Competition on Urban Traffic Signal Control Problems

Kaizhou Gao[1] and Mohammed El-Abd[2]

[1]*Macau Institute of Systems Engineering, Macau University of Science and Technology*

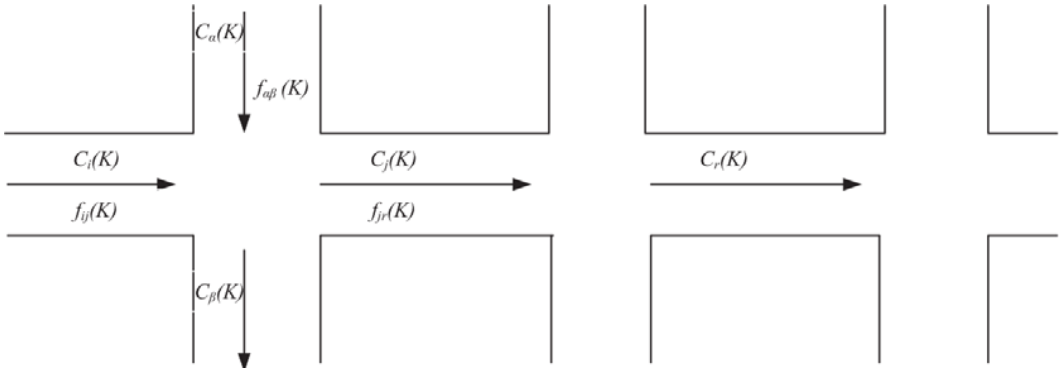
[2]*College of Engineering and Applied Sciences, American University of Kuwait*

Introduction

According to the recent study in [1], one clear aspect that is lacking in the domain of Traffic Signal Control (TSC) is the availability of common libraries for benchmarking and comparing different algorithms. Metaheuristics have been employed in the domain of TSC [2-6]. A vast majority of publications apply proposed algorithms on their own generated artificial networks and/or networks from the authors' cities/countries without comparing with previous work. This poses a huge challenge in identifying the state-of-the-art in TSC. A comprehensive literature review was presented in [1] in chronological order and rarely it was found that authors provide comparisons with existing works. Having benchmark libraries containing networks of different sizes and under different conditions (undersaturated, saturated, and oversaturated) would ensure that a proposed algorithm is properly evaluated and would provide the research community with a baseline against which algorithms could be fairly evaluated.

Problem Model

A traffic network consists of a set of links and junctions. For instance, a simple unidirectional traffic network is depicted in **Fig. 1**, where each junction has only two antagonistic traffic flows. A discrete time model has been proposed based on the cell transmission. To simplify our technical discussions, some key notations in urban traffic signal scheduling problem formulation are listed as given in **Table 1**.



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)

Fig. 1. A simple schematic view of a traffic network.

Table 1 Terminology used in the traffic signal scheduling problem.

Notation	Definition
$C_i(k)$	Number of vehicles in link i in the time interval k.
$f(k)$	Exit flow rate from link i to link j in interval k.
${}^{ij}N$	The count of sampling intervals in one prediction horizon.
Δ	Time interval.
L	Set of all one-way links.
J	The set of intersections.
ω	The stage in intersection.
Ω_J	Set of the stages of all intersections.
F_J	$F_J \subseteq L \times L$, the set of all streams in intersection J, i.e., $(i, j) \in F_J$ means that there exists a traffic stream from link i to link j via intersection J.
$h_J : \Omega_J \to 2^{F_J}$	Association of each stage to relevant compatible streams.

The following assumptions about a traffic network are made to allow deterministic analysis:

- Network demands, entrance and exit models are known and there is an infinite queue for each entrance demand.
- The link turning ratios in the network are known.
- Each vehicle inside the network leaves the network after a delay partially caused by traffic signals.

Library Description

This competition provides the first attempt to introduce instances of different sizes to be tackled using metaheuristic algorithms. The library includes networks ranging in size from 3x3 (9 intersections) to 20x20 (400 intersections).

Traffic signals coordinate the traffic movements at the intersections and a smart traffic signal coordination algorithm is the key to transportation efficiency. For a four-leg intersection (see **Fig. 2**) [5], 1 of 4 types of signal phases can be selected in each time step, serving a pair of non-conflict traffic movements (e.g., phase-3 gives right-of-way for left-turn traffic from northern and southern approaches). Participants need to develop a model to select traffic signal phases at intersections of a road network to improve road network traffic performance.

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)

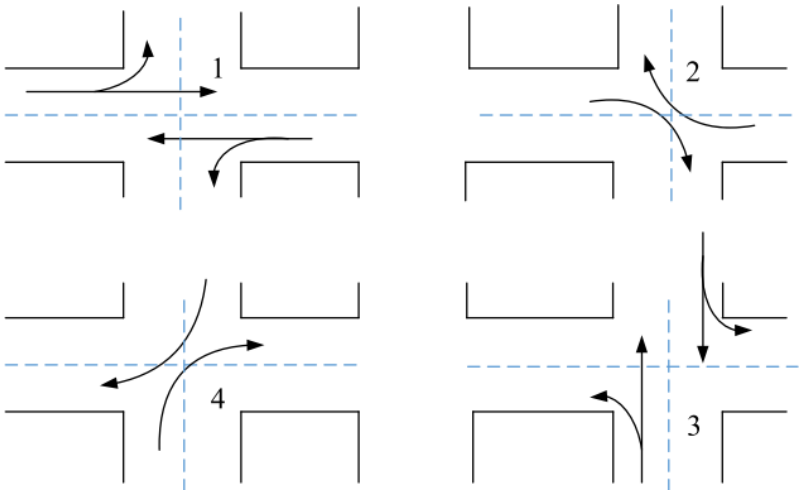


Fig. 2. Four phases considered in a junction.

In the final phase, a city-scale road network sample traffic data is provided. We use exactly the same road network but different scale (from 3-3 to 20-20) traffic data for scoring your submissions.

Evaluation

The objective is to minimize the total network-wise delay time of all vehicles within N time intervals. The objective function can be given as follows:

$$Min\sum_{i\in L}\sum_{k=1}^{N}(C_i(k)-\frac{L_i}{v_{i,\max}}\sum_{j\in L:(i,j)\in\cup_{J\in J}F_J}f_{ij}(k))\Delta \qquad (1)$$

where N is the count of sampling intervals in one prediction horizon, where $C_i(k)$ is the link volume. L_i is the length of link i, $f_{ij}(k)$ is the exit flow rate from link i to link j in the time interval k. Δ is the time interval. $v_{i,\max}$ is the maximum speed in link i.

Experimental setup

In all test cases, a time window of 60s is used. The sampling period of 15s is adopted for the experiments. The scales of considered cases (Case 1 to Case 18) are from 9 to 400 junctions. The proposed model has been coded in C++. In the sample code, the population size and maximum iterations number are set to 50 and 1000, respectively.

All experiments are to be repeated for 30 independent runs with a maximum number of function evaluations of 50,000 per a single run. Participants are required to obtain the maximum value, average value and minimum value of the objective function obtained over the 30 runs.

Code Explanation

```
const int MaxRowNum = 50;// The maximum number of row links.
const int MaxColumnNum = 50;// The maximum number of column links.
const int Max_Sched_Time = 60; // The maximum sampling interval is 60s.
```

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)

```
const int Min_Win = 5; // The minimum time window in each scheduling.
const int PopSize = 50;    // The size of population.
int Gen; //The number of iterations of the algorithm.
int Repeat;// The repetitions of the experiments.
int Sched_Time; // The time of each sampling interval in the experiments.
int Win;// The time window in each scheduling in the experiments.
int RowNum;  // The number of row links.
int ColumnNum; //The number of column links.
int C_Upper_Row[MaxRowNum][MaxColumnNum]; // The upper capacity value of row direction (left to right).
int C_Upper_Column[MaxColumnNum][MaxRowNum]; // The upper capacity value of column direction (above to below).
int C_Row_Initial[MaxRowNum][MaxColumnNum]; // The Initial capacity value of row direction (left to right).
int C_Column_Initial[MaxColumnNum][MaxRowNum]; // The Initial capacity value of column direction (above to below).
int C_Row[MaxRowNum][MaxColumnNum]; // The capacity value of row direction (left to right).
int C_Column[MaxColumnNum][MaxRowNum]; // The capacity value of row direction (above to below).
int C_Upper_Row1[MaxRowNum][MaxColumnNum]; // The upper capacity value of row direction (right to left).
int C_Upper_Column1[MaxColumnNum][MaxRowNum]; // The upper capacity value of column direction (below to above).
int C_Row1_Initial[MaxRowNum][MaxColumnNum]; // The Initial capacity value of row direction (right to left).
int C_Column1_Initial[MaxColumnNum][MaxRowNum]; // The Initial capacity value of column direction (below to above).
int C_Row1[MaxRowNum][MaxColumnNum]; // The capacity value of row direction (right to left).
int C_Column1[MaxColumnNum][MaxRowNum];// The capacity value of row direction (below to above).
int f_Row[MaxRowNum][MaxColumnNum];// The flowing rate in row direction (left to right).
int f_Row1[MaxRowNum][MaxColumnNum]; // The  flowing rate in row direction (right to left).
int f_Column[MaxColumnNum][MaxRowNum];// The  flowing rate column direction  (above to below).
int f_Column1[MaxColumnNum][MaxRowNum]; //The flowing rate in column direction (below to above).
int Pop[PopSize][Max_Sched_Time / Min_Win][MaxRowNum][MaxColumnNum]; //The population (The phase chosen by each individual at each sampling moment)
int New_Pop[PopSize][Max_Sched_Time / Min_Win][MaxRowNum][MaxColumnNum];// Generateing new population
int Best_Pop[Max_Sched_Time / Min_Win][MaxRowNum][MaxColumnNum]; // The best solution
int New_Car_Row_Initial[Max_Sched_Time / Min_Win][MaxRowNum];    // Generate new vehicles in a row direction (right to left) at each sampling time
int New_Car_Column_Initial[Max_Sched_Time / Min_Win][MaxColumnNum];// Generate new vehicles in a column direction (above to below) at each sampling time
int New_Car_Row1_Initial[Max_Sched_Time / Min_Win][MaxRowNum];// Generate new vehicles in a row direction (left to right) at each sampling time
int New_Car_Column1_Initial[Max_Sched_Time / Min_Win][MaxColumnNum];// Generate new vehicles in a column direction (below to above) at each sampling time
int Obj[PopSize]; // The objective function value for each current individual.
int New_Obj; // Record the leaving of the vehicle.
int count2;// Determine whether the calculated objective function value is the initial population or the new population
int Total_Obj[PopSize]; // The objective function value for each individual.
int K; // Record the sampling moment.
int popi; // Record the population.
int totalobj, I;// The objective function value of the best solution.
```

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)

```
int TotalNum; // total objective value
float ExitRat[MaxRowNum][MaxColumnNum][12]; //Ratio of vehicles leaving in the left, right above and below directions .
int ExitTurnNum = 2 * 2 * 3;// Different leaving directions of the vehicles.
float EntraRat[MaxRowNum][MaxColumnNum][8]; //Ratio of vehicles entering in the left, right above and below directions .
int EntraTurnNum = 2 * 2 * 2;// Different entering directions of vehicles..
int Lij[5][Max_Sched_Time / Min_Win]; // The first row is for the turn left maximum speed when there is just one lane; the second mean the straight spped for one lanes and for the left turn speed when there are muptiple lanes.
int Lanes[MaxRowNum][MaxColumnNum][2];// The first value is for left and right direction, the second value is for the above and below direction.
int L[MaxRowNum][MaxColumnNum][2];//The length of links connected to each intersection, left, right above and below.
int Len;// The average length to store each car in link.
```

Provided Files

All files explained below could be downloaded at: https://www.researchgate.net/project/2022-WCCI-CEC-Competition-on-Urban-Traffic-Signal-Control-Problems

The Initial folder contains the following:

1. Data_Record

(1) Best_Solutions

I. Case_1.csv

The best solutions of 30 repeated experiments of the 3*3 scale traffic network.

.....

Case_18.csv

The best solutions of 30 repeated experiments of the 20*20 scale traffic network.

(2) Results

I. Case_1.csv

The total delay time results of 30 repeated experiments of the 3*3 scale traffic network.

.....

Case_18.csv

The total delay time results of 30 repeated experiments of the 20*20 scale traffic network.

II. Standard_Data.xlsx

The minimum, average and maximum values of 30 repeated experiments of 18 different cases.

2. Different_Scale_Data

For every case, the traffic data provided is as follows:

- Entrance Ratio.csv

  Ratio of vehicles entering in different directions. 1 means there is a vehicle entering from this direction, while 0 means there is no car entering.
- Exit Turn Ratio.csv

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)

The turning ratio of vehicles in different directions.

- GenRepeat.csv
  The turning ratio of vehicles in different directions. Gen is the number of iterations of the algorithm. Repeat is the repetitions of the experiments.
- Instance.csv
  Traffic conditions on lanes in different directions.
- L.csv
  The first line means the length of each vehicle in the network is fixed. The length of lanes on different links in different directions.
- Lanes.csv
  Number of lanes on different links in different directions.
- Lij.csv
  The different vehicle speed category. The bigger the consecutive green light number, the higher speed category is selected.
- New Car.csv
  Generate new vehicles in different directions at each sampling time.

3. Standard_Model.cpp

Code for the basic problem model.

Example display

The Example folder provides the use of Harmony Search (HS) [] as an illustrative example.

1. Improved_model_solutions

Best Solutions of 18 different cases corresponding to the code.

I. Case_1.csv

The best solutions of 30 repeated experiments of the 3*3 scale traffic network.

.....

Case_18.csv

The best solutions of 30 repeated experiments of the 20*20 scale traffic network.

2. Improved_model.cpp

Code for the basic problem model by adding harmony search algorithm.

3. Improved_model_results_summary.xlsx

Delay time results of 18 different cases corresponding to the code. And the summary of the maximum, average and minimum delay time results of 18 different cases.

Required Submission:

Competition participants are required to submit the following:

- Improved code (after adding new algorithm)
- Best Solutions corresponding to the results of 18 cases of different scales
- The results of 18 cases of different scales corresponding to the code.

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)

    - Results of each experiment.
    - A summary of the maximum, average, and minimum values corresponding to the results of 18 different scale cases
- Similar examples can be found in the **Example** folder.

References

[1] Palwasha W. Shaikh, **Mohammed El-Abd**, Mounib Khanafer, and Kaizhou Gao. "A Review on Swarm Intelligence and Evolutionary Algorithms for Solving the Traffic Signal Control Problem". Accepted for publication in the IEEE Transactions on Intelligent Transportation Systems, vol. 23, issue 1, pp. 48-63, 2022.

[2] Kaizhou Gao, Yicheng Zhang, Rong Su, Fajun Yang, P. N. Suganthan, and MenChu Zhou, "Solving Traffic Signal Scheduling Problems in Heterogeneous Traffic Network by using Meta-Heuristics," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 9, pp. 3272 - 3282, 2018.

[3] Kaizhou Gao, Yi Zhang, Yicheng Zhang, Rong Su, and P. N. Suganthan, "Meta-Heuristics for Bi-Objective Urban Traffic Light Scheduling Problems," IEEE Transactions on Intelligent Transportation Systems, vol. 20, no. 7, pp. 2618-2629, 2019.

[4] Yi Zhang, Kaizhou Gao, Yicheng Zhang, and Rong Su. "Traffic Light Scheduling for Pedestrian-Vehicle Mixed-Flow Networks," IEEE Transactions on Intelligent Transportation Systems, vol. 20, issue 4, pp. 1468-1483, 2019.

[5] Kaizhou Gao, Yi Zhang, Ali Sadollah, Antonios Lentzakis, and Rong Su. "Jaya, Harmony Search and Water Cycle Algorithms for Solving Large-Scale Real-Life Urban Traffic Light Scheduling Problem". Swarm and Evolutionary Computation, vol. 37, pp. 58-72, 2017.

[6] Kaizhou Gao, Yicheng Zhang, Ali Sadollah, and Rong Su, "Optimizing Urban Traffic Light Scheduling Problem using Harmony Search with Ensemble of Local Search," Applied Soft Computing, vol. 48, pp. 359-372, 2016.

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)