Applied Soft Computing 48 (2016) 359–372

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Optimizing urban traffic light scheduling problem using harmony search with ensemble of local search

Kaizhou Gao*, Yicheng Zhang, Ali Sadollah, Rong Su

School of Electrical and Electronic Engineering, Nanyang Technological University, 639798, Singapore

ARTICLE INFO

Article history:
Received 20 February 2016
Received in revised form 15 July 2016
Accepted 15 July 2016
Available online 22 July 2016

Keywords:
Traffic light scheduling
Harmony search algorithm
Local search
Ensemble

ABSTRACT

This study addresses urban traffic light scheduling problem (UTLSP). A centralized model is employed to describe the urban traffic light control problem in a scheduling framework. In the proposed model, the concepts of cycles, splits, and offsets are not adopted, making UTLSP fall in the class of model-based optimization problems, where each traffic light is assigned in a real-time manner by the network controller. The objective is to minimize the network-wise total delay time in a given finite horizon. A swarm intelligent algorithm, namely discrete harmony search (DHS), is proposed to solve the UTLSP. In the DHS, a novel new solution generation strategy is proposed to improve the algorithm's performance. Three local search operators with different structures are proposed based on the feature of UTLSP to improve the performance of DHS in local space. An ensemble of local search methods is proposed to integrate different neighbourhood structures. Extensive computational experiments are carried out using the traffic data from partial traffic network in Singapore. The DHS algorithm with and without local search operators and ensemble is evaluated and tested. The comparisons and discussions verify the effectiveness of DHS algorithms with local search operators and ensemble for solving UTLSP.

1. Introduction

With the increasing of population and vehicles, the traffic congestion becomes more and more serious in urban area. Thus, how to schedule traffic lights effectively turns to be more and more important for metropolises. A reasonable traffic lights scheduling can reduce the delay time in an urban traffic network system [1–4]. An urban traffic network consists of a set of road links connecting with each other via intersections. Each intersection consists of a number of approaches and the crossing area [5,6]. An approach may have one or more lanes but has a unique, independent queue. Approaches are used by corresponding traffic streams (veh/h), which means the number of vehicles through one intersection in one hour. Two compatible streams can safely cross the intersection simultaneously, while antagonistic streams cannot. In traditional traffic signal control, a signal cycle is one repetition of the basic series of stages at an intersection, where each stage consists of simultaneous traffic light signals allowing predefined compatible traffic streams to cross the intersection simultaneously. The duration of a cycle is called cycle time [7,8]. For safety reasons, constant lost (or inter-green) times of a few seconds are necessarily inserted between consecutive stages to avoid interference between antagonistic streams. For each traffic light, the ratio of the green time and the red time within one cycle is called the split, and the delay between the starting times of green periods of two neighbouring traffic lights along the same traffic route is called offset [9,10].

There are basically four types of characteristics that distinguish different traffic signal control strategies, i.e., fixed time strategies versus traffic responsive strategies, and isolated strategies versus coordinated strategies. Notable strategies proposed in the last few decades include, e.g., MAXBAND [11,12], TRANSYT [11,12], SCOOT [13], OPAC [14], PRODYN [15], CRONOS [16], and RHODES [17]. To solve the traffic light control problem, many researchers proposed various optimization approaches, e.g., particle swarm optimization [18], distributed coordination of exploration and exploitation [19] and Mixed-integer programming [20] and so on. Among the various approaches, meta-heuristics are very flexible and robust to the problem scale and random variables, have significant advantages in computational efficiency in terms of CPU time for large-scale real-lift applications, and become the new trend for solving real-life traffic light control problems. Existing models and approaches focused on many aspects of traffic light control problems. However, there are some insufficiencies. The first one is the limited prob-

* Corresponding author.
E-mail addresses: gaokaizh@aliyun.com, kgao001@e.ntu.edu.sg (K. Gao), yzhang088@e.ntu.edu.sg (Y. Zhang), sadollah@ntu.edu.sg (A. Sadollah), rsu@ntu.edu.sg (R. Su).

http://dx.doi.org/10.1016/j.asoc.2016.07.029
1568-4946/

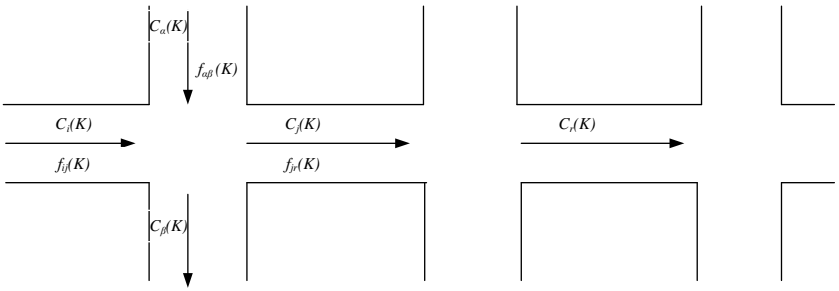

Fig. 1. Schematic view of a simple traffic network.

lem scale and constraints. Most models and approaches focused on one intersection or very limited regions. The second one is that most models and approaches have limitations in practical applications. The third one is most models are cycle programs and the light periods are fixed.

In this paper, we describe a traffic network by a dynamic flow model based on Daganzo's cell transmission models [21,22]. The novelty in this model is to describe each outgoing flow rate as a nonlinear mixed logical switching function over the source link's density, the destination link's density and capacity, and the driver's potential psychological response to the past traffic light signals. This outgoing flow rate model makes our approach applicable to both under-saturated and over-saturated situations. In our model, the traditional concepts of cycles, splits, and offsets are not adopted, making UTLSP fall in the class of model-based optimization problems, where each traffic light is assigned with a green light period in a real-time manner by the network controller. Upon this novel traffic flow model, we formulate the UTLSP, aiming to reduce the total waiting time over a given finite horizon by real-time scheduling the traffic signal (i.e., either GREEN or RED) for each traffic light in each intersection. The key technical idea is to coordinate traffic light signals in all intersections in response to real-time traffic conditions, which puts our UTLSP in the category of optimization-based coordinated traffic responsive strategies. Due to the large scale of an ordinary traffic network, which usually consists of hundreds of intersections and thousands of road links, the high computational complexity in optimization becomes the major hurdle for our real-time scheduling strategy.

In recent years, many meta-heuristics have been employed to solve many scheduling problems, especially Harmony Search (HS) algorithm. The HS is a relatively recent meta-heuristic method developed by Geem et al. [23] for solving optimization problems. It imitates the music improvisation process of musicians. A harmony in music is regarded analogous to a solution vector in the corresponding optimization problem, while the musician's improvisations are regarded analogous to local and global search processes in optimization algorithms. The HS and its variants have been employed to solve various optimization problems, e.g., water distribution networks [24], flow shop scheduling problems[25,26], job shop scheduling problems [27], knapsack problems [28], Electromagnetic Railgun problems [29], Distributed-Generation System [30], and so forth. Besides, an evolutionary harmony search algorithm was proposed for the loading pattern optimization (LPO) [31]. The literature [32] applied the HS algorithm to improve the connectivity in wireless sensor network. A quasi-oppositional HS algorithm is used to control automatic generation in power system [33]. Also, the HS algorithm is proposed for the vehicle routing problem with time window [34]. The literature [35] proposed an island-based model to improve the convergence of HS. HS and its variants were also employed for solving image reconstruction problems from projections [36], energy-efficient routing problems in wireless sensor networks [37], and highly constrained nurse rostering problems [38]. In these applications, HS is compared to many state-of-art meta-heuristics and the results and comparisons have verified that the HS algorithm can solve practical optimization problems with high competitiveness. The standard HS algorithm has some limitations. For instance, the large number of parameters used in the HS may be as its potential limitation. In each iteration, just one new harmony is generated. The population size should not be very large for the high speed convergence.

Building on the successful real-life applications and limitations of the HS algorithm, we propose a discrete harmony search algorithm to solve the UTLSP with minimizing the total network-wise delay time. In the DHS, a novel new solution generation strategy is proposed, and the number of new harmonies in each iteration is equal to the number of harmonies in population. Three local search approaches with different structures are proposed to improve the performance of DHS in local search space. An ensemble of local search methods is proposed to integrate different neighbourhood structures. The proposed DHS is employed to solving sixteen cases with different scales of the traffic network bearing features of the real traffic network in Singapore. The comparisons and discussions verify the effectiveness of the DHS algorithm with local search approaches and ensemble for solving UTLSP.

The remainder of this study is organized as follows. Section 2 describes the proposed model of the urban traffic light scheduling problem. In Section 3, the proposed DHS algorithm with ensemble of local search operators is presented in detail. The experiment design, comparisons and discussions are shown in Section 4. Finally, we conclude this paper with future works given in Section 5.

2. Urban traffic light scheduling problem

A traffic network consists of a set of links and intersections. For example, a simple unidirectional traffic network is depicted in Fig. 1, where each intersection has only two antagonistic traffic flows. We have proposed a discrete time model based on the cell transmission model. To simplify our technical discussions, some key notations in urban traffic signal scheduling problem formulation are listed as follows:

$C_i(k)$	The number of vehicles in link i in the time interval k.
$f_{ij}(k)$	The exit flow rate from link i to link j in interval k.
Δ	The sampling interval.
$\mathcal{L}$	The set of all one-way links.
$\mathcal{J}$	The set of all intersections.
ω	The stage in intersection, which consists of simultaneous traffic light signals allowing a predefined compatible traffic streams to cross the intersection simultaneously.
Ω_J	The set of stages in intersection J, $J \in \mathcal{J}$.
$\mathcal{F}_J$	$\mathcal{F}_J \subseteq \mathcal{L} \times \mathcal{L}$, the set of all streams in intersection J, i.e., $(i,j) \in \mathcal{F}_J$ means that there exists a traffic stream from link i to link j via intersection J.
$h_J : \Omega_J \to 2^{\mathcal{F}_J}$	The association of each stage to relevant compatible streams.

The following assumptions about a traffic network have been considered, which is suitable for a deterministic analysis:

- The network entrance and exit models are known.
- The link turning ratios in the network are known.
- Each vehicle inside the network will leave the network, delayed only by traffic signals.

In our traffic flow model, we consider the following types of constraints: (1) the stage status constraints, (2) the link volume dynamic constraints, and (3) the flow dynamic constraints. We aim to minimize the network-wise delay time. Next, we will describe each type of constraints and the overall optimization goal one by one.

1) The Stage Status Constraints

At each time interval$k$, there exists only one active stage for an intersection J, as captured by the following constraints:

$$\left(\forall \omega \in \Omega_J\right) \theta_\omega(k) = 0 \Rightarrow \left(\forall (i,j) \in h_J(\omega)\right) f_{i,j}(k) = 0 \tag{1a}$$

$$\sum\nolimits_{\omega \in \Omega_J} \theta_\omega(k) = 1 \tag{1b}$$

$$\left(\forall \omega \in \Omega_J\right) (\forall k \in \mathbb{N}) \theta_\omega(k) \in \left\{0, 1\right\} \tag{1c}$$

where $\theta_\omega(k) = 0$ and $\theta_\omega(k) = 1$ denote the RED and GREEN traffic lights associated with stage ω respectively, and $\mathbb{N}$ denotes the set of natural numbers. Condition (1a) means that if the stage traffic is RED, then all relevant flow rates are zero. Condition (1b) indicates that there can be only one GREEN traffic stage at any time.

2) The Link Volume Dynamic Constraints

Due to the conservation of vehicles, each link $J \in \mathcal{J}$ has the following volume dynamics:

$$C_j(k+1) = C_j(k) + \left(d_j(k) - s_j(k)\right) \Delta \tag{2a}$$

$$(\forall k \in \mathbb{N}) C_j(k) \in \mathbb{N} \tag{2b}$$

where $d_j(k)$ and $s_j(k)$ are entrance and exit flow rates of link $j$in time interval $k$respectively, i.e.:

$$d_j(k) = \sum\nolimits_{i \in \mathcal{L}:(i,j) \in \cup_{J \in \mathcal{J}} \mathcal{F}_J} f_{ij}(k)$$

$$s_j(k) = \sum\nolimits_{i \in \mathcal{L}:(j,i) \in \cup_{J \in \mathcal{J}} \mathcal{F}_J} f_{ji}(k)$$

3) The Flow Dynamic Constraints

For the example shown in Fig. 1, $d_j(k) = f_{ij}(k)$ and $s_j(k) = f_{ji}(k)$. For each stage $\omega \in \Omega_J$ and each stream $(i,j) \in h_J(\omega)$, the exit flow $f_{ij}(k)$ is determined by the current upstream link volume $C_j(k)$, the current remaining downstream link capacity $\hat{C}_j - C_j(k)$, where $\hat{C}_j$ is the maximum volume of link j, and the traffic light signals in the past $r+1$ time intervals $\theta_\omega(k-r), \ldots, \theta_\omega(k)$, i.e.,

$$f_{ij}(k) = g_{ij}(C_i(k), \hat{C}_j - C_j(k), \theta_\omega(k-r), \ldots, \theta_\omega(k)) \tag{3}$$

where $g_{ij}(\cdot)$ is a nonlinear function. The motivation of this model is that if stage ω has been active for the past r intervals, then the drivers intend to keep a high speed as long as the downstream link has sufficient capacity to receive the flow. The following method is employed to define $g_{ij}(\cdot)$. Assume that there are $r+1$ monotonically non-increasing speed categories: $l_{ij}^0 \geq l_{ij}^1 \geq \ldots \geq l_{ij}^r$, which denotes speed ranges from high to low. The actual speed category $l_{ij}(k)$ is determined as follows:

$$l_{ij}(k) = \sum\nolimits_{p=0}^{r} \delta_{ij}^p(k) l_{ij}^p \tag{4a}$$

$$\sum\nolimits_{p=0}^{r} \delta_{ij}^p(k) - \theta_\omega(k) = 0 \tag{4b}$$

$$(\forall q : 0 \leq q \leq r-1), \left(1 - \theta_\omega(k-q-1)\right) \prod\nolimits_{p=0}^{q} \theta_\omega(k-p) = 1 \Leftrightarrow \delta_{ij}^{r-q}(k) = 1 \tag{4c}$$

$$\prod\nolimits_{p=0}^{r} \theta_\omega(k-p) = 1 \Leftrightarrow \delta_{ij}^0(k) = 1 \tag{4d}$$

$$(\forall p : 0 \leq p \leq r) \delta_{ij}^p \in \left\{0, 1\right\} \tag{4e}$$

Condition (4b) indicates that if the traffic light of stage ω is RED, i.e., $\theta_\omega(k) = 0$, then $\sum_{p=0}^{r} \delta_{ij}^p(k) = 0$, which by Condition (4a) means $l_{ij}(k) = 0$, i.e., the speed in the stage ω must be zero. If the traffic light of stage $\omega$is GREEN, i.e. $\theta_\omega(k) = 1$, then by Condition (4a), $l_{ij}(k)$ can only choose one speed category because of $\sum_{p=0}^{r} \delta_{ij}^p(k) = 1$. Conditions (4c)–(4d) means that the actual speed category depends on the number of consecutive green light intervals from k backward in time, the larger the number of consecutive green intervals, the higher the speed category. There is simple example to explain the equations from (4a) to (4e). Assume that the system contains 5 speed items (r = 4) in the speed category, i.e., $l_{ij}^0, l_{ij}^1, l_{ij}^2, l_{ij}^3, l_{ij}^4$. If the consecutive green light intervals in one intersection get the number 3, i.e., p = r − 3 + 1 = 4 − 3 + 1 = 2, thespeedcategory$\delta_{ij}^2(k) = 1$, all other items in speed category equal to 0, $\delta_{ij}(k) = 0$. Thus, the speed category for this intersection is $l_{ij}(k) = \sum_{p=0}^{r} \delta_{ij}^p(k) l_{ij}^p = l_{ij}^2$. For another intersection, if the consecutive green light number is 1, then the vehicles pass this intersection with a speed of $l_{ij}(k) = l_{ij}^4$. The bigger the consecutive green light number, the higher the speed category is selected.

After $l_{ij}(k)$ being determined, the link flow rate $f_{ij}(k)$ is given as follows:

$$f_{ij}(k)\Delta = \min\{\lambda_{ij}(k) C_j(k), l_{ij}(k)\left(\hat{C}_j - C_j(k)\right)\} \tag{5a}$$

$$f_{ij}(k)\Delta \in \mathbb{N} \tag{5b}$$

where $\lfloor \cdot \rfloor$ means the largest integer not greater than input argument, and $\lambda_{ij}(k)$ is the turning ratio of vehicles in link i towards link j at time interval k, which is assumed to be known in advance. It is clear that $\sum_{j \in \mathcal{L}:(i,j) \in \cup_{J \in \mathcal{J}} \mathcal{F}_J} \lambda_{ij}(k) = 1$, each vehicle in link $i$will move to some downstream link j. Condition (5a)–(5b) indicate that the number of vehicles in one time interval, $f_{ij}(k)\Delta$, is the largest integer that is upper bounded by the upstream volume $\lambda_{ij}(k) C_j(k)$ of link i and the downstream remaining capacity $\hat{C}_j - C_j(k)$ weighted by the speed category $l_{ij}(k)$.

4) Objective Function

The total network-wise delay time within N time intervals can be estimated as follows:

$$\sum\nolimits_{i \in L} \sum\nolimits_{k=1}^{N} C_i(k) \left(1 - \frac{\bar{v}_i(k)}{v_{i,max}}\right) \Delta = \sum\nolimits_{i \in L} \sum\nolimits_{k=1}^{N} \left(C_i(k) - \frac{L_i}{v_{i,max}} s_i(k)\right) \Delta$$

where $\bar{v}_i(k)$ is the average speed of link i in time interval k, which can be approximated by the ratio of the exit flow rate $s_i(k)$ and the average link density $C_i(k)/L_i$, where L_i is the length of link i, due to the assumption that vehicles in link i are uniformly distributed with identical speeds, i.e., the acceleration step is negligible. Based on the Conditions (2a)–(2b), the following formula is used as objective function for the scheduling purpose [39]:

$$\text{Min} \sum\nolimits_{i \in L} \sum\nolimits_{k=1}^{N} \left(C_i(k) - \frac{L_i}{v_{i,max}} \sum\nolimits_{j \in \mathcal{J}:(i,j) \in \cup_{J \in \mathcal{J}} \mathcal{F}_J} f_{ij}(k)\right) \Delta \tag{6}$$

3. A discrete harmony search (DHS) algorithm for UTLSP

3.1. Standard HS

The HS algorithm was inspired by the natural musical performance process whereby a musician searches for a better state of harmony. In the HS, each solution is called a "harmony" which is a n dimensional vector. The basic single objective HS algorithm is summarized as given follows:

Step 1: initialization

The harmony vector or a candidate solution is represented as a n dimensional real-valued vector $X = \{x(1), x(2), \ldots, x(n)\}$, where each element $x(k)$ denotes a decision variable of the optimization problem. This step sets parameters and fills the harmony memory (HM) with a population of initial harmonies (solutions). Parameters include the harmony memory size (HMS), harmony memory consideration rate ($HMCR$), pitch adjustment rate (PAR), and distance bandwidth (BW). The HM is initialized by randomly generated solution vectors.Let $X_i = \{x_i(1), x_i(2), \ldots, x_i(n)\}$ represent the i^{th} harmony vector in the HM, which is generated as follows:

$X_i(j) = \mathrm{LB}(j) + (UB(j) - LB(j)) * \mathrm{rand}()$ forj = 1, 2, and , i = 1, 2, . . ., HMS (7)where $rand()$ is a uniform random number in the range of [0,1], $LB(j)$ and $UB(j)$ are the lower and upper bounds of the decision variable $X_i(j)$, respectively.

Step 2: improvising a new harmony

The process of generating a new harmony vector, X_{new}, is called improvisation. X_{new} is produced by applying three rules: a memory consideration, a pitch adjustment, and a random selection. First, if a uniform random number $rand()$ in the range of [0,1] is less than $HMCR$, the decision variable $X_{new}(j)$ is generated by the memory consideration. Otherwise, $X_{new}(j)$ is obtained by a random selection generated between the lower and upper bounds of the decision variable $x_i(j)$, as shown in Formula (10). Secondly, each decision variable $X_{new}(j)$ will undergo a pitch adjustment with a probability of PAR if it is updated by the memory consideration. The memory consideration, pitch adjustment and random selection are given in Eqs. (8),–(10), respectively, as given follows.

$$if(rand1() < HMCR)$$
$$X_{new}(j) = X_i(j) \tag{8}$$

$$if\ (rand2() < PAR)$$
$$X_{new}(j) = X_{new}(j) \pm \mathrm{rand}() \times \mathrm{BW} \tag{9}$$

else

$$X_{new}(j) = LB(j) + (UB(j) - LB(j)) * rand() \tag{10}$$

where $i \in (1, 2, \ldots, HMS)$, $X_i(j)$ is randomly selected from HM.

Step 3: updating the HM

The***HM*** will be updated if X_{new} is better than the worst harmony vector (X_w) in the HM. Thus, X_{new} will replace $\boldsymbol{X_w}$ and become a member of the updated ***HM***

Step 4: stopping condition

If a termination criterion (e.g., the maximum number of iteration) is met, return the best harmony vector (X_B) in the HM. Otherwise, go to **Step 2**.

Metaheuristic optimization methods have been widely and efficiently applied for solving real-life optimization problems. Indeed, metaheuristics are inspired by metaphors which are taken from nature or other phenomena. Therefore, various optimization methods based on a variety of metaphors have been proposed in the recent literature. Despite the fact that some of these methods have been numerically proven in many different fields in terms of reliably and efficiency, some argue against these methods with respect to irrelevant metaphors have been raised. Hence, the HS has not been excluded from such arguments.

Every metaheuristic algorithm shares similarity as well as uniqueness with the other optimization methods (e.g., evolutionary strategy). Also, there is a chance for HS to become a general form of another algorithm. Thus, for instance, if HS is a special form of evolutionary strategy (ES), ES can be a special form of genetic algorithm (GA), and GA can be a special form of HS. Hence, there is a chance that GA and ES are closer than HS and ES [40]. More basically, every metaheuristic algorithm contains only two important phases which are global search and local search. The key factor to an algorithm's success is how efficiently two phases contribute using certain number of operations (even sometimes different algorithms share the identical operation). In fact, any algorithm can be survived as long as it is the fittest or at least fitter than others instead of being novel. More discussions on HS regarding these issues (e.g., metaphors and novelty) have been addressed in the recent literatures [40,41].

In this paper, based on the standard HS algorithm, we propose a discrete harmony search (DHS) algorithm. Moreover, an ensemble of local search operators is proposed to improve the neighbouring search performance of the DHS algorithm. The corresponding contents will be presented in the following sections.

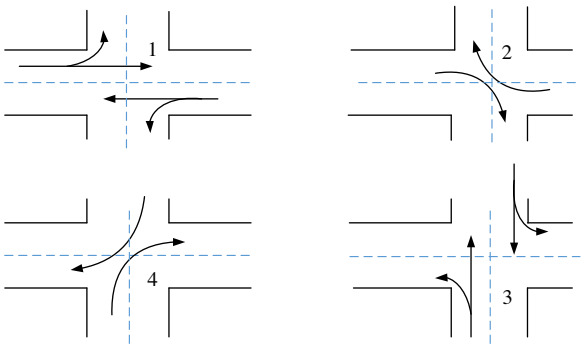

Fig. 2. Four stages of an intersection.

Table 1
Vector solution generated randomly for nine intersections in one sampling interval.

Intersection Number	1	2	3	4	5	6	7	8	9
Traffic stage $H_i(k)$	2	1	1	3	4	4	2	3	3

3.2. The proposed DHS

3.2.1. Solution representation

Fig. 2 shows the four stages as a cycle in an intersection. Different stages include different compatible traffic flows. In the fixed cycle model, the four stages are repeated based on the order 1–2–3–4. However, in the proposed model, each stage is assigned with a period in a real-time manner by the network controller instead of the repetition of cycles. In the proposed DHS algorithm, four integers, "1", "2", "3" and "4", are used to denote the four stages shown in Fig. 2. A harmony is employed to represent the stages of all intersections at one-time window of a traffic network. An example of harmony is shown in Table 1. There are nine intersections and each intersection is assigned one stage. The harmony in Table 1 is a traffic stage assignment in one sampling interval and the length of the harmony is equal to the number intersections. Each value represents an assigned stage of one intersection in one sampling interval. To evaluate total network-wise delay time of N sampling intervals, the length of harmony would be N times the number of intersections.

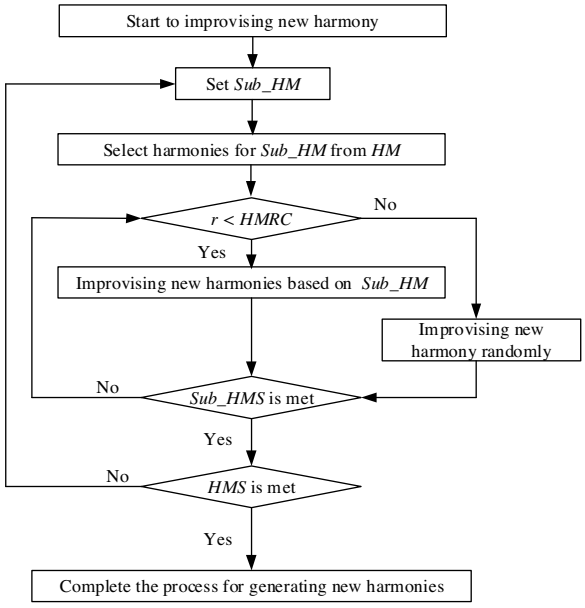

Fig. 3. Flow chart for generating new harmony in the proposed DHS.

3.2.2. Improvising new harmonies

In the standard HS algorithm, a new harmony is improvised from the existing harmonies in *HM* or randomly, and in each iteration, only one new harmony is improvised. In this study, a sub-harmony-memory (*sub_HM*) based method is proposed to improvise new harmonies. The size of *sub_HM* is less than the half size of *HM* (*HMS*). Therefore, the total size of all *sub_HM* should be equal to the *HMS*. The flow chart for generating a new harmony is shown in Fig. 3 and the detailed procedure for improvising the new harmony is shown as follows:

Different with the standard HS, the proposed method generates a new harmony based on a small harmony memory so called *sub_HM*. The purpose of the proposed procedure is to improve the convergence speed of the HS algorithm. The small harmony memory size can reduce the search space to improve the performance of local search and intensify the exploration process in the proposed method, which will be verified in the results and discussion section. The global search performance can be kept because the harmonies in the *sub_HM* are selected from the *HM*. Hence, the proposed method for generating new harmony balances the local search and global search performances of the DHS algorithm.

3.2.3. Ensemble of local search

In the UTLSP, one intersection is linked to the other intersections in different directions. Based on the structure of the traffic network and the coding technique in Section 3.1, three neighbourhood structures (Fig. 4) have been proposed and the corresponding local search operators have been developed as follows.

The first neighbourhood structure is for one intersection and the traffic light signals are from all sampling intervals in one simulated time window. For example, the simulation time is 60 s and the sampling interval time is 15 s. Hence, there are four traffic light signal samplings. One intersection would have four signals in 60 s simulated time window. As shown in Fig. 4A, n is equal to 4 and intersection i has four traffic signals. The second neighbourhood structure is related to the multiple connected intersections in the same direction and the traffic light signals of these intersections are selected from one randomly sampling interval among all sampling intervals. Fig. 4B presents the traffic light signals of 3 connected intersections (i–k) in one time interval. The third neighbourhood structure is related to one sub-region of the whole traffic network and the traffic light signals are selected from one randomly sampling interval among all sampling intervals. Fig. 4C shows the traffic light signals of one sub-region with four intersections (a–d) in one sampling interval.

Procedure: Improvising new harmony
Set sub_HM and sub_HM is less than half of HMS.
For $i = 0$ to HMS
  For $j = 0$ to sub_HMS
    Randomly select two harmonies H_1 and H_2 from the HM
    If $(r1 < p1)$ // $r1 \in [0, 1]$ and $p1$ is the probability to select the best harmony between H_1 and H_2.
      If the fitness of H_1 is better than that of H_2, $H_1 \in sub\_HM$; else $H_2 \in sub\_HM$.
    Else
      If the fitness of H_1 is better than that of H_2, $H_2 \in sub\_HM$; else $H_1 \in sub\_HM$.
    End if
  End for
  For $j = 0$ to sub_HMS
    If $r < HMCR$ // $r \in [0, 1]$.
      For $k = 0$ to length of H
        If $(r2 < PAR)$ // $r2 \in [0, 1]$.
          $H_{new}^k = H_j^k$
        Else // Select element from H_{j+1} in HM
          $H_{new}^k = H_{j+1}^k$
      Compare the new harmony H_{new} and harmony H_{j+1} in the HM.
      If the fitness of H_{new} is better than that of H_{j+1}, H_{new} replaces H_{j+1} in HM.
    Else
      Improvising new harmony randomly to replace the current one.
      Calculate the objective values of the new harmony.
    End if
  End for
  $i = i + sub\_HMS$
End for

Three local search operators are proposed for the above three neighbourhood structures. For one harmony in the DHS algorithm, the local search operators are employed to search the neighbouring space of the harmony. To integrate the three neighbourhoods together, an ensemble of local search operators is proposed to improve the performance of the local search. The flow chart of the ensemble is shown in Fig. 5 and the detailed procedure of the ensemble is shown as follows:

```
Procedure: Ensemble of local search operators
R is a random integer number between zero to two
If (R == 0)
   Select one intersection i in one harmony H
   For k=1 to N //N is the number of sampling intervals
      Select a traffic stage for H_i(k) randomly
   End for
Else
   If (R == 1)
      Select one sampling interval s_i in one harmony H
      Select K connected intersections at sample interval s_i of harmony H
      For k=1 to K
         Select a traffic stage for H^{si}(k) randomly
      End for
   Else
      Select one sampling interval s_i in one harmony H
      Select one sub-region with M intersections at sample interval s_i of harmony H
      For k=1 to M
         Select a traffic stage for H^{si}(k) randomly
      End for
   End if
End if
Get new solution H'
Evaluate the objective value of H'
If H' is better than H
   Replace H using H'
Else
   Remain H
End if
```

This ensemble of local search operators is defined to find neighbouring solutions of one harmony. In each iteration of the DHS algorithm, the ensemble is executed for each harmony to find better solutions in the neighbouring space. The three neighbour structures are selected with the same possibilities to improve the performance of the DHS algorithm in order to intensify the exploitation.

3.2.4. Procedure of the proposed DHS algorithm

Besides the novel method for improvising a new harmony and the ensemble of local search operators, we utilized a mechanism inspired by the artificial bee colony (ABC) algorithm [42] to avoid one solution falling into local extremal. Therefore, a limit iteration number (*LIN*) as a parameter is set. If the objective of one harmony cannot be improved in *LIN* iterations, the harmony will be given up and a new harmony will be generated to replace the current harmony. Based on the *LIN* mechanism and the ensemble of local search operators, the DHS algorithm can balance the exploration

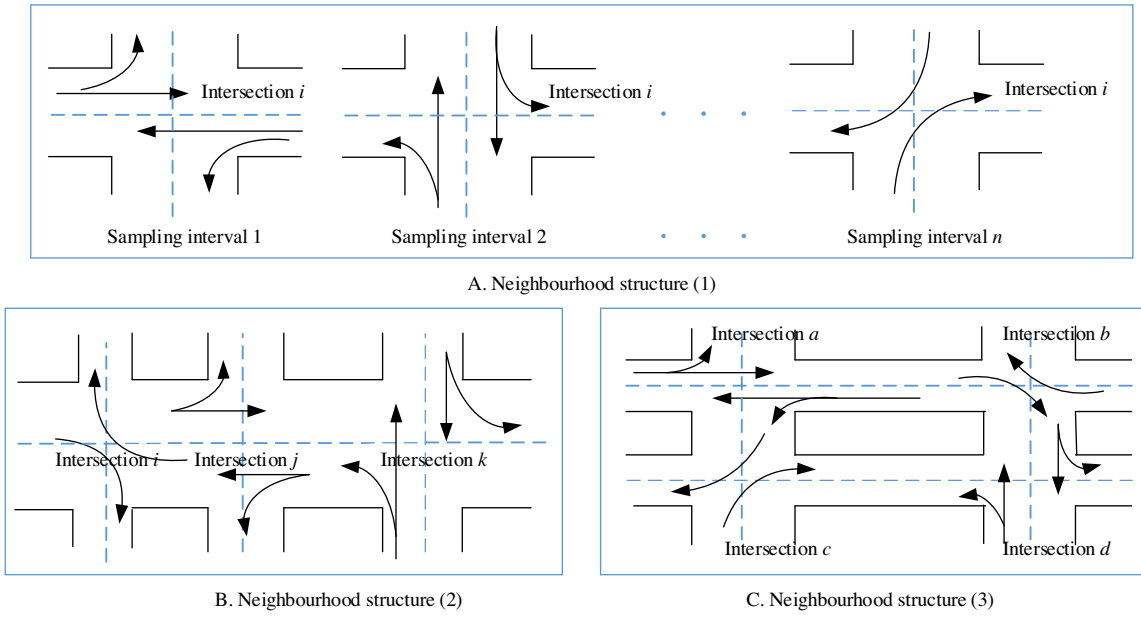

Fig. 4. Three neighbouring structures of traffic network.

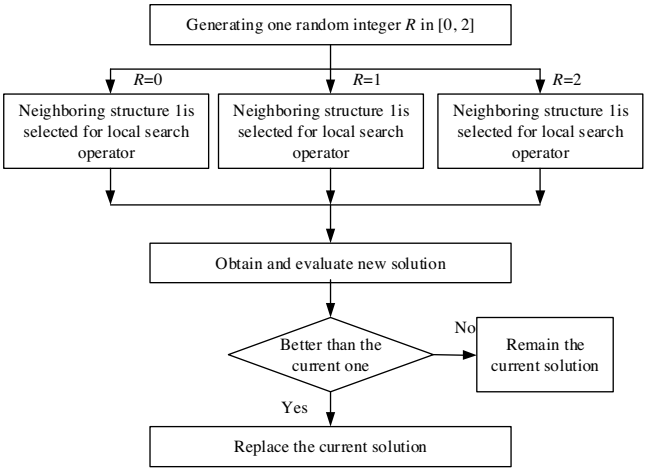

Fig. 5. Flow chart of ensemble of local search operators.

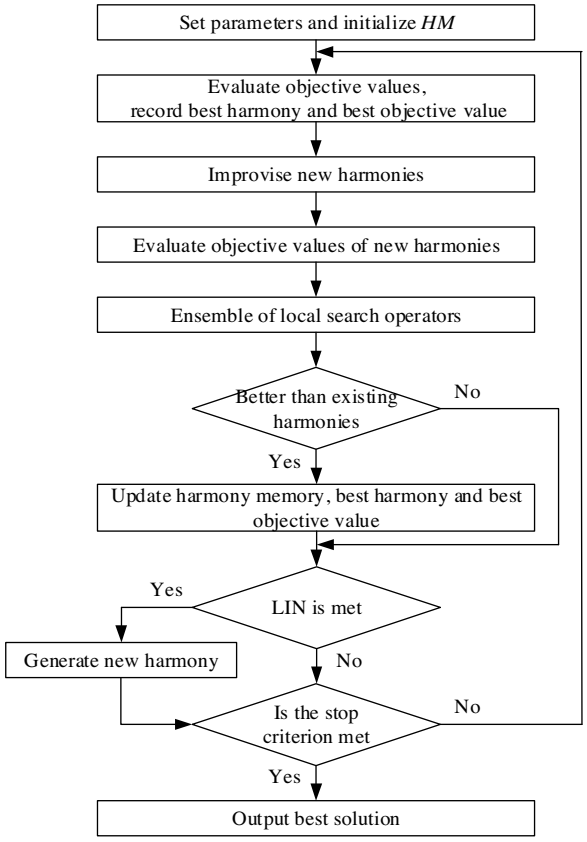

Fig. 6. Framework of DHS algorithm.

Fig. 6. Framework of DHS algorithm.

and exploitation performance. To clearly show the process of DHS, the framework of the DHS algorithm is given as shown in Fig. 6.

4. Experiment evaluations and comparisons

4.1. Experimental setup

To evaluate the performance of the proposed DHS algorithm, experimental evaluations and comparisons are conducted. Two sets of traffic light scheduling cases are generated based on the real-life traffic data in Singapore. The first set consists of eight cases with a time window of 30 s. Accordingly, the second set includes eight cases with a time window of 60 s. The scales of cases are from 3×3, nine intersections to 10×10, one hundred intersections. The proposed DHS algorithm is coded in C++ and run on an Intel 3.40 GHz PC with 8 GB memory. Based on our research experience and suggested initial parameters in the literature [43,44], the HMS and the maximum number of iteration, respectively, are set to 50 and 1000, HMCR value of 0.95, PAR value of 0.5 and the size of Sub_HM of 10 have been selected as initial parameters used in HS and its variants. All experiments are carried out with 30 independent replications.

For the UTLSP, the sampling time is an important parameter in the model. It decides how many intervals in one-time window and how many times to set the traffic light signal for each intersection. For instance, if the sampling time is set to 5 s, the interval number in 60 s time window is 12. If the sampling time is set to 10 s, the corresponding interval number would be 6. Hence, the length of sampling time would affect the optimal value of the network-wise total delay time. Based on the characteristics of UTLSP, the sampling time is set to different values (i.e., 5, 10 15, 20, and 30) to test the influences of the sampling time on the network-wise total delay time. The case with the largest scale (i.e., 100 intersection) in Set two is solved with five sampling time values. For each value, the concerned case is solved 30 repeats and the minimum value and average value of the total delay time are computed. The corresponding results are shown in Fig. 7. It is clear that the applied case has the best minimum and average results in 30 runs when the sampling time is set to 15 s. Hence, the sampling time is set to 15 s for all the following experiments.

For the DHS algorithm, the user parameter *LIN* is inspired by ABC algorithm to avoid the algorithm falling into local optimal. To test the influences of *LIN* for algorithm's performance, the *LIN* is set to three values (i.e., 20, 50, and 100) for solving the case with the largest scale in set two. For each *LIN* value, the algorithm runs 30 repeats and the minimum value and average value of the total delay time are reported. The corresponding results are shown in Fig. 8. It is clear that the case has the best minimum and average results in 30 runs when the *LIN* number is set to 50. Hence, the value of *LIN* is set to 50 for all the following experiments.

Besides of the total delay time results of the whole traffic network, the relative percentage deviation (RPD) has been calculated over the minimum and average results in 30 repeats obtained by each compared algorithm for each case. The formula to compute the RPD is shown as follows:

$$RPD(k) = \frac{R_k - R_{FCS}}{R_{FCS}} \times 100 \quad (11)$$

where, R_{FCS} is the result by the fixed cycle traffic light control system (FCS) in Singapore which assigns dynamic time ranges for each stage, R_k is the result by the k^{th} compared algorithm. In the comparison results, also the average relative percentage deviation (ARPD) of each algorithm has been computed for each case in each set. For all compared algorithms, the smaller RPD and ARPD values mean the better performance of corresponding algorithms.

Based on the aforementioned experimental setup, we will compare the standard HS algorithm with the DHS algorithm in Section 4.2. In Section 4.3, we will discuss the performance of the DHS algorithm with and without local search operators and ensemble of local search operators.

4.2. Comparisons of standard HS and DHS

In this section, the standard HS and DHS algorithms have been compared with the FCS. For sixteen cases in two sets, the results of the network-wise total delay time are calculated. The minimum (Min) values and average (Ave) values of results in 30 repeats are

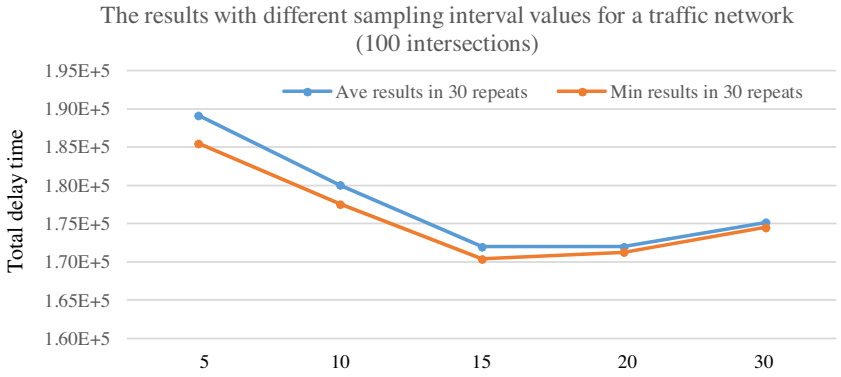

Fig. 7. Comparisons of results with different sampling time values for 100 intersections.

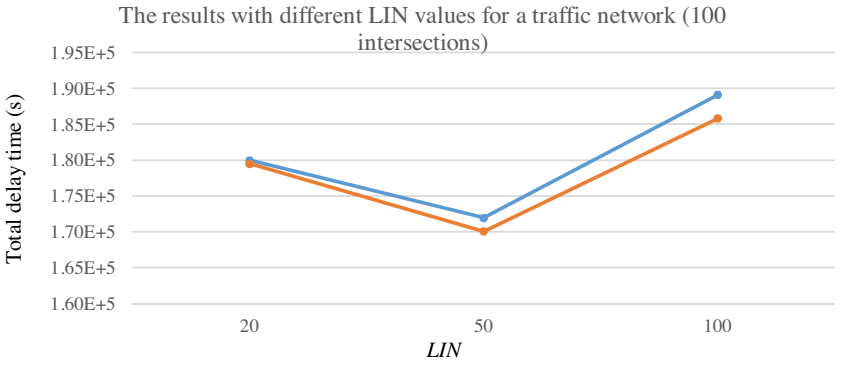

Fig. 8. Comparisons of results with different LIN values for 100 intersections.

Table 2
Comparisons of standard HS and DHS for eight cases with 30 s time window in set one.

Case	FCS (s)	Standard HS (s)		DHS (s)		RPD (%)_Standard HS		RPD (%)_DHS	
		Min	Ave	Min	Ave	Min	Ave	Min	Ave
3 × 3	9534	8490	8625	8325	8325	−11.0	−9.5	−12.7	−12.7
4 × 4	16,609	15,045	15,570	14,175	14,175	−9.4	−6.3	−14.7	−14.7
5 × 5	24,122	21,645	22,350	19,740	19,740	−10.3	−7.3	−18.2	−18.2
6 × 6	33,781	29,940	31,200	26,895	26,910	−11.4	−7.6	−20.4	−20.3
7 × 7	44,666	41,520	42,855	35,820	35,880	−7.0	−4.1	−19.8	−19.7
8 × 8	58,007	54,285	55,020	46,845	46,845	−6.4	−5.2	−19.2	−19.2
9 × 9	72,960	68,895	70,140	59,115	59,130	−5.6	−3.9	−19.0	−19.0
10 × 10	89,704	84,480	85,230	71,835	71,910	−5.8	−5.0	−19.9	−19.8
ARPD (%)						−8.4	−6.1	−18.0	−17.9

counted. Compared to the FCS, the RPD values of the Min and Ave results are obtained from standard HS and DHS algorithms, respectively. For the two concerned sets, we also compute the ARPD. Table 2 shows the total delay time and RPD values for the eight cases with 30 s time window while

Table 3 shows the corresponding results and RPD values for the eight cases with 60 s time window.

It can be seen from Tables 2 and 3 that the standard HS algorithm is able to obtain results of the total delay time objective better than the FCS for sixteen cases in two sets. The Min and Ave results by the standard HS algorithm are smaller than those by the FCS. For the eight cases in set one, the RPD values of the Min and Ave results by the standard HS algorithm are from −5.6% to −11.4% and from −3.9% to −9.5%, respectively. For the eight cases in set two, the RPD values of the Min and Ave results by the standard HS algorithm are from −5.6% to −11.4% and from −3.9% to −9.5%, respectively. The ARPD values of both Min and Ave results by the standard HS algorithm are −8.4% and −6.1%, respectively. Hence, the standard HS algorithm has better performance than the FCS for traffic light control.

Compared to the standard HS algorithm, the DHS algorithm offers better Min and Ave results of the total delay time for all cases in two sets. The RPD values of the Min and Ave results by the DHS algorithm are also better than those by the standard HS algorithm. For eight cases in set one, the ARPD values of the Min and Ave results by the DHS algorithm (-18.0%, −17.9%) are better than those (i.e., −8.4% and −6.1%) by the standard HS algorithm. For eight cases in set two, the ARPD values of the Min and Ave results by the DHS algorithm (i.e., −31.0% and −30.3%) are better than those (i.e., −21.7% and −21.8%) by the standard HS algorithm. Hence, the DHS algorithm is more competitive than the standard HS algorithm for UTLSP.

Besides the total delay time results and the RPD values obtained by the standard HS and DHS algorithms, we also record the computational time of the standard HS and DHS algorithms. The average computational times of the standard HS and DHS algorithms for all

Table 3
Comparisons of standard HS and DHS for eight cases with 60 s time window in set two.

Case	FCS (s)	Standard HS (s)		DHS (s)		RPD (%)_Standard HS		RPD (%)_DHS	
		Min	Ave	Min	Ave	Min	Ave	Min	Ave
3 × 3	26,456	22,230	22,740	20,265	20,280	−16.0	−14.0	−23.4	−23.3
4 × 4	44,576	39,810	40,515	32,910	33,045	−17.3	−18.4	−26.2	−25.9
5 × 5	66,846	57,810	59,415	45,195	45,720	−21.8	−23.0	−32.4	−31.6
6 × 6	93,471	81,450	82,590	61,785	62,265	−24.1	−24.6	−33.9	−33.4
7 × 7	121,887	107,430	109,275	80,970	81,705	−24.6	−25.2	−33.6	−33.0
8 × 8	158,442	139,920	141,915	105,630	106,530	−24.5	−24.9	−33.3	−32.8
9 × 9	199,707	175,470	176,685	134,100	136,830	−23.6	−22.6	−32.9	−31.5
10 × 10	244,812	211,410	216,135	165,855	169,200	−21.5	−21.7	−32.3	−30.9
ARPD (%)						−21.7	−21.8	−31.0	−30.3

Table 4
The computation time (second) of standard HS and DHS algorithms for sixteen cases in two sets.

Case	30 s time window		60 s time window	
	Standard HS (s)	DHS (s)	Standard HS (s)	DHS (s)
3 × 3	0.230	0.192	0.443	0.354
4 × 4	0.356	0.279	0.705	0.537
5 × 5	0.542	0.387	1.031	0.709
6 × 6	0.706	0.505	1.445	0.972
7 × 7	0.945	0.640	1.972	1.240
8 × 8	1.179	0.851	2.450	1.608
9 × 9	1.550	1.024	3.425	2.112
10 × 10	1.823	1.347	4.182	2.642

cases are shown in Table 4. By observing Table 4, computational time of standard HS and DHS algorithms are less than 2 s for eight cases with 30 s time window while the corresponding computation times for the cases with 60 s time window are less than 4.2 s. Therefore, the DHS algorithm shows some superiority in term of computational time for the same case. For instance, the average computational time of the standard HS is 4.182 s for the largest case (i.e., 100 intersections) with 60 s time window, while the correspond consumed time of DHS algorithm is 2.642 s. It means that the computational time of DHS algorithm is more satisfying for the practical situations of UTLSP.

In summary, the HS algorithm as a meta-heuristic can efficiently solve the UTLSP with better performance than the current FCS and its competitiveness is more obvious with the increasing of the time window (30 s in Set one and 60 s in Set two). The proposed DHS algorithm can considerably improve the results over the standard HS algorithm. Indeed, the computational time of the standard HS algorithm and the DHS algorithm are reasonable for real-life UTLSP.

4.3. DHS with ensemble of local search operators

To further improve the performance of the DHS algorithm, three neighbourhood structures and an ensemble of local search operators for the three neighbourhoods are proposed in Section 3.2.3. In this section, the performance of five algorithms, DHS, DHS with each of three local search operator (DHS_Local1, DHS_Local2, and DHS_Local3), and DHS with ensemble (DHS_Ensemble) have been examined. The optimization results for the eight cases in Set one are shown in Table 5, while the corresponding results for the eight cases in set two are shown in Table 6. For each case, the best Min results and best Ave results by all compared algorithms highlighted in bold. The ARPD values have been compared among the all optimizers. The ARPD values for the eight cases in set one are shown in Fig. 9 while the corresponding results for the eight cases in Set two are shown in Fig. 10.

It can be seen from Table 5 that the DHS and DHS_Local3 algorithms could just obtain the best Min and Ave results for only one case (i.e., Case 3 × 3, nine intersections) out of eight cases in Set one. The DHS_Local1 algorithm obtained the best Min results for five out of eight cases and the best Ave results for six out of eight cases, while the DHS_Local2 algorithm found the best Min results for seven out of eight cases and the best Ave results for three out of eight cases. Compared to the DHS and DHS with single local search operator, DHS_ ensemble algorithm could attain better results. For eight cases in set one, DHS_ ensemble could find the best Min results and the best Ave results for all cases except Case 7 × 7 (i.e., forty-nine intersections). It can be seen from Fig. 9 that the DHS_Local1, DHS_Local2, and DHS_Ensemble algorithms could considerably improve the ARPD values by DHS algorithm. As a conclusion, among the three algorithms, DHS_Ensemble has obtained the best ARPD value for the eight cases in Set one.

However, it can be seen from Table 6 that the DHS and DHS_Local3 algorithms could not get the best Min and Ave results for any case. The DHS_Local1 algorithm obtained the best Min results for three out of eight cases and the best Ave results for two out of eight cases, while the DHS_Local2 algorithm found the best Min results and the best Ave results for two out of eight cases. Compared with the DHS and DHS with single local search operator, DHS_ ensemble algorithm could obtain much better results. For eight cases in set two, the DHS_ ensemble could attain the best Min results and the best Ave results for all cases except Case 5 × 5 (i.e., twenty-five intersections). It can be seen from Fig. 10 that the DHS_Local1, DHS_Local2 and DHS_Ensemble algorithms could considerably improve the ARPD values versus the DHS algorithm. However, the comparison among the three algorithms shows the superiority of DHS_Ensemble in term of the best ARPD value for all cases in Set two.

Based on the given comparisons and discussions, it can be concluded that the DHS_Local3 algorithm has not shown any obvious advantage with respect to the DHS algorithm. The DHS_Local1, DHS_Local2, and DHS_Ensemble algorithms could distinctly improve the results obtained by the DHS, and the DHS_Ensemble algorithm has the best performance among the three algorithms. The computational time of DHS and four variants are tabulated and shown in

Table 7 Compared to the DHS algorithm, the computational time of DHS with local search operators and ensemble are almost the same. It is worth mentioning that compared with the sampling interval time (i.e., 15 s), all considered algorithms can satisfy the real-life UTLSP.

For the cases in Set one, the ARPD values for Min and Ave results by DHS are −17.97% and −17.93%, respectively, while the corresponding values obtained by the DHS_Ensemble algorithm are −18.26% and −18.23, respectively. Compared to the DHS, the DHS_Ensemble algorithm could improve the Min and Ave results by 0.29% and 0.30%, respectively. For the cases in Set two, the corresponding improvements for the Min and Ave results are 2.00% and 2.51%, respectively. Hence, compared to the DHS algorithm,

Table 5
Comparisons of DHS and DHS with ensemble for eight cases with 30 s time window.

Case	DHS (s)		DHS_Ensemble (s)		DHS_Local1 (s)		DHS_Local2 (s)		DHS_Local3 (s)	
	Min	Ave	Min	Ave	Min	Ave	Min	Ave	Min	Ave
3 × 3	**8325**	**8325**	**8325**	**8325**	**8325**	**8325**	**8325**	**8325**	**8325**	**8325**
4 × 4	14,175	14,175	**14,160**	**14,160**	**14,160**	**14,160**	**14,160**	**14,160**	14,175	14,175
5 × 5	19,740	19,770	**19,650**	**19,650**	**19,650**	**19,650**	**19,650**	19,740	19,740	19,830
6 × 6	26,910	26,910	**26,835**	**26,835**	**26,835**	**26,835**	**26,835**	**26,835**	26,895	26,910
7 × 7	35,820	35,850	35,715	35,730	35,715	**35,715**	**35,700**	35,745	35,895	35,895
8 × 8	46,845	46,875	**46,590**	**46,590**	**46,590**	**46,590**	**46,590**	46,680	46,845	46,845
9 × 9	59,115	59,145	**58,725**	**58,755**	58,860	58,980	58,920	59,010	58,995	59,130
10 × 10	71,835	71,880	**71,445**	**71,580**	71,520	71,595	**71,445**	71,685	71,715	71,805

Table 6
Comparisons of DHS and DHS with ensemble for eight cases with 60 s time window.

Case	DHS (s)		DHS_Ensemble (s)		DHS_Local 1 (s)		DHS_Local 2 (s)		DHS_Local 3 (s)	
	Min	Ave	Min	Ave	Min	Ave	Min	Ave	Min	Ave
3 × 3	20,235	20,295	**20,100**	**20,160**	**20,100**	**20,160**	**20,100**	**20,160**	20,220	20,340
4 × 4	32,910	32,940	**32,835**	**32,835**	32,880	32,880	32,880	32,940	32,940	33,180
5 × 5	45,915	46,140	45,165	45,180	45,060	45,180	**45,045**	**45,165**	45,090	45,330
6 × 6	62,010	62,790	**60,450**	**60,630**	**60,450**	60,930	60,960	61,290	61,995	62,580
7 × 7	81,240	82,785	**77,925**	**78,420**	78,990	79,245	78,915	79,305	79,635	80,790
8 × 8	105,765	106,950	**100,710**	**100,935**	100,800	101,205	101,415	101,880	104,610	105,540
9 × 9	133,635	136,155	**128,040**	**129,210**	128,820	129,390	129,090	130,425	131,265	132,705
10 × 10	164,550	167,430	**155,235**	**155,850**	**155,235**	**155,850**	156,015	157,560	160,500	163,500

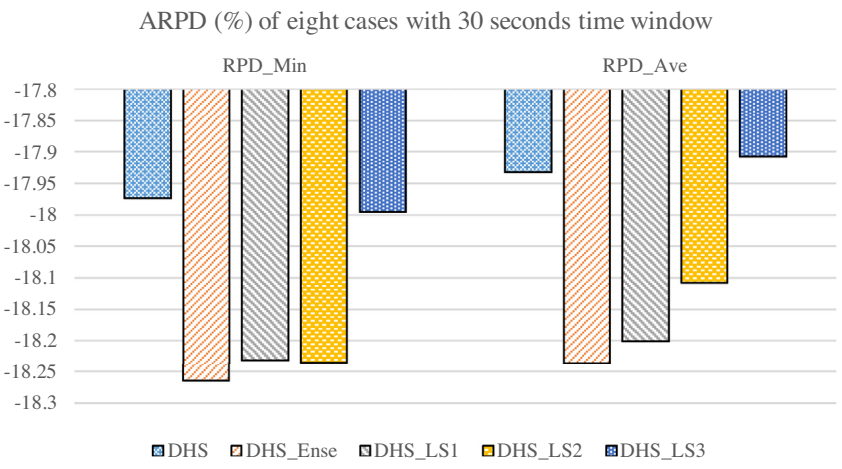

Fig. 9. ARPD results by the DHS with and without local search for the cases in set one.

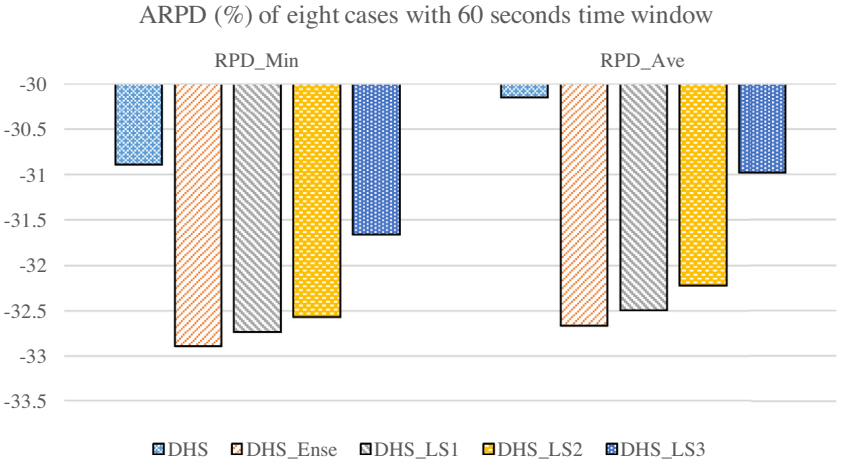

Fig. 10. ARPD results by the DHS with and without local search for all cases.

Table 7
The computational time (second) of DHS and DHS with three local search operators and ensemble.

Case	30 s					60 s				
	DHS	DHS_Ensemble	DHS_L1	DHS_Ll2	DHS_L3	DHS	DHS_Ensemble	DHS_L1	DHS_L2	DHS_L3
3 × 3	0.19	0.30	0.32	0.31	0.30	0.35	0.49	0.50	0.48	0.49
4 × 4	0.28	0.42	0.43	0.44	0.43	0.54	0.72	0.72	0.71	0.72
5 × 5	0.39	0.58	0.58	0.58	0.57	0.71	1.01	1.03	1.00	0.99
6 × 6	0.51	0.76	0.75	0.76	0.74	0.97	1.37	1.38	1.33	1.33
7 × 7	0.64	0.97	0.94	0.97	0.96	1.24	1.79	1.78	1.76	1.77
8 × 8	0.85	1.19	1.17	1.21	1.18	1.61	2.31	2.21	2.23	2.22
9 × 9	1.02	1.48	1.47	1.50	1.45	2.11	2.92	2.80	2.85	2.80
10 × 10	1.35	1.77	1.72	1.83	1.72	2.64	3.65	3.47	3.70	3.49

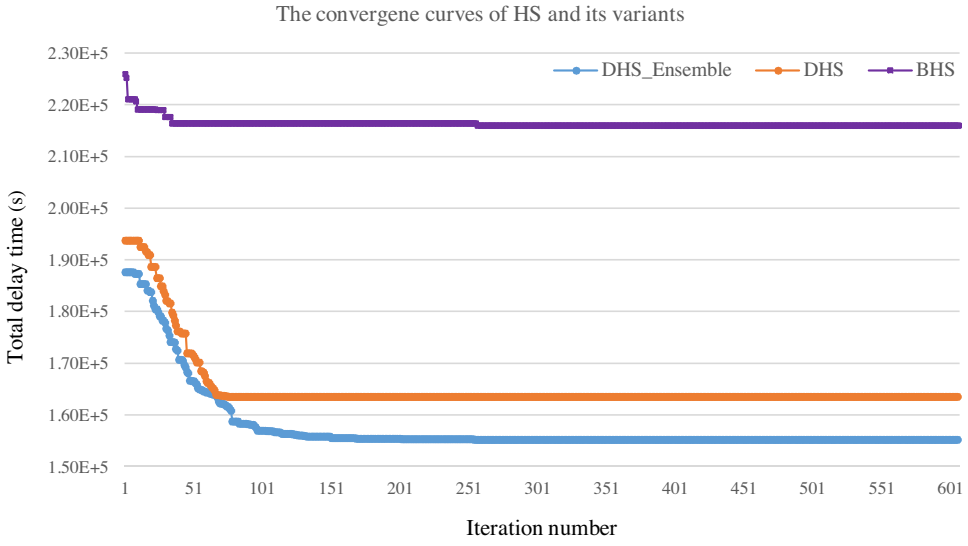

Fig. 11. Convergence curves of HS, DHS and DHS_Ensemble for the largest case.

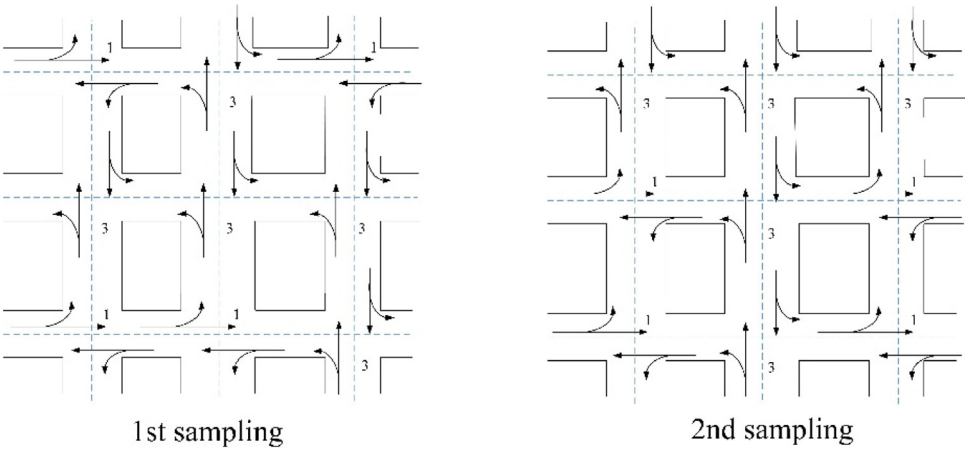

Fig. 12. The schematic view of the best results for 3 × 3 with 30 s time window.

the DHS_Ensemble algorithm proves a greater advantage with the increasing of the time window (30 s in Set one and 60 s in Set two). Regarding the reduction history, the convergence curves of three algorithms for the largest case (i.e., 100 intersections and 60 time window) is shown in Fig. 11. It is clearly that the DHS_Ensemble algorithm has a better convergence rate and better solution quality with respect to the DHS and standard HS algorithms. For further clarify and to show the best results by the DHS with ensemble, the schematic views of two instances (i.e., Case 3 × 3 with 30 and 60 s simulation time windows) are depicted in Figs. 12 and 13.

4.4. Null hypothesis significance testing

To appropriately evaluate the performance of the reported optimizers, two quality indicators are utilized in this paper. The first one is the value-based method which is the solution quality in terms of two performance metrics including the minimum and average total delay time.

The second metric is the nonparametric statistical methods, which have been suggested by different methodologies in the literature. In this paper, the Friedman test which is a nonparametric statistical test with α value of 0.05 (level of significance) is used to

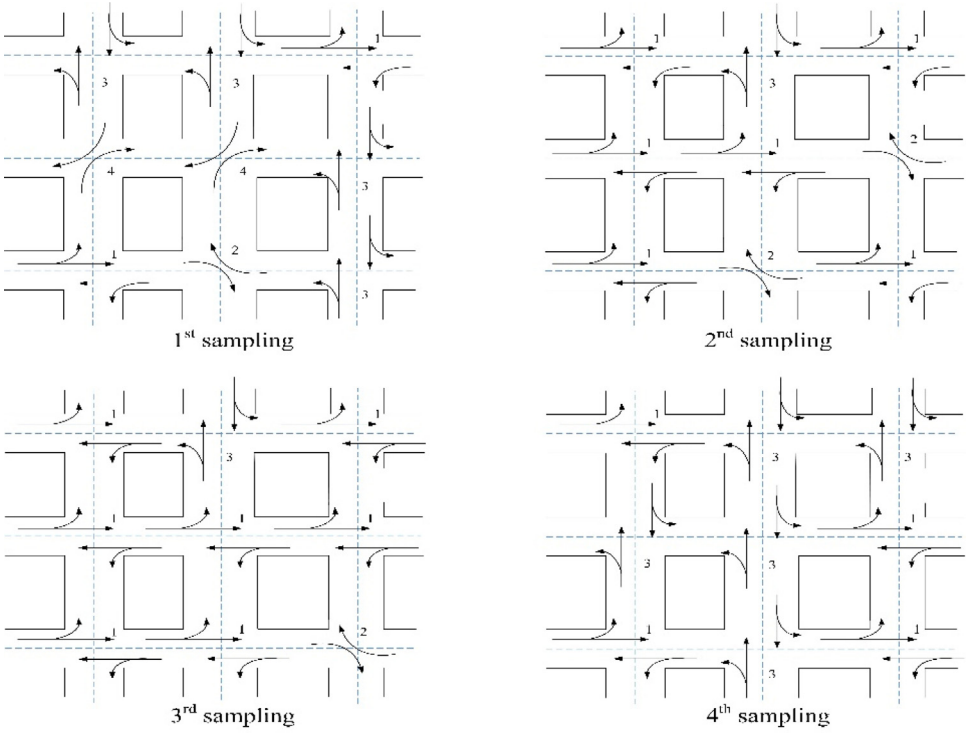

Fig. 13. The schematic view of the best results for 3 × 3 with 60 s time window.

Table 8
Statistical optimization results for eight cases with 30 s time window.

Case	Standard HS (s)		DHS (s)		DHS_Ensemble (s)		P-value ($\alpha = 0.05$)
	Min	Ave	Min	Ave	Min	Ave	
3 × 3	8490	8625	8325	8325	8325	8325	2.55e-17
4 × 4	15,045	15,570	14,175	14,175	14,160	14,160	1.89e-17
5 × 5	21,645	22,350	19,740	19,740	19,650	19,650	6.56e-17
6 × 6	29,940	31,200	26,895	26,910	26,835	26,835	1.96e-17
7 × 7	41,520	42,855	35,820	35,880	35,715	35,730	1.77e-17
8 × 8	54,285	55,020	46,845	46,845	46,590	46,590	2.66e-17
9 × 9	68,895	70,140	59,115	59,130	58,725	58,755	3.37e-17
10 × 10	84,480	85,230	71,835	71,910	71,445	71,580	4.69e-17

distinguish the differences among the reported algorithms. Table 8 shows obtained statistical optimization results along with the *p*-value test using the applied optimizers for different cases with the time window of 30 s.

Based on the obtained statistical optimization results as shown in Table 8, the DHS_Ensemble represents better statistical results in terms of the minimum and average total delay time. Also, looking at the obtained *p*-value using the Friedman test for each experiment, the same conclusion can be obtained. The *p*-value is a measure of how unusual the value of the test statistic is given that the null hypothesis is true. Here, the hypothesis is that there is no difference in the average costs of the algorithms. The null hypothesis is rejected when the *p*-value turns out to be less than a predetermined significance level. The *p*-value is a number between 0 and 1, and for 95% confidence level it is interpreted in the following way:

a) A small *p*-value (typically ≤ 0.05) indicates strong evidence against the null hypothesis, so the null hypothesis is rejected.
b) A large *p*-value (>0.05) indicates weak evidence against the null hypothesis, so it is failed to reject the null hypothesis.
c) p-values very close to the cut off (0.05) are considered to be marginal (could go either way).

The *p*-values obtained from the optimization results given in Table 8 are significantly smaller than the predefined α, which indicates the null hypothesis is rejected at 95% confidence level, meaning that the mean values of total delay time of three algorithms are not the same. In fact, using the Friedman test, there is significance in terms of statistical results using the applied optimizers and it shows that at least one optimization method (i.e., DHS_Ensemble) performs better than the others. Accordingly, Table 9 tabulates the statistical optimization results accompanied with *p*-value obtained by Friedman test for all considered cases with the time window of 60 s.

By observing Table 9, it can be seen that, in terms of statistical optimization results (value-based method), the DHS_Ensemble method shows better performance compared with other optimizers in this paper. The reported *p*-values using the Friedman test for all considered cases (60 s time window) are considerably smaller than the predefined significance value (see Table 9), therefore, the null hypothesis is strongly rejected at the 95% confidence level, and

Table 9
Statistical optimization results for eight cases with 60 s time window.

Case	Standard HS (s)		DHS (s)		DHS_Ensemble (s)		P-value ($\alpha = 0.05$)
	Min	Ave	Min	Ave	Min	Ave	
3×3	22,230	22,740	20,265	20,280	20,100	20,160	5.89e-17
4×4	39,810	40,515	32,910	33,045	32,835	32,835	1.20e-16
5×5	57,810	59,415	45,195	45,720	45,165	45,180	5.83e-13
6×6	81,450	82,590	61,785	62,265	60,450	60,630	9.51e-17
7×7	107,430	109,275	80,970	81,705	77,925	78,420	1.56e-16
8×8	139,920	141,915	105,630	106,530	100,710	100,935	1.61e-16
9×9	175,470	176,685	134,100	136,830	128,040	129,210	1.87e-16
10×10	211,410	216,135	165,855	169,200	155,235	155,850	1.99e-16

there is significance difference among the attained optimization results.

5. Conclusions and future works

In this study, a centralized urban traffic light scheduling problem (UTLSP) is described. A discrete harmony search (DHS) algorithm has been proposed to minimize the network-wise delay time within a fixed time horizon. To improve the DHS algorithm's performance, a novel new harmony improvising method and a mechanism to limit iteration numbers have been proposed. The improved DHS algorithm aims to balance the exploration and exploitation performances. Three neighbourhood structures and corresponding local search operators have been utilized based on the feature of the UTLSP. Moreover, an ensemble of local search operators has been proposed to integrate three local search operators. In the experiment section, the DHS algorithm has been used to solve sixteen cases generated from real-life traffic data in Singapore. The DHS algorithm and its variants with and without local search operators and ensemble have been compared to the fixed cycle traffic light control system (FCS) to show the effectiveness of the proposed methods. The comparisons and discussions have shown that the HS algorithm as a meta-heuristic has a better performance than the FCS. Indeed, the proposed DHS algorithm could considerably improve the results of the standard HS algorithm, and the DHS with ensemble has been proved to achieve the best performance among those achieved by other variants for the UTLSP.

As the future studies, we will extend our work in the following directions: 1) to improve the proposed traffic network flow model by considering more real-time constraints and including event-driven features to well capture drivers' decision processes – the latter makes it possible to bring the discrete-event system theory in the traffic light scheduling framework with all sorts of advanced modelling techniques as illustrated in, e.g., [45–47]; 2) to explore more efficient scheduling algorithms which are able to tackle both high computational complexity and uncertainties.

Acknowledgement

This research is supported by the NTU-NXP Intelligent Transport System Test-Bed Living Lab Fund S15-1105-RF-LLF from the Economic Development Board, Singapore.

References

[1] J. McCrea, S. Moutari, A hybrid macroscopic-based model for traffic flow in road networks, Eur. J. Oper. Res. 207 (1) (2010) 676–684.
[2] J. Saínchez, M. Galaín, E. Rubio, Applying a traffic lights evolutionary optimization technique to a real case: Las Ramblas area in Santa Cruz de Tenerife, IEEE Trans. Evol. Comput. 12 (1) (2008) 25–40.
[3] J.C. Spall, D.C. Chin, Traffic-responsive signal timing for system-wide traffic control, Transport. Res. Part C: Emerging Technol. 5 (3–4) (1997) 153–163.
[4] J. García-Nieto, E. Alba, A. Carolina Olivera, Swarm intelligence for traffic light scheduling: application to real urban areas, Eng. Appl. Artif. Intel. 25 (2) (2012) 274–283.
[5] J. Miller, A computer control system for traffic networks, Proc. 2nd Int. Symp. Traffic Theory (1963) 200–220.
[6] D.I. Robertson, TRANSYT method for area traffic control, Traffic Eng. Control 10 (1969) 276–281.
[7] R.E. Allsop, SIGSET: a computer program for calculating traffic capacity of signal-controlled road junctions, Traffic Eng. Control 12 (1971) 58–60.
[8] R.E. Allsop, SIGCAP: a computer program for assessing the traffic capacity of signal-controlled road junctions, Traffic Eng. Control 17 (1976) 338–341.
[9] G. Improta, G.E. Cantarella, Control systems design for an individual signalized junction, Transp. Res. B 18 (1984) 147–167.
[10] R.A. Vincent, C.P. Young, Self-optimizing traffic signal control using microprocessor: the TRRL MOVA strategy for isolated intersections, Traffic Eng. Control 27 (1986) 385–387.
[11] D.I. Robertson, TRANSYT method for area traffic control, Traffic Eng. Control 10 (1969) 276–281.
[12] M.T. Li, A.C. Gan, Signal timing optimization for oversaturated networks using TRANSYT-7F, 78th Annu. Meeting Transportation Research Board (1999).
[13] P.B. Hunt, D.L. Robertson, R.D. Bretherton, The SCOOT on-line traffic signal optimization technique, Traffic Eng. Control 23 (1982) 190–192.
[14] N.H. Gartner, OPAC: a demand-responsive strategy for traffic signal control, U.S. Dept. Transp., Washington, DC, Transp. Res. Rec. 906 (1983).
[15] J.L. Farges, J.J. Henry, J. Tufal, The PRODYN real-time traffic algorithm, Proc. 4th IFAC Symp. Transportation Systems (1983) 307–312.
[16] F. Boillot, J.M. Blosseville, J.B. Lesort, V. Motyka, M. Papageorgiou, S. Sellam, Optimal signal control of urban traffic networks, Proc. 6th IEE Int. Conf. Road Traffic Monitoring and Control (1992) 75–79.
[17] S. Sen, L. Head, Controlled optimization of phases at an intersection, Transp. Sci. 31 (1997) 5–17.
[18] H. Wang, L. Yan, B. Du, Swarm intelligence for traffic light scheduling. Application to real urban areas, Eng. Appl. Artif. Intel. 25 (2012) 274–283.
[19] T. Brys, T.T. Pham, M.E. Taylor, Distributed learning and multi-objective in traffic light control, Connect. Sci. 26 (1) (2014) 65–83.
[20] S. Gottlich, M. Herty, U. Ziegler, Modeling and optimizing traffic light setting in road networks, Comput. Oper. Res. 55 (2015) 36–51.
[21] C.F. Daganzo, The cell transmission model: a dynamic representation of highway traffic consistent with the hydrodynamic theory, Transport. Res. B Methodol. 28 (4) (1994) 269–287.
[22] C.F. Daganzo, The cell transmission model, part II: network traffic, Transport Res. B-Methodol. 29 (2) (1994) 79–93.
[23] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2) (2001) 60–68.
[24] Z.W. Geem, Optimal cost design of water distribution networks using harmony search, Eng. Optim. 38 (3) (2006) 259–280.
[25] L. Wang, Q.K. Pan, M.F. Tasgetiren, Minimizing the total flow time in a flow shop with blocking by using hybrid harmony search algorithms, Expert Syst. Appl. 37 (12) (2010) 7929–7936.
[26] K.Z. Gao, Q.K. Pan, J.Q. Li, Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion, Int. J. Adv. Manuf. Technol. 56 (5-8) (2011) 683–692.
[27] K.Z. Gao, P.N. Suganthan, Q.K. Pan, T.J. Chua, T.X. Cai, C.S. Chong, Pareto-based grouping harmony search algorithm for multi-objective flexible job shop scheduling, Inform. Sci. 289 (2014) 76–90.
[28] X. Kong, L. Gao, H. Ouyang, S. Li, Solving large-scale multidimensional knapsack problems with a new binary harmony search algorithm, Comput. Oper. Res. 63 (2015) 7–22.
[29] T. Chao, Y. Yan, P. Ma, M. Yang, Y.W. Hu, Optimization of electromagnetic railgun based on orthogonal design method and harmony search algorithm, IEEE Trans. Plasma Sci. 43 (5) (2015) 1546–1554.
[30] M.N. Ambia, H.M. Hasanien, A. Al-Durra, S.M. Muyeen, Harmony search algorithm-based controller parameters optimization for a distributed-generation system, IEEE Trans. Power Deliver. 30 (1) (2015) 246–255.
[31] N. Poursalehi, Development of a new approach evolutionary harmony search algorithm, for the LPO problem, Prog. Nucl. Energy 81 (2015) 78–90.
[32] Alsaadi A. Shanon, W. Tat-Chee, M. Alhamza, Application of harmony search optimization algorithm to improve connectivity in wireless sensor network with non-uniform density, J. Inform. Sci. Eng. 31 (4) (2015) 1475–1489.

[33] S.C. Kumar, G. Shankar, V. Mukherjee, Automatic generation control of power system using a novel quasi-oppositional harmony search algorithm, Int. J. Electr. Power 73 (2015) 787–804.
[34] E.T. Yassen, M. Ayob, M.Z.A. Nazri, N.R. Sabar, Meta-harmony search algorithm for the vehicle routing problem with time windows, Inform. Sci. 325 (2015) 140–158.
[35] M.A. Al-Betar, M.A. Awadallah, A.T. Khader, Z.A. Abdalkareem, Island-based harmony search for optimization problems, Expert Syst. Appl. 42 (4) (2015) 2026–2035.
[36] A. Ouaddah, D. Boughaci, Harmony search algorithm for image reconstruction from projections, Appl. Soft Comput. 46 (2016) 924–935.
[37] B. Zeng, Y. Dong, An improved harmony search based energy-efficient routing algorithm for wireless sensor networks, Appl. Soft Comput. 41 (2016) 135–147.
[38] M. Awadallah, M. Al-Betar, A. Khader, A. Bolaji, M. Alkoffash, Hybridization of harmony search with hill climbing for highly constrained nurse rostering problem, Neural Comput. Appl. (2016), http://dx.doi.org/10.1007/s00521-015-2076-8.
[39] Y.C. Zhang, R. Su, K.Z. Gao, Urban road traffic light real-time scheduling, Proc. of 54th IEEE conference on decision and control (CDC'15), Dec 15–18, 2810-28152015, Osaka, Japan.
[40] Z.W. Geem, Survival of the fittest algorithm or the novelest algorithm? IJAMC 1 (4) (2010) 75–79.
[41] M.P. Saka, O. Hasançebi, Z.W. Geem, Metaheuristics in structural optimization and discussions on harmony search algorithm, Swarm Evol. Comput. 28 (2016) 88–97.
[42] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Technical Report TR06, Computer Engineering Department, Erciyes University, Turkey, 2005.
[43] Q.K. Pan K.Z. Gao, J.Q. Li, A hybrid harmony search algorithm for the no-wait flow shop scheduling problems, Asia Pac. J. Oper. Res. 29 (2) (2012) 1–23, 1250012.
[44] K.Z. Gao, P.N. Suganthan, Q.K. Pan, T.J. Chua, T.X. Cai, C.S. Chong, Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives, J. Intell. Manuf. 27 (2) (2016) 363–374.
[45] S. Forschelen, F.J.M. Van de Mortel, R. Su, J.E. Rooda, Application of supervisory control theory to theme park vehicles, J. Discrete-Event Dyn. Syst. 22 (4) (2012) 511–540.
[46] R. Su, J.H. van Schuppen, J.E. Rooda, Maximum permissive coordinated distributed supervisory control of nondeterministic discrete-event systems, Automatica 48 (7) (2012) 1237–1247.
[47] R. Su, G. Woeginger, String execution time for finite languages: max is easy, min is hard, Automatica 47 (10) (2011) 2326–2329.