.3211 90.9454 92.9837 93.2856 93.4202 94.224 95.6884 96.367 96.4513 96.6194 96.6512 96.8234 97.2306 97.
499 99.7474 100.231 101.279 102.337 103.144 103.361 103.673 104.537 105.943 106.56 106.686 107.957 108.219 108.37 108.394 109.044
 109.207 110.526 110.56 110.682 110.806 110.998 112.196 112.59 113.461 113.886 114.579 >

 //--- --- --- --- --- --- --- 199th generation --- --- --- --- --- --- --- //
< 67.2244 67.6884 69.5532 69.6636 71.2252 72.0381 74.6091 77.3692 79.1436 80.3884 82.3171 83.6019 84.0547 84.848 85.1686 86.0768
87.6933 87.9781 88.8752 89.3211 90.9454 92.9837 93.2799 93.2856 93.4202 94.224 95.6884 96.367 96.4513 96.6194 96.6512 96.8234 97.
2306 97.499 99.7474 100.231 101.279 102.337 103.144 103.361 103.673 104.537 105.943 106.56 106.686 107.957 108.219 108.37 108.394
 109.044 109.207 110.526 110.56 110.682 110.806 110.998 112.196 112.59 113.461 113.886 >

 //--- --- --- --- --- --- --- 200th generation --- --- --- --- --- --- --- //
< 67.2244 67.6884 69.5532 69.6636 71.2252 72.0381 74.6091 77.3692 79.1436 80.3884 82.3171 83.6019 84.0547 84.848 85.1686 86.0768
87.6933 87.9781 88.8752 89.3211 90.9454 92.9837 93.2799 93.2856 93.4202 94.224 95.6884 96.367 96.4513 96.6194 96.6512 96.8234 97.
2306 97.499 99.7474 100.231 101.279 102.337 103.144 103.361 103.673 104.537 105.943 106.56 106.686 107.957 108.219 108.37 108.394
 109.044 109.207 110.526 110.56 110.682 110.806 110.998 112.196 112.59 113.461 113.886 >
->> the top chromosome are printed out.
average daley: 67.2244
 intersection_id: 0      Cycle: 97      Offset: 66      phase : [17     30      35      ]
 intersection_id: 2      Cycle: 97      Offset: 81      phase : [36     51      ]
 intersection_id: 10     Cycle: 97      Offset: 45      phase : [17     47      0       16      ]
 intersection_id: 4      Cycle: 97      Offset: 91      phase : [31     21      24      ]
 intersection_id: 5      Cycle: 82      Offset: 26      phase : [46     26      ]
 intersection_id: 13     Cycle: 101     Offset: 22      phase : [23     14      11      28      ]
 intersection_id: 9      Cycle: 97      Offset: 45      phase : [27     12      17      20      ]
 intersection_id: 11     Cycle: 97      Offset: 53      phase : [55     32      ]


317 seconds run time
```

NGA **Example2**

Example: NGA; no_generation200; Populcation_size200; Elite_rate0.2; crossover_multiplier1.3; Mutation_rate0.005; Power_factor4; All cycle changeable; offset changeable; Include the initial_seed

Ave delay 67.6884 is the initial seed when all intersection cycle time are 97

All intersections are within the same intersection group, so they synchronized

```
//--- --- --- --- --- --- --- 197th generation --- --- --- --- --- --- --- //
< 63.6204 67.6884 69.5874 73.9068 83.1529 88.4017 89.2326 89.6169 91.3516 92.5781 92.7241 94.0913 94.2782 95.5
443 96.9238 97.0295 97.0575 97.3858 97.4651 98.5117 99.0468 99.5828 99.7462 100.055 100.401 100.471 100.902 10
1.283 101.529 101.963 101.963 102.044 102.128 103.143 103.202 103.671 104.091 104.225 104.784 104.841 105.474
106.249 106.382 106.744 106.812 107.921 108.097 108.35 108.373 109.566 109.774 110.174 110.588 110.853 110.949
 111.074 111.248 111.55 111.554 111.583 >

 //--- --- --- --- --- --- --- 198th generation --- --- --- --- --- --- --- //
< 63.6204 67.6884 69.5874 73.9068 83.1529 88.4017 89.2326 89.6169 91.3516 92.5781 92.7241 94.0913 94.2782 95.5
443 96.9238 97.0295 97.0575 97.3858 97.4651 98.5117 99.0468 99.5828 99.7462 100.055 100.401 100.471 100.902 10
1.283 101.529 101.963 101.963 102.044 102.128 103.143 103.202 103.671 104.091 104.225 104.784 104.841 105.474
106.249 106.382 106.744 106.812 107.921 108.097 108.35 108.373 109.566 109.774 110.174 110.588 110.853 110.949
 111.074 111.248 111.55 111.554 111.583 >

 //--- --- --- --- --- --- --- 199th generation --- --- --- --- --- --- --- //
< 63.6204 67.6884 69.5874 73.9068 83.1529 88.4017 89.2326 89.6169 91.3516 92.5781 92.7241 94.0913 94.2782 95.5
443 96.9238 97.0295 97.0575 97.3858 97.4651 98.5117 99.0468 99.5828 99.7462 100.055 100.401 100.471 100.902 10
1.283 101.529 101.963 101.963 102.044 102.128 103.143 103.202 103.671 104.091 104.225 104.784 104.841 105.474
106.249 106.382 106.744 106.812 107.921 108.097 108.35 108.373 109.566 109.774 110.174 110.588 110.853 110.949
 111.074 111.248 111.55 111.554 111.583 >

 //--- --- --- --- --- --- --- 200th generation --- --- --- --- --- --- --- //
< 63.6204 67.6884 69.5874 73.9068 83.1529 88.4017 89.2326 89.6169 91.3516 92.5781 92.7241 94.0913 94.2782 95.5
443 96.9238 97.0295 97.0575 97.3858 97.4651 98.5117 99.0468 99.5828 99.7462 100.055 100.401 100.471 100.902 10
1.283 101.529 101.963 101.963 102.044 102.128 103.143 103.202 103.671 104.091 104.225 104.784 104.841 105.474
106.249 106.382 106.744 106.781 106.812 107.921 108.097 108.35 108.373 109.566 109.774 110.174 110.588 110.853
 110.949 111.074 111.248 111.55 111.554 >
->> the top chromosome are printed out.
average daley: 63.6204
 intersection_id: 0      Cycle: 97      Offset: 66      phase : [22     32      28      ]
 intersection_id: 10     Cycle: 97      Offset: 45      phase : [19     43      1       17      ]
 intersection_id: 2      Cycle: 97      Offset: 81      phase : [45     42      ]
 intersection_id: 4      Cycle: 97      Offset: 91      phase : [38     20      18      ]
 intersection_id: 13     Cycle: 97      Offset: 22      phase : [22     11      12      27      ]
 intersection_id: 5      Cycle: 97      Offset: 26      phase : [63     24      ]
 intersection_id: 9      Cycle: 97      Offset: 45      phase : [27     11      17      21      ]
 intersection_id: 11     Cycle: 97      Offset: 53      phase : [62     25      ]

367 seconds run time

C:\Users\royce\source\repos\tryDISCO_GA_MK\x64\Release\tryDISCO_GA_MK.exe (process 3916) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close
the console when debugging stops.
```

GA

New functions

01
Intersection group

They would have the synchronized cycle time

02
Wholegene

Only changeable elements are extracted out into wholegene for GA processing (crossover mutation)

03
Unfreeze group

Could both used in NGA and SGA, to freeze of unfreeze(optimize) the intersections

04(a)
Phase split : fixed by value

Exactly unchanged, fixed with its initial value

04(b)
Phase split : fixed by ratio

Green split ratio would kept unchanged

$$ratio = \frac{initial\ phase}{total\ phase}$$

04(c)
Phase split : random by ratio

According the scale function.

$$PD_{ji} = (C - C^{j}\min)\left(\frac{PDI_{ji}}{\sum_{i} PDI_{ji}}\right) + G\min \qquad (12)$$

for determine green durations

Case2 : Whole gene (continuous GA, combining all genes while crossover)

GA logic (we created)

Whole gene GA 1st generation (Cont)

Other computational procedures are the same with the above NGA but only except from Crossover session

8. Crossover

a.) pick up a pair of population from previous selection in intermediate parents (same in 7. **Selections of single run crossover**)
b.) randomly selected the targeted chromosome (including offset, cycle length, phase green; those could be reselected and not guarantee every element is selected to undergo crossover
c.) randomly located multiple crossover point on each chromosome

Example:

4(a)

```
Pop1: [offset + cycle time + green1 + green2]
Pop2: [offset + cycle time + green1 + green2]
Pop3: [offset + cycle time + green1 + green2]
```

4(a)

```
Pop4: [offset + cycle time + green1 + green2]
......
Pop100: [offset + cycle time + green1 + green2]
```

4(b)

```
Pop1: [0010010011010010110100001010001]
Pop4: [1000100011010101110100011100110]
```

4(c)

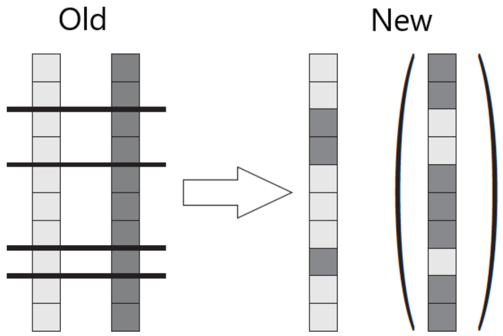
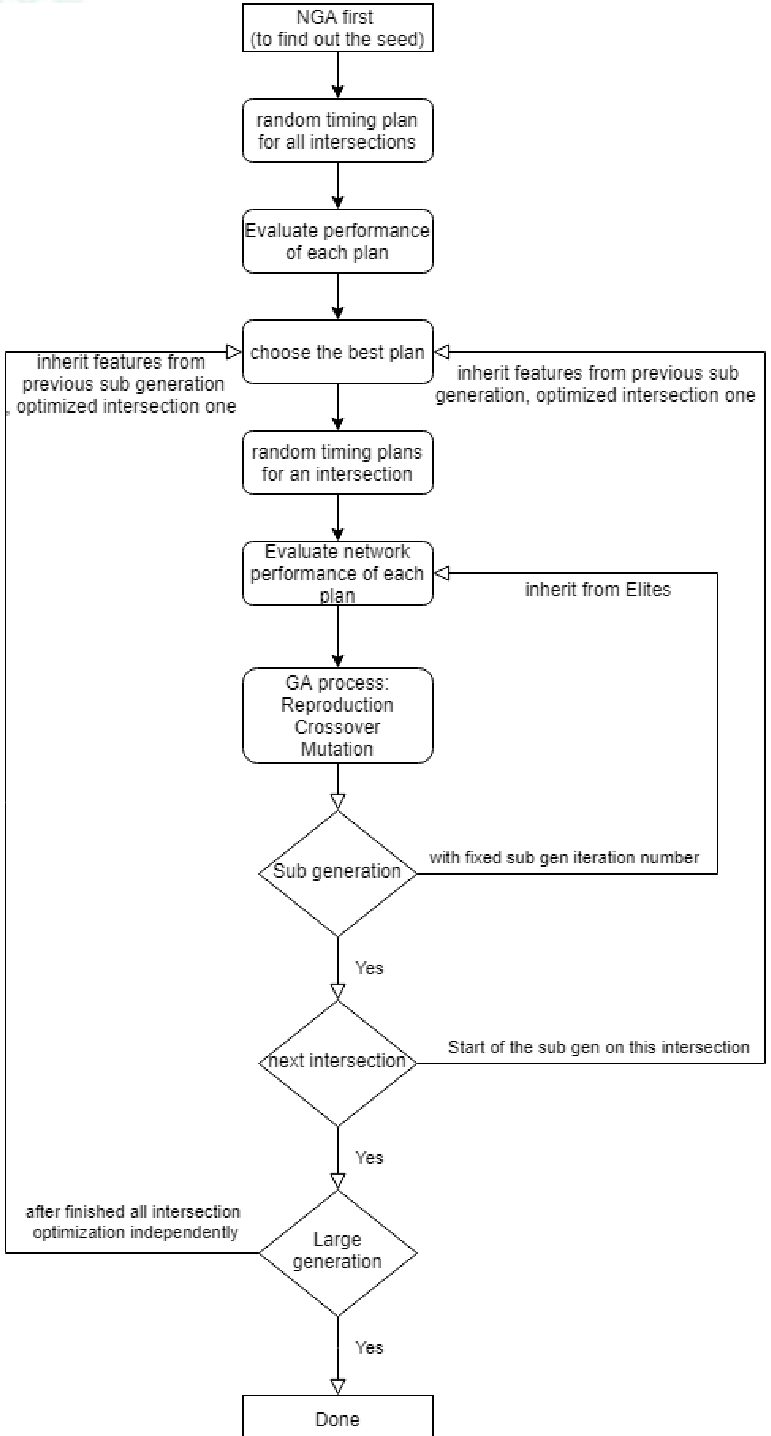


Fig.: Multi-point Crossover (MPX).

Case3 : Sequential GA (SGA)

SGA logic

Flow chart of SGA



SGA

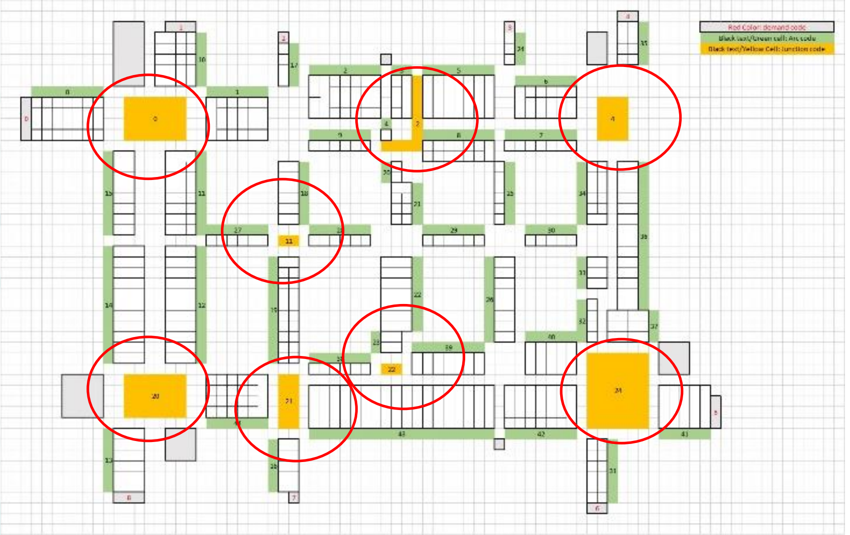
Features:

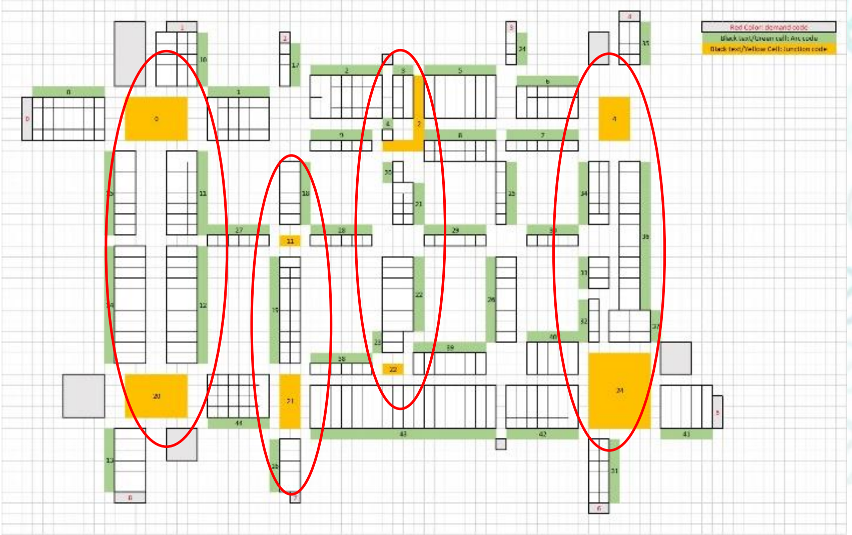
- Cycle time , offset, phases could independently treated according to manually settings either fixed or changeable

- The optimization of intersection(s) within sub generation loops could be either single one or a group of intersections

```
// --- --- --- --- --- --- --- --- --- --- building up unfreeze_list for SGA --- --- --- --- --- ---
std::vector <std::set <int >> This_unfreeze_list = { {0} ,{2 }, {10} ,{ 4}, {5}, {13}, {9}, {11} };
GA_Obj.set_unfreeze_list(This_unfreeze_list);
```

```
// --- --- --- --- --- --- --- --- --- --- building up unfreeze_list for SGA --- --- --- ---
std::vector <std::set <int >> This_unfreeze_list = { {0, 2} ,{10, 4 }, {5, 13} ,{ 9, 11} };
GA_Obj.set_unfreeze_list(This_unfreeze_list);
```





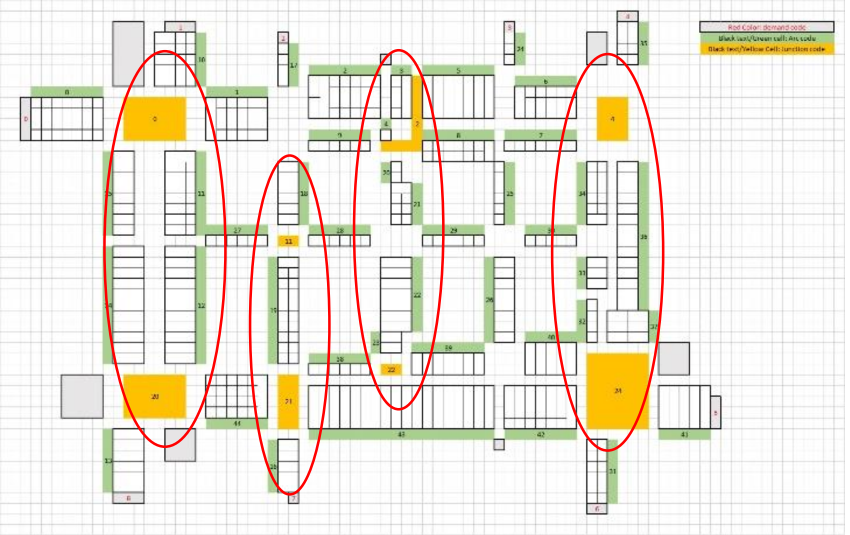
SGA

Features:

- When intersection group is applied, those should be within the same session of unfreeze list, otherwise it will crash if the cycle time is changeable

```
// --- --- --- --- --- --- --- --- --- --- building up unfreeze_list for SGA --- --- --- ---
std::vector <std::set <int >> This_unfreeze_list = { {0, 2} ,{10, 4 }, {5, 13} ,{ 9, 11} };
GA_Obj.set_unfreeze_list(This_unfreeze_list);
```

```
// --- --- --- --- --- --- --- --- --- --- intersection group --- --- --- --- --- --- --- --- -
std::vector<std::vector<int>> intersectionID_in_gps = { {0, 2} ,{10, 4 }, {5, 13} ,{ 9, 11} };
GA_Obj.set_intersections_gp(intersectionID_in_gps);
```



SGA Example

```
// --- --- --- --- --- --- --- --- --- --- building up unfreeze_list for SGA --- --- --- --- --- ---
std::vector <std::set <int >> This_unfreeze_list = { {0} ,{2 }, {10} ,{ 4}, {5}, {13}, {9}, {11} };
GA_Obj.set_unfreeze_list(This_unfreeze_list);
```

Example: SGA; no_sub_generation6; no_generation4; Populcation_size200; Elite_rate0.2; crossover_multiplier1.3; Mutation_rate0.005; Power_factor4; All cycle time fixed; offset changeable; Include the initial_seed; Optimized intersection by intersection

Microsoft Visual Studio Debug Console

```
 intersection_id: 2     Cycle: 97     Offset: 68     phase : [56    31     ]
 intersection_id: 4     Cycle: 97     Offset: 89     phase : [40    25     11     ]
 intersection_id: 13    Cycle: 97     Offset: 22     phase : [20    10     16     26     ]
 intersection_id: 5     Cycle: 97     Offset: 20     phase : [71    16     ]
 intersection_id: 9     Cycle: 97     Offset: 45     phase : [27    11     17     21     ]
 intersection_id: 11    Cycle: 97     Offset: 83     phase : [68    19     ]


  * * *  * * *  * * *  * * * intersections 9  r being optimized * * *  * * *  * * *  * * *

 //~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ 1th sub generation ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ //
< 58.2103 64.439 81.8094 84.5743 85.7273 88.1145 92.5811 95.8167 98.1559 98.9256 102.747 102.915 104.998 106.571 106.882 111.626
111.698 113.014 113.788 113.945 117.134 117.775 118.537 119.75 120.374 121.173 122.032 124.445 124.572 124.799 124.803 125.069 12
5.372 126.479 126.496 127.106 127.215 128.01 128.194 128.366 >

 //~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ 2th sub generation ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ //
< 58.2103 61.7357 64.439 67.4895 81.0637 81.8094 84.5743 85.7273 87.927 88.1145 92.5811 95.8167 98.1559 98.9256 102.747 102.915 1
04.998 106.571 106.882 110.426 111.626 111.698 113.014 113.788 113.945 117.134 117.775 118.537 119.75 120.374 121.173 122.032 124
.445 124.572 124.799 124.803 125.069 125.147 125.35 125.372 >

 //~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ 3th sub generation ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ //
< 58.2103 61.7357 64.439 67.4895 70.6302 81.0637 81.0655 81.8094 84.5743 84.6808 85.7273 87.927 87.927 88.1145 92.5811 95.8167 98
.1559 98.9256 102.747 102.915 104.998 106.571 106.882 110.426 111.626 111.698 113.014 113.788 113.945 117.134 117.775 118.537 118
.807 119.75 120.374 121.173 122.032 124.445 124.572 124.799 >

 //~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ 4th sub generation ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ //
< 58.2103 61.7357 64.439 67.4895 70.6302 76.8925 81.0637 81.0655 81.8094 82.3877 84.5743 84.6808 85.7273 87.927 87.927 88.1145 88
.5788 90.8304 92.5811 95.8167 98.1559 98.9256 102.747 102.915 104.998 106.571 106.882 110.426 111.626 111.698 113.014 113.223 113
.788 113.945 117.134 117.775 118.537 118.807 119.75 120.374 >

 //~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ 5th sub generation ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ //
< 58.2103 61.7357 64.439 67.4895 70.6302 70.6432 70.6927 76.8925 76.8925 77.4569 81.0637 81.0655 81.8094 82.3877 84.5743 84.6808
85.7273 87.927 87.927 88.1145 88.5788 90.8304 92.5811 95.8167 98.1559 98.9256 102.747 102.915 104.998 106.571 106.882 110.426 111
.626 111.698 113.014 113.223 113.788 113.945 117.134 117.775 >

 //~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ 6th sub generation ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ //
< 52.138 58.2103 61.7357 62.2862 64.439 67.4895 70.6302 70.6432 70.6927 76.8925 76.8925 77.4569 81.0637 81.0655 81.8094 82.3877 8
2.2904 84.5743 84.6808 85.7273 86.3298 87.927 87.927 87.927 88.1145 88.5788 90.8304 92.5811 95.8167 98.1559 98.9256 102.747 102.9
15 104.998 106.571 106.882 110.426 110.516 111.626 111.698 >
->> the top chromosome are printed out.
average daley: 52.138
 intersection_id: 0     Cycle: 97     Offset: 60     phase : [18    37     27     ]
 intersection_id: 2     Cycle: 97     Offset: 68     phase : [56    31     ]
 intersection_id: 10    Cycle: 97     Offset: 46     phase : [23    31     2      24     ]
 intersection_id: 4     Cycle: 97     Offset: 89     phase : [40    25     11     ]
 intersection_id: 5     Cycle: 97     Offset: 20     phase : [71    16     ]
 intersection_id: 13    Cycle: 97     Offset: 22     phase : [20    10     16     26     ]
 intersection_id: 9     Cycle: 97     Offset: 43     phase : [28    12     16     20     ]
 intersection_id: 11    Cycle: 97     Offset: 83     phase : [68    19     ]
```

```
  * * *  * * *  * * *  * * * intersections 11  r being optimized * * *  * * *  * * *  * * *

 //~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ 1th sub generation ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ //
< 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 5
2.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.1
38 52.138 52.138 52.138 >

 //~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ 2th sub generation ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ //
< 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 5
2.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.1
38 52.138 52.138 52.138 >

 //~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ 3th sub generation ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ //
< 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 5
2.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.1
38 52.138 52.138 52.138 >

 //~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ 4th sub generation ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ //
< 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 5
2.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.1
38 52.138 52.138 52.138 >

 //~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ 5th sub generation ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ //
< 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 5
2.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.1
38 52.138 52.138 52.138 >

 //~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ 6th sub generation ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ //
< 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 5
2.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.138 52.1
38 52.138 52.138 52.138 >
->> the top chromosome are printed out.
average daley: 52.138
 intersection_id: 0     Cycle: 97     Offset: 60     phase : [18    37     27     ]
 intersection_id: 2     Cycle: 97     Offset: 68     phase : [56    31     ]
 intersection_id: 10    Cycle: 97     Offset: 46     phase : [23    31     2      24     ]
 intersection_id: 4     Cycle: 97     Offset: 89     phase : [40    25     11     ]
 intersection_id: 5     Cycle: 97     Offset: 20     phase : [71    16     ]
 intersection_id: 13    Cycle: 97     Offset: 22     phase : [20    10     16     26     ]
 intersection_id: 9     Cycle: 97     Offset: 43     phase : [28    12     16     20     ]
 intersection_id: 11    Cycle: 97     Offset: 80     phase : [59    28     ]


308 seconds run time

C:\Users\royce\source\repos\tryDISCO_GA_MK\x64\Release\tryDISCO_GA_MK.exe (process 20404) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when d
ebugging stops.
Press any key to close this window . . .
```

SGA Example

```cpp
// --- --- --- --- --- --- --- --- --- --- building up unfreeze_list for SGA --- --- --- --- --- ---
std::vector <std::set <int >> This_unfreeze_list = { {0} ,{2 }, {10} ,{ 4}, {5}, {13}, {9}, {11} };
GA_Obj.set_unfreeze_list(This_unfreeze_list);
```

It does shows the drop of average delay within a sub generation while optimizing intersection id 9

```
//--- --- --- --- --- --- --- 9th generation --- --- --- --- --- --- --- //

//~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ 1th sub generation ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ //
->> the top 8 chromosome are printed out.
0th average daley: 65.6427
 intersection_id: 0     Cycle: 118    Offset: 0     phase : [29    50    24          ]
 intersection_id: 2     Cycle: 118    Offset: 0     phase : [58    50    ]
 intersection_id: 10    Cycle: 118    Offset: 0     phase : [33    18    18    28    ]
 intersection_id: 4     Cycle: 118    Offset: 0     phase : [46    32    19    ]
 intersection_id: 5     Cycle: 118    Offset: 84    phase : [27    81    ]
 intersection_id: 13    Cycle: 118    Offset: 90    phase : [32    13    19    30    ]
 intersection_id: 9     Cycle: 118    Offset: 0     phase : [25    25    24    24    ]
 intersection_id: 11    Cycle: 118    Offset: 70    phase : [83    25    ]

1th average daley: 67.1829
 intersection_id: 0     Cycle: 118    Offset: 0     phase : [26    53    24          ]
 intersection_id: 2     Cycle: 118    Offset: 0     phase : [58    50    ]
 intersection_id: 10    Cycle: 118    Offset: 0     phase : [33    18    18    28    ]
 intersection_id: 4     Cycle: 118    Offset: 0     phase : [46    32    19    ]
 intersection_id: 5     Cycle: 118    Offset: 84    phase : [27    81    ]
 intersection_id: 13    Cycle: 118    Offset: 90    phase : [32    13    19    30    ]
 intersection_id: 9     Cycle: 118    Offset: 0     phase : [25    25    24    24    ]
 intersection_id: 11    Cycle: 118    Offset: 70    phase : [83    25    ]

2th average daley: 117.736
 intersection_id: 0     Cycle: 118    Offset: 13    phase : [22    55    26          ]
 intersection_id: 2     Cycle: 118    Offset: 0     phase : [58    50    ]
 intersection_id: 10    Cycle: 118    Offset: 0     phase : [33    18    18    28    ]
 intersection_id: 4     Cycle: 118    Offset: 0     phase : [46    32    19    ]
 intersection_id: 5     Cycle: 118    Offset: 84    phase : [27    81    ]
 intersection_id: 13    Cycle: 118    Offset: 90    phase : [32    13    19    30    ]
 intersection_id: 9     Cycle: 118    Offset: 0     phase : [25    25    24    24    ]
 intersection_id: 11    Cycle: 118    Offset: 70    phase : [83    25    ]

3th average daley: 355.522
 intersection_id: 0     Cycle: 118    Offset: 51    phase : [24    33    46          ]
 intersection_id: 2     Cycle: 118    Offset: 0     phase : [58    50    ]
 intersection_id: 10    Cycle: 118    Offset: 0     phase : [33    18    18    28    ]
 intersection_id: 4     Cycle: 118    Offset: 0     phase : [46    32    19    ]
 intersection_id: 5     Cycle: 118    Offset: 84    phase : [27    81    ]
 intersection_id: 13    Cycle: 118    Offset: 90    phase : [32    13    19    30    ]
 intersection_id: 9     Cycle: 118    Offset: 0     phase : [25    25    24    24    ]
 intersection_id: 11    Cycle: 118    Offset: 70    phase : [83    25    ]
```

```
4th average daley: 383.663
 intersection_id: 0     Cycle: 118    Offset: 67    phase : [12    34    57          ]
 intersection_id: 2     Cycle: 118    Offset: 0     phase : [58    50    ]
 intersection_id: 10    Cycle: 118    Offset: 0     phase : [33    18    18    28    ]
 intersection_id: 4     Cycle: 118    Offset: 0     phase : [46    32    19    ]
 intersection_id: 5     Cycle: 118    Offset: 84    phase : [27    81    ]
 intersection_id: 13    Cycle: 118    Offset: 90    phase : [32    13    19    30    ]
 intersection_id: 9     Cycle: 118    Offset: 0     phase : [25    25    24    24    ]
 intersection_id: 11    Cycle: 118    Offset: 70    phase : [83    25    ]

5th average daley: 473.057
 intersection_id: 0     Cycle: 118    Offset: 24    phase : [36    35    32          ]
 intersection_id: 2     Cycle: 118    Offset: 0     phase : [58    50    ]
 intersection_id: 10    Cycle: 118    Offset: 0     phase : [33    18    18    28    ]
 intersection_id: 4     Cycle: 118    Offset: 0     phase : [46    32    19    ]
 intersection_id: 5     Cycle: 118    Offset: 84    phase : [27    81    ]
 intersection_id: 13    Cycle: 118    Offset: 90    phase : [32    13    19    30    ]
 intersection_id: 9     Cycle: 118    Offset: 0     phase : [25    25    24    24    ]
 intersection_id: 11    Cycle: 118    Offset: 70    phase : [83    25    ]

6th average daley: 608.561
 intersection_id: 0     Cycle: 118    Offset: 2     phase : [20    37    46          ]
 intersection_id: 2     Cycle: 118    Offset: 0     phase : [58    50    ]
 intersection_id: 10    Cycle: 118    Offset: 0     phase : [33    18    18    28    ]
 intersection_id: 4     Cycle: 118    Offset: 0     phase : [46    32    19    ]
 intersection_id: 5     Cycle: 118    Offset: 84    phase : [27    81    ]
 intersection_id: 13    Cycle: 118    Offset: 90    phase : [32    13    19    30    ]
 intersection_id: 9     Cycle: 118    Offset: 0     phase : [25    25    24    24    ]
 intersection_id: 11    Cycle: 118    Offset: 70    phase : [83    25    ]

7th average daley: 766.621
 intersection_id: 0     Cycle: 118    Offset: 43    phase : [14    26    63          ]
 intersection_id: 2     Cycle: 118    Offset: 0     phase : [58    50    ]
 intersection_id: 10    Cycle: 118    Offset: 0     phase : [33    18    18    28    ]
 intersection_id: 4     Cycle: 118    Offset: 0     phase : [46    32    19    ]
 intersection_id: 5     Cycle: 118    Offset: 84    phase : [27    81    ]
 intersection_id: 13    Cycle: 118    Offset: 90    phase : [32    13    19    30    ]
 intersection_id: 9     Cycle: 118    Offset: 0     phase : [25    25    24    24    ]
 intersection_id: 11    Cycle: 118    Offset: 70    phase : [83    25    ]
```

SGA Example

```
 //--- --- --- --- --- --- --- 9th generation --- --- --- --- --- --- --- //

 //~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ 1th sub generation ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ ~~~ //
->> the top 8 chromosome are printed out.
0th average daley: 65.6427
 intersection_id: 0      Cycle: 118      Offset: 0       phase : [29     50      24      ]
 intersection_id: 2      Cycle: 118      Offset: 0       phase : [58     50      ]
 intersection_id: 10     Cycle: 118      Offset: 0       phase : [33     18      18      28      ]
 intersection_id: 4      Cycle: 118      Offset: 0       phase : [46     32      19      ]
 intersection_id: 5      Cycle: 118      Offset: 84      phase : [27     81      ]
 intersection_id: 13     Cycle: 118      Offset: 90      phase : [32     13      19      30      ]
 intersection_id: 9      Cycle: 118      Offset: 0       phase : [25     25      24      24      ]
 intersection_id: 11     Cycle: 118      Offset: 70      phase : [83     25      ]
```

More testing strategies

1.) Element by element NGA(random)

Every chromosome element is selected in random, could be resected or not even bother with crossover and mutation

O

2.) Element by element NGA(forced)

Forced every chromosome element is selected once to under go crossover and mutation

?

3.) Whole gene NGA

Combined all chromosome element into one gene then do crossover and mutation

O

Its not workable because if that of intersection is so congest, then the optimized cycle time would be so large in comparison to the others intersection, then this would become the best result population gene, the following generation would base on this large cycle to do crossover and mutation. This make the result bad

4.) SGA by intersection (S-> Sequential)

Each generation only focus changes on 1 intersection only and so on for the next generation and next intersection, till finished the loop. In each intersection loop may have sub iteration

?

Now do it like this,
Given intersections in order “1,2,3,4,5,6,7,8,9”; intersection group to be 3 intersection.
Then the first generation “1,2,3” would be the first 3 intersections, crossover and mutation would only undertake in this group. It would keep looping for the certain sub generation number
Next generation “2,3,4”
Next generation “3,4,5” …
Next generation “9,1,2” … until it reach the total generation number

5.) SGA by intersection group

Each generation only focus changes on intersection group (consist of several intersections) only and so on for the next generation and next intersection groups, till finished the loop. In each intersection loop may have sub iteration

?

Thank You