Swarm intelligence for traffic light scheduling: Application to downtown Area in Los Angeles

Zhiao Zhou
Center for Urban Science and Progress
New York University
Email: zz1749@nyu.edu

Hongkai He
Center for Urban Science and Progress
New York University
Email: hh1827@nyu.edu

Chenrui Zhang
Center for Urban Science and Progress
New York University
Email: cz1605@nyu.edu

1. Introduction

Nowadays, most cities around the world are suffering from congestion, security, parking, pollution and many other problems derived from an excessive vehicular traffic. Finding an better principle for scheduling traffic lights is commonly considered to be a helpful way to reduce those problems by improving the flow of vehicles through the cities when most city planning departments are not willing to make significant improvement on urban area infrastructure due to many reasons. However, especially in urban area of large cities like New York and Los Angeles, thousands of new traffic light has emerged with the urban's development, making traffic lights schedule more complicated dramatically because of a great number of new combinations. In this situation, applying an optimal algorithm for scheduling traffic lights is a necessary choice.

Among several evolutionary methods, optimization strategy based on particle swarm optimization(PSO) has proved its usefulness to the optimization of cycle programs of traffic lights. One related research proposed a swarm intelligence approach to find successful cycle programs of traffic lights[1] . Using a microscopic traffic simulator, the solutions obtained by PSO are evaluated in the context of two large and heterogeneous metropolitan areas located in the cities of Malaga and Sevilla in Spain which typically represents regular and irregular road

structure respectively. The simulation result shows that, for both regular and irregular road structure, significant profits are obtains in terms of two main indicators: the number of vehicles that reach their destinations on time and the global trip time. However, a semi-regular road structure area(combination of both regular and irregular road) has not been considered, like Los Angeles which represents most large cities. The purpose of this paper is to test the robustness of PSO in terms of providing the optimal traffic lights cycle programs by implementing the optimization and simulation processes elaborated in the paper on a central area of downtown in Los Angeles. After running the algorithm for 10 iterations (1000 global steps), our objective function -- the number of cars completing their trips has increased by 22.3% to 252 from 206 at step 0. However, we found that PSO is too slow to conduct in real-world problems and the states of the vehicles could be more complex and more dynamic than we set in this paper. Consequently, more algorithms should be tried and compared with PSO in future works.

2. Literature Review

There are different approaches to solve traffic light staging problems. Much effort has been made in the sense that the “real” time impact of the traffic cycle duration on the traffic network is considered for adaptive traffic lights, mainly concerning the use of detectors to sense the traffic and to change the duration of cycle programs, taking into account the actual flow of vehicles.[2]

An adaptive traffic control model of signal lights is introduced by Bretherton et. al [3] consisting on the SCOOT (Split Cycle Offset Optimisation Technique) platform. SCOOT is an adaptive system used to manage and control traffic signals in urban areas, automatically responding to fluctuations in traffic flow by on-street detectors embedded in the road. SCOOT can always solves problems if the traffic patterns are unpredictable.

Employed mathematical techniques has also been used. For example, McCrea et al. [4] combined continuous calculus based models and knowledge based models in order to describe the traffic flow in road networks. Tolba et al. [5] introduced a Petri Net based model to represent the traffic flow, from a macroscopic viewpoint (where only global variables are observed) and from a microscopic one (where the individual trajectories of vehicles are considered).

Recently, biologically inspired techniques such as Cellular Automata (CA) and Neural Networks (NN) have been used for tackling the underlying combinatorial optimization problems,and in particular for solving traffic light staging problems. Brockfeld et al. [6] applied a CA model in which the city network was implemented as a simple square with a few normal streets and four intersections. In this approach, the vehicles needed an additional module for the data management.

Related to the biologically inspired techniques, metaheuristic algorithms [7] have become very popular for solving traffic light staging problems. A first attempt was made by Rouphail et al. [8], where a Genetic Algorithm (GA) was coupled with the CORSIM [9] microsimulator for the timing optimization of nine intersections in the city of Chicago (USA). The results, in terms of total queue size, were limited due to the delayed convergence behavior of the GA.

A few publications related to the application of PSO for the schedule of traffic lights also exist. One of the most representative was developed by Chen and Xu [10], where they applied a PSO for training a fuzzy logic controller located at each intersection by determining the effective time of green for each phase of the traffic lights. A very simple network with two basic junctions was used for testing this PSO. Another representative was developed by J.Garcia-Nieto, E.Alba and A.Carolina Olivera who proposed a swarm intelligence approach to find successful cycle programs of traffic lights. Using a microscopic traffic simulator, the solutions obtained by PSO are evaluated in the context of two large and heterogeneous metropolitan areas located in the cities of Malaga and Sevilla in Spain which typically represents regular and irregular road structure respectively. The simulation result shows that, for both regular and irregular road structure, significant profits are obtains in terms of two main indicators: the number of vehicles that reach their destinations on time and the global trip time.

We tried to use PSO algorithm and SUMO simulation software in this research since they are able to help us deal with real world data and to find optimized cycle programs for scenarios with a great number of traffic lights, vehicles, and other elements. Additionally, we set downtown Los Angeles as our area to be evaluated as downtown Los Angeles is comprised of both of regular and irregular traffic structures.

3. Problem Formulation

In this project, our aim is to test PSO on optimizing traffic light management in Downtown Los Angeles. To solve this problem, we should divide the problem into several subproblems:

1. We need to decide which part of downtown Los Angeles where we are going to test the model as OpenStreetMap from which we generate our map XML file from does not support exporting XML file of a large area.
2. Scratch a Python script for PSO algorithm where we need to choose what the objective function(fitness function) should be. An objective function is aimed to measure the performance of an iteration such as the lowest time when all vehicles reach the destination.
3. We need to decide the number of the particles and iteration numbers.
4. We do not know if our script would work properly so we need to first test it on a cropped area of our original map.

5. Based on the result above, modify the script if needed and test it on the original downtown Los Angeles.

4.Data and Methodology

4.1 Data source

The final file tree of our repository is:

```
.
├──LA.sumo.cfg
├──LA2.rou.alt.xml
├──LA2.rou.xml
├──LAedited.net.xml
├──LATripInfo.xml
├──randomTrips.py
├──trips.trips.xml
├──main.py
├──viewsettings.xml
```

First, a SUMO simulation requires an net.xml file of an specific area. In this paper, we first obtained an OMS file of an area in downtown Los Angeles from OpenStreetMap which contains 309 junction IDs ,as shown in Figure 1 (there's a limit area that can be exported in OpenStreetMap).

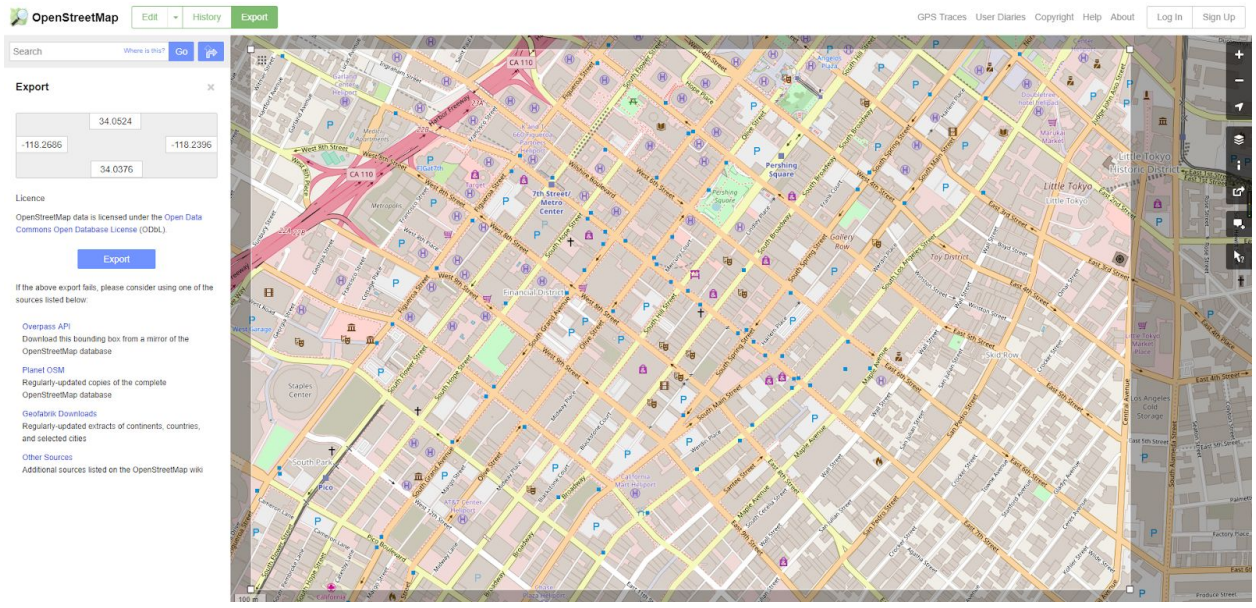

Figure 1. The selected area in downtown Los Angeles

And then we transferred it to an XML file named "LAedited.net.xml" using SUMO's built-in netconvert function which contains the complete network in this area including roads, traffic

lights, sidewalks, highways and so on as shown in Figure 2. Additionally, we need a rou.xml file indicating the start point, path and destination point of every vehicle, which we generated from SUMO's built-in randomTrips.py script and from that we obtained LA2.rou.xml, LA2.rou.alt.xml and trips.trips.xml files. What's more, "LATripInfo.xml" is generated by every simulation which contains the phase duration and states of all of the traffic lights in the area and "viewsettings.xml" which is the display format for a simulation in SUMO-GUI. Last but not least, the LA.sumo.cfg file is the sumo configuration file where we can glue everything together.

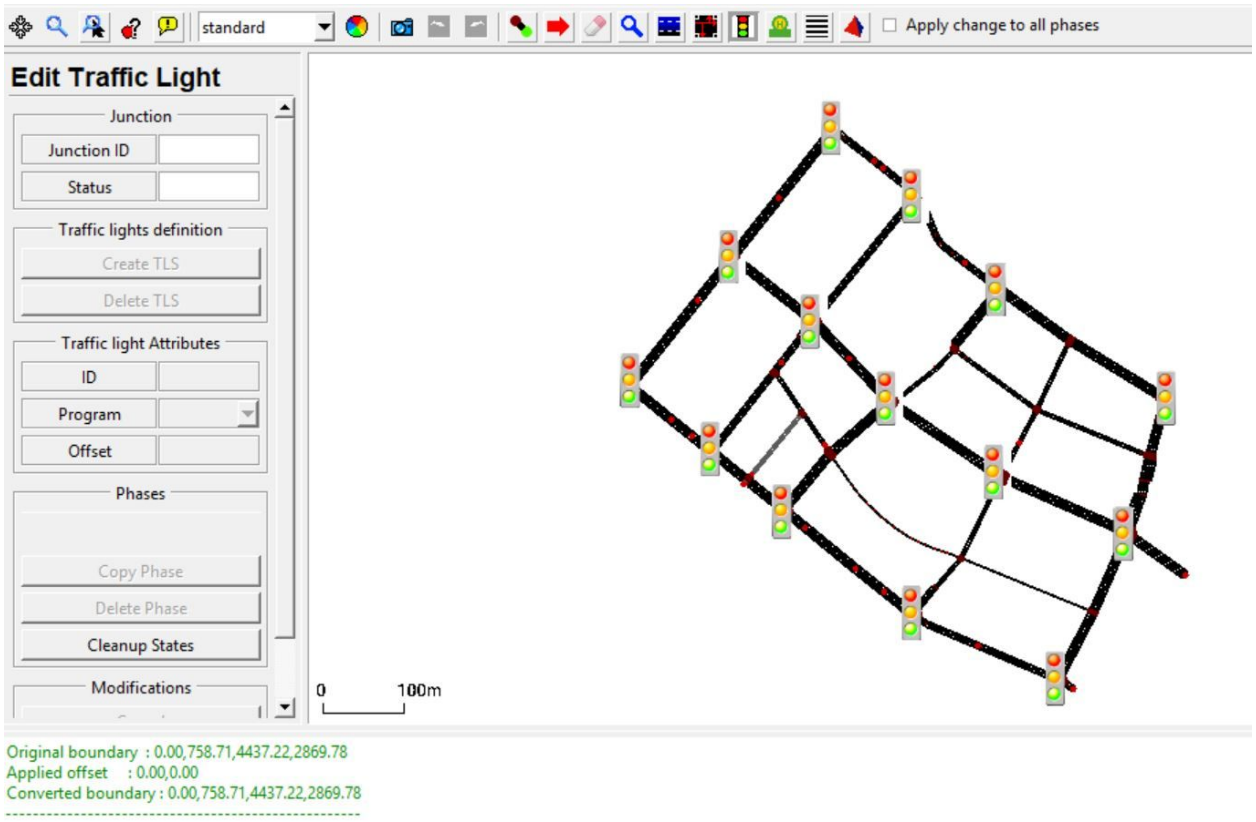

Figure 2. Road structures in SUMO

4.2 Particle swarm optimization

Particle-Swarm Optimization (PSO) is an evolutionary algorithm developed by J. Kennedy and R. C. Eberhart et. al. with the inspiration from the collective behavior of simple animal societies, such as bee swarms and bird flocks. The fundamental idea is to find the optimal global solution of a problem facing the group, by virtue of collaboration and information sharing among individual members. It is helpful to use an idealized scenario of a bird flock foraging for food to facilitate the explanation of PSO.

Suppose a group of flying bird is looking for food in a given area. There is only one worm in the area and no bird knows the exact location of the worm. However, every bird knows the distance

from itself to the prey and the location of the bird that is nearest to the prey (the global best) at every moment. Hence if bird alpha claims that it is the nearest to the prey, then bird beta, gama, delta……will adjust their flying trajectories to approach the location of bird alpha. In the meantime, as the birds locations are constantly changing, every bird will also make reference to its own historical records of nearest location to food. Apart from these factors, the movement of a bird is also subject to its inertia. Let i represent the ith bird (particle) in the flock (swarm), X as the position of a single bird, V as the velocity or increment of position of a single bird, g as the gth adjustment (iteration) of the position of the whole flock, then we obtain the following two formula to describe the movement of an individual bird:

$$x_{g+1}^{i} \leftarrow x_{g}^{i} + v_{g+1}^{i}$$

$$v_{g+1}^{i} \leftarrow w \cdot v_{g}^{i} + \varphi_1 \cdot UN(0,1) \cdot (p_{g}^{i} - x_{g}^{i}) + \varphi_2 \cdot UN(0,1) \cdot (b_{g} - x_{g}^{i}).$$

Constant adjustment of the trajectories and positions f each bird will result in the approximation of the flock as a whole towards the food, until the birds capture the prey, ie., objective function is optimized.

4.3 Implementation of PSO Algorithm

In the original paper the author used MALLBA, a C++ based framework of metaheuristic algorithms for solving optimization problems. We have made many attempts to install this package but none of them succeeded. In addition, given that all three group members' coding proficiency lies in Python, we decide to turn to Python for the implementation of PSO algorithm. With reference to a published Python code on Github, we create our own implementation of PSO in Python. We actualize the important steps in the PSO algorithm by defining modular python functions.

The first important function initiates the very first swarm. Each particle is defined as a list of numbers, each of which represent the phase duration of a single traffic light. The length of the list equals to the number of traffic lights in the selected road structure, which in this case is 309. The swarm size chosen is 100, which means there are 100 such list of phase durations. A sample particle is demonstrated below. Please note that the key for each list of phase durations corresponds to an intersection ID.

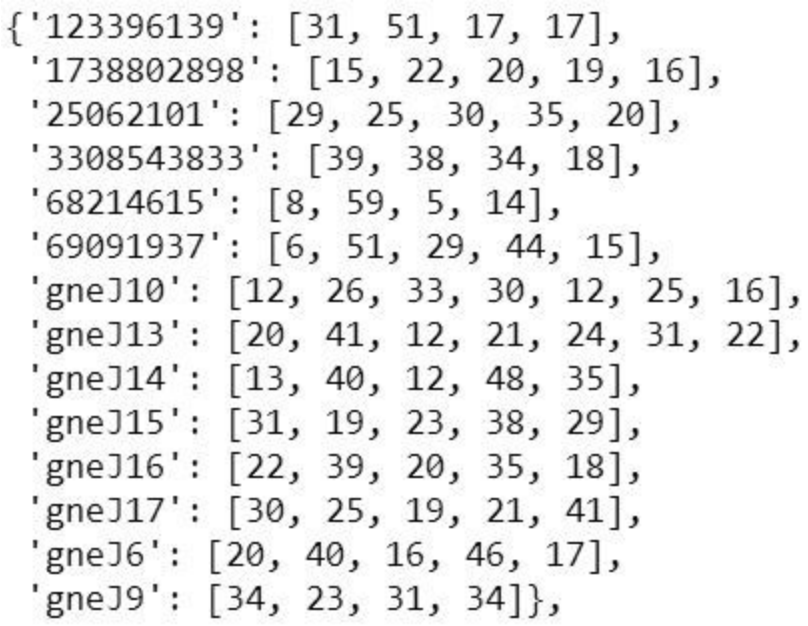

Figure 3 : A sample particle that contains a list of phase durations for each intersection.

Each particle can be seen as a scenario where the sumo simulation implement these color phase durations at every junction and the number of cars that complete their trips is returned as the result we want to optimize. The swarm size we choose is 100, which means there are 100 such particles, or 100 scenarios. For each iteration, we implement each scenario and obtain 100 results. The largest result will be chosen as the global best result in this iteration since we would like the number of cars completing their trips to be as large as possible. This global best result and its corresponding particle will be stored and be used to update the phase durations to generate new particles for next iteration.

The velocity, , i.e., the incremental change of particles, are generated by the initial velocity function. The results returned from the function is of similar format to those returned from swarm function which is key value pairs with junction ID as keys and list of velocities as values. Then a function named update_particle is created to add the incremental change (velocity) to the particle according to the velocity formula and particle update formula explained in previous text. In this experiment, we set inertia weight to be 0.5 while global best solution weight and local best solution weight to be random values drawn from a uniform distribution between 0 and 1.

Moving to the next step another function is defined to update the network file in terms of traffic lights cycle programs according to the particles in the swarm and particle velocity.

Fitness value, i.e. the number of cars completing their trips for implementing each particle is obtained by the following function. Instead of using the complicated fitness function used in the

original paper which incorporates multiple variables, due to limitation of computational power of our machines, in our experiment we use the number of cars that completes their trips to be the single target variable to be optimized (maximized) by PSO.

```python
def get_fitness(outputfile):
    tree = ET.parse(outputfile)
    """
        Takes in tripinfovalue file as input to calculate fitness.
        Returns a tuple (tot_time, no of vehicles which reached destination within sim time)
        INPUT: sumo file containing trip details after a sim has ended.
        OUTPUT: (tot_traveltime, no of vehicles completing the trip)
    """
    root =tree.getroot()
    tot_time =0
    for i in root:
        tot_time +=float(i.attrib["duration"])
    return (tot_time, len(root))
```

Figure 4: python function that calculate fitness value.

The Traffic Control Interface (TraCI) packaged with SUMO is called to run simulation. The simulation time is set to be 500s and there are 5000 time steps in each simulation, with each step interval to be 0.1s.

After creating all the aforementioned functions, we write a final summary function that calls all these functions to implement the whole procedure of this experiment. For detailed code implementation please refer to our github link provided in reference section.

4.4 Architecture

The architecture of this project, along with sub-structure of each steps could be organized as a flow chart shown in Figure 5.

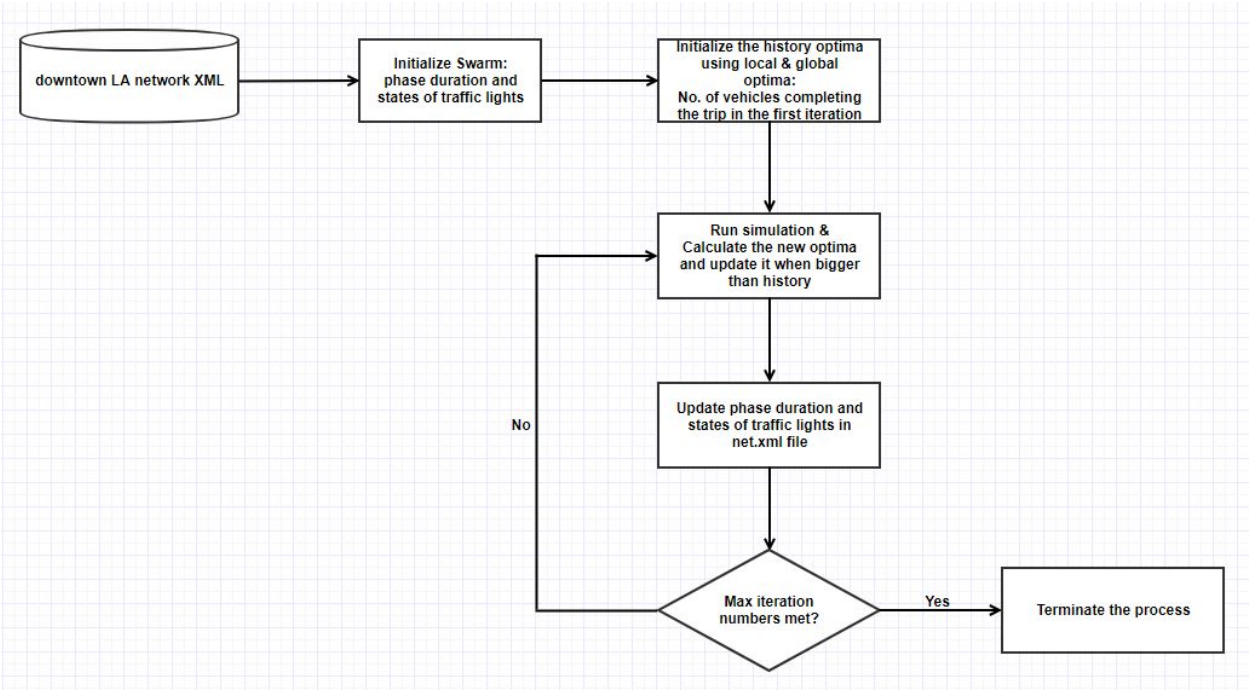

Figure 5: The full PSO structure

4.5 Link to the Code

The guthub link to our Python code is:
https://github.com/zhiaozhou/Traffic_Light_Control_PSO

5.Results & Discussion

With the number of particles = 100 and the number of iterations = 10, the number of global steps should be 1000. At step one we can see in Figure 6 that there are quite a few traffic jams in some intersections which looks like a mess, resulting in a objective function of only 256 which means only 256 cars have reached their destinations. After 10 iterations (1000 global steps), the objective function has increased by over 8% as shown in Figure 7, resulting in the No. of cars arriving of 277 at the final step. In addition, we can see that as iteration number grows, the performance is also improving steadily.

Accordingly, it can be concluded that the PSO algorithm is mighty effective on optimizing traffic light management. And it's very clear and interpretable as outputs at every step can be printed and checked. However, there are still plenty of disadvantages we found in this research. First, running PSO on CPU ,which is commonly used nowadays, is very slow. It took us more than 10 hours to run 10 iterations on the downtown Los Angeles area. What's more, when addressing real-world situations, the states of the vehicles could be more complex than in theory. For example, in New York City, drivers will drive at over 40 miles/hr on some wide streets while the

speed limit is only 25 mile/hr. So the configuration should be more carefully set in a real-world situation.

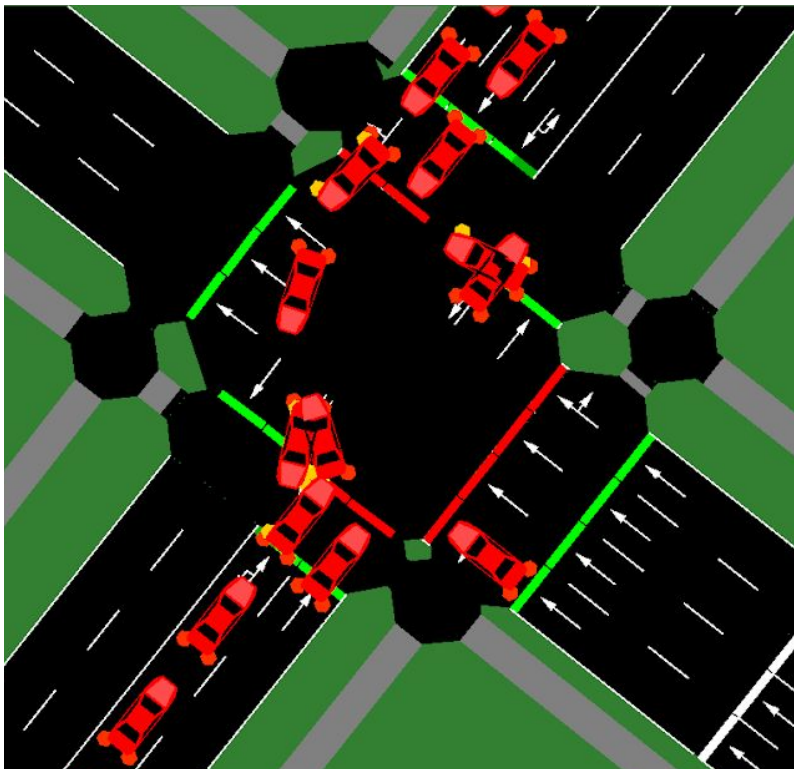

Figure 6. Traffic jams at step 0

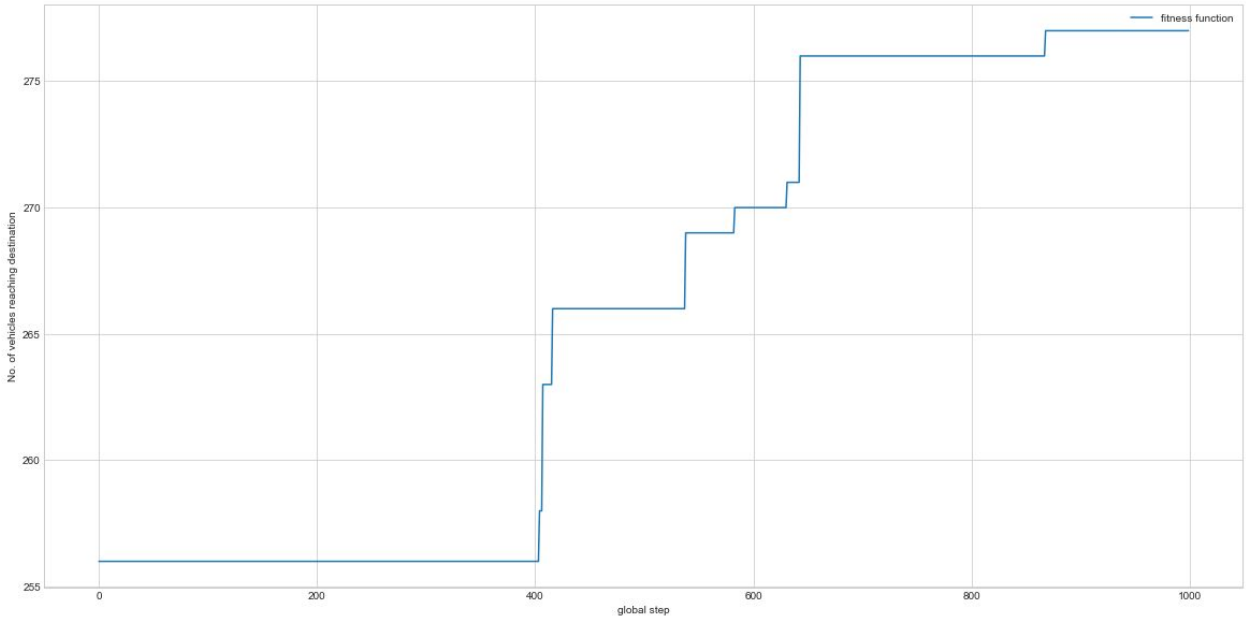

Figure 7. Objective function by global steps

6. Conclusion

From above, we can see that PSO is good at optimizing traffic light management on both regular and irregular road structures. Nevertheless in the future, we should try more state-of-art algorithms such as GPU-PSO or Neural Networks in that PSO is too slow which is not practical when carried out on real life occasions.

References

[1] J.Garcia-Nieto, E.Alba, A.Carolina Olivera(2011). Swarm intelligence for traffic light scheduling:Application to real urban areas. Engineering Applications of Artificial Intelligence 25 (2012) 274-283.

[2] Jose Garcia-Nieto, Ana Carolina Olivera, and Enrique Alba(2013). Optimal cycle program of traffic lights with particle swarm optimization. IEEE Transactions on Evolutionary Computation (Volume: 17, Issue: 6, Dec. 2013)

[3] R. Bretherton, N. Hounsell, and B. Radia(1996). Public transport priority in scoot. Available: http://eprints.soton.ac.uk/75299/

[4] J.McCrea and S.Moutari(2010). A hybrid macroscopic-based model for traffic flow in road networks. European Journal of Operational Research, vol. 207, no. 1, pp. 676–684, 2010.

[5] C.Tolba, D.Lefebvre, P.Thomas, and A.E.Moudni(2005). Continuous and timed petri nets for the macroscopic and microscopic traffic flow modelling. Simulation Modelling Practice and Theory, vol. 13, no. 5, pp. 407 – 436, 2005.

[6] E.Brockfeld, R.Barlovic, A.Schadschneider, and M.Schreckenberg(2001). Optimizing traffic lights in a cellular automaton model for city traffic. Phys. Rev. E, vol. 64, no. 5, p. 056132, Oct 2001.

[7] C.Blum and A.Roli(2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys (CSUR), vol. 35, no. 3, pp. 268–308, 2003.

[8] N.M.Rouphail, B.B.Park, and J.Sacks(2000). Direct signal timing optimization: Strategy development and results. XI Pan American Conference in Traffic and Transportation Engineering, Tech. Rep., 2000.

[9] P. Holm, D.Tomich, J.Sloboden, and C.Lowrance(2007). Traffic analysis toolbox volume iv: Guidelines for applying corsim microsimulation modeling software. Nat.Tech. Information Service - 5285 Port Royal Road Springfield, VA 22161 USA - Final Report, Tech. Rep., 2007.

[10] J. Chen and L. Xu(2006). Road-junction traffic signal timing optimization by an adaptive particle swarm algorithm. ICARCV, 2006, pp. 1–7.