University Course Timetabling Using a Hybrid Harmony Search Metaheuristic Algorithm

Mohammed Azmi Al-Betar, Ahamad Tajudin Khader, *Member, IEEE*, and Munir Zaman, *Member, IEEE*

***Abstract*—University course timetabling problem (UCTP) is considered to be a hard combinatorial optimization problem to assign a set of events to a set of rooms and timeslots. Although several methods have been investigated, due to the nature of UCTP, memetic computing techniques have been more effective. A key feature of memetic computing is the hybridization of a population-based global search and the local improvement. Such hybridization is expected to strike a balance between exploration and exploitation of the search space. In this paper, a memetic computing technique that is designed for UCTP, called the hybrid harmony search algorithm (HHSA), is proposed. In HHSA, the harmony search algorithm (HSA), which is a metaheuristic population-based method, has been hybridized by: 1) hill climbing, to improve local exploitation; and 2) a global-best concept of particle swarm optimization to improve convergence. The results were compared against 27 other methods using the 11 datasets of Socha *et al.* comprising five small, five medium, and one large datasets.The proposed method achieved the optimal solution for the small dataset with comparable results for the medium datasets. Furthermore, in the most complex and large datasets, the proposed method achieved the best results.**

***Index Terms*—Harmony search, hill climbing, memetic computing, optimization, particle swarm optimization (PSO), timetabling.**

I. Introduction

UNIVERSITY timetabling is a taxing process for university administration which is required to be repeated every academic semester. In computing terms, university timetabling is a hard combinatorial optimization problem which in most cases has a large search space. This search space houses a large number of local optimal solutions and so does not lend itself to be tackled using classical methods [1]. There are two forms to university timetabling: examination and course timetabling, the latter being the focus of this paper.

University course timetabling problem (UCTP) includes assigning a set of events, each with particular features, to a set of rooms and timeslots on a weekly basis according to a set of constraints [2]. There are two types of constraints: *hard* and *soft*. A timetabling solution *must* satisfy hard constraints in order to be "feasible," but need not satisfy all soft constraints. The quality of a timetabling solution is optimized by *minimizing* the number of soft-constraint violations. The most common soft constraints reflect the preferences of students and teachers. The basic question is, "how to optimize a feasible timetable?"

UCTP has been addressed using several optimization techniques which have been surveyed in [1]. The most appealing have been the metaheuristic-based methods. In [3], the metaheuristics were classified based on the number of solutions that are used at the same time into local search-based (*or* trajectory) methods (i.e., hill climbing, tabu search, simulated annealing) and population-based methods (i.e., genetic algorithm, harmony search algorithm, ant colony optimization).

It is noted that memetic computing techniques have given momentum to UCTP to be even more successful by means of hybridizing a local search-based within a population-based method, which is often referred to as memetic algorithm (MA) [4]–[10]. The MA has come to light as population-based metaheuristic algorithm by means of hybridizing the natural (population-based) and cultural (local search-based) selection principles drawing from the principles of natural selection and the notion of an meme [4]. The meme functions as a local/individual refinement agent in cultural evolution. As an analogy with gene and meme in both biological and cultural selections, the MA makes use of such terms for generality (gene) and problem specificity (meme). The MA is often designed to strike a balance between exploration and exploitation of the search space complementing the advantages of population-based and local search-based methods. This type of hybridization scheme is one of the key features evident of the memetic computing techniques [4].

Harmony search algorithm (HSA) is a metaheuristic population-based method that is developed by Geem *et al*. [11]. Compared with other metaheuristic methods, the HSA has several characteristics: It stipulates fewer mathematical requirements and iteratively generates a new solution after considering all the existing solutions [12]; it has a novel stochastic derivative which reduces the number of iterations that are required to converge toward local minima [13]; it can handle both discrete and continuous variables, and others such as simplicity, flexibility, adaptability, generality, and scalability. The HSA has been successfully applied to a wide variety of optimization problems such as the fractional order controller [14], economic load dispatch [15], and many others that are overviewed in [16]. In some cases, the HSA has outperformed the genetic algorithm (GA) as well as the traditional branch and bound method for combinatorial optimization problems [17]. For the type of combinatorial optimization problems as UCTP, the HSA works well because of its strength in handling exploration and exploitation [18]–[20]. Furthermore, several improved and hybridized variants of the HSA have been proposed [12], [21]–[24], and surveyed in [25].

Manuscript received December 31, 2010; revised July 26, 2011; accepted October 23, 2011. Date of publication January 23, 2012; date of current version August 15, 2012. This paper was recommended by Associate Editor M.-H. Lim.

M. A. Al-Betar is with the Universiti Sains Malaysia (USM), 11800 USM Pulau Pinang, Malaysia, and also with Al-Zaytoonah University of Jordan, Amman 11733, Jordan (e-mail: mohbetar@cs.usm.my).

A. T. Khader and M. Zaman are with the Universiti Sains Malaysia (USM), 11800 USM Pulau Pinang, Malaysia (e-mail: tajudin@cs.usm.my; mzaman@cs.usm.my).

Digital Object Identifier 10.1109/TSMCC.2011.2174356

The authors of [26], [27] mathematically analyzed the evolution of population variance for the HSA and proposed a small but effective amendment to the HSA, thus increasing its explorative power.

The HSA is an iterative improvement method initiated with a number of provisional solutions that are stored in the "harmony memory (HM)." At each iteration, a new solution called "new harmony" is generated that is based on three operators: 1) "memory consideration," which makes use of accumulative search; 2) "random consideration," to diversify the new harmony; and 3) "pitch adjustment," to be analogous to local search. A new harmony is then evaluated against an objective function and substituted with the worst harmony that is stored in HM. This process is repeated until an acceptable solution is obtained.

An initial exploration to adapt the HSA to the UCTP has been presented in [28]. The results were compared with ten other published methods against 11 datasets that established by Socha *et al*. [29]. The datasets comprised five small, five medium, and 1 large datasets. Although the results were promising, a feasible solution was not obtained in the case of the large dataset. The authors identified two shortcomings with adapting the HSA to the UCTP: 1) convergence speed and 2) local exploitation. The authors addressed the first by introducing a new selection mechanism in memory consideration based on the global-best concept stemming from particle swarm optimization (PSO). The second was addressed by designing the pitch adjustment as a local search engine. This local search made limited local changes to the new harmony solution guided by the objective function (i.e., the worse local changes are omitted). To improve the local exploitation further, an HSA with multipitch adjusting rate was proposed in [30]. Each pitch adjusting effects certain local changes to the new harmony with the hope to visit unexplored regions in the search space. However, both works in [28] and [30] did not guarantee to fine tune the new harmony to the local optimal solution in its search space region.

This paper extends the work in [28] and [30]. The proposed memetic computing technique called the hybrid harmony search algorithm (HHSA) has been designed for the specific real-world scheduling problem of UCTP. The hybridization of the HHSA is carried out by: 1) adding a new operator called the hill climbing optimizer (HCO); and 2) incorporating the global-best concept of PSO for the memory consideration process. As hill climbing is a local optimizer, the HCO operator finds a local optimal solution in the search space region of the new harmony. To complement this, the PSO concept provides a targeted selection process, by substituting the random selection of candidate solutions from the HM in the memory consideration operator. This improves the convergence rate. The HHSA aims to explore different regions of the UCTP search space in parallel, and exploit the acquired knowledge in the form of memes (using HCO), to guide the search process toward global optima.

The proposed method has been compared against 27 other published methods which have also used the 11 datasets that are defined by Socha *et al*. [29]. The Socha dataset has been used to evaluate a number of optimization methods over the past several years. Although other datasets exist, the Socha dataset provides a suitably wide range of comparative methods against which the proposed method can be evaluated.

This paper is organized as follows: Section II describes the background of this research, Section III formally defines the UCTP, Section IV provides an overview of the basic HSA, and Section V describes the proposed HHSA method. A small real-world example of the UCTP is illustrated in Section VI. Section VII presents analysis results of the hybridizing components of the HHSA, and compares the results with 27 other methods using the Socha dataset. Section VIII discusses the results and the hybridization method. In Section IX, we present conclusions and suggest future directions that are followed by acknowledgment in the final section.

II. Background

A. Related Works

The UCTP has been widely studied by operational research and artificial intelligence research communities during the past five decades [31]. Several methods have been investigated over the years. The earlier attempts included graph-coloring heuristic methods, where the timetabling solution was constructed by assigning timeslots and rooms for each event, individually, based on the ordering criteria [2]. However, these heuristics are often used nowadays to construct an initial timetabling solution for other iterative improvement methods [32], [33].

The most used methods for the UCTP have been based on metaheuristics [1]. Metaheuristic methods are based on an iterative improvement process, where a set of operators is applied to navigate the search space. Each metaheuristic method has its own type of operators. Generally, metaheuristic methods can be classified based on the number of candidate solutions operated upon at each iteration: 1) local search-based method (e.g., great deluge [34]–[36], variable neighborhood search [32], [37], and random restart local search [29]); and 2) population-based method (e.g., MAX–MIN ant colony [29], artificial immune system [38], and genetic algorithm (GA) [39], [40]).

The process to hybridize a local search-based method within a population-based method has gained much interest [3]. It is aimed to strike a balance between exploration and exploitation of the search space to reap the dividends of both population-based and local search-based methods. This type of hybridization is a key attribute to memetic computing techniques [41] and its importance is emphasized in the timetabling literature. In a recent comprehensive survey to examine timetabling, Qu *et al*. [42] suggested "There are many research directions generated by considering the hybridization of meta-heuristic methods particularly between population-based methods and other approaches."

In general, there are many research trends that highlight the effectiveness to use local search-based methods within population-based methods. For example, Blum and Roli [3] in an influential article on metaheuristics concluded "In summary, population-based methods are better in identifying promising areas in the search space, whereas trajectory methods are better in exploring promising areas in the search space. Thus, metaheuristic hybrids that in some way manage to combine the advantage of population-based methods with the strength of trajectory methods are often very successful."

Population-based methods are able to explore multiple search space regions at a time. However, they are often poorer to find a precise local optimal solution at each search region to which they converge [41], [43]. Local search-based methods are able to fine-tune the search space region to which they converge and find what might be a precise local optimal solution. However, they go through a trajectory in the search space without doing a wider scan of the entire search space [3].

Hybridizing the HCO in the evolutionary algorithms (EA) has been addressed by many timetabling researchers [44]. The motivation behind this hybridization approach is that it restricts the search process to the local optimal solutions instead of the entire search space [45], [46].

The most fruitful results for the UCTP have been obtained by memetic computing techniques, such as hybrid EA with variable neighborhood search [47], hybridized GA with local search [48]–[51], hybridized evolutionary approach with nonlinear great deluge [52], hybrid ant colony systems [53], and hybridized electromagnetism-like mechanism which is a population-based method with great deluge [36]. As we will see in Section VII-B, these methods produced most favored results for the UCTP.

The hyperheuristic approaches have also been investigated. A hyperheuristic is considered to be a heuristic that selects from a set of heuristic methods which include tabu-search hyperheuristic [54], graph-based hyperheuristics [55], and nonlinear great deluge hyperheuristics [35]. Furthermore, in [33] a fuzzy inference rule in fuzzy multiple ordering was employed . Other methods were surveyed in [1] and [56].

B. Harmony Search Algorithm

In [43], an extension to the original HSA was proposed to engineer optimization problems to overcome some shortcomings in the convergence features of the HSA. Their work hybridized the HSA by incorporating sequential quadratic programming as a local optimizer to improve the new harmony. This local optimizer also refined all solutions in the HM. Their results showed that this method outperformed the original HSA.

An idea to hybridize a PSO concept with HSA operators has been presented in [21], which is called the "global-best harmony search." In their work, the *pitch adjustment operator* is modified, where the values of the decision variables are directly selected from the best solution in the HM. The modification of the pitch adjustment operator improved the convergence rate. In [57] a particle swarm HSA was proposed which added a *new operator* for the water network design. Results showed that the method converged significantly faster than the basic HSA, especially for small- or medium-sized networks.

C. Datasets

The context of the UCTP used here, which is formulated in [58], is for postenrolment. It was initially considered an application problem by the Metaheuristics Network (MN).[1]

[1]MN is a European commercial research project that is shared by five European institutions from 2000 to 2004, to investigate the efficiency of different metaheuristics on different combinatorial optimization problems (http://www.metaheuristics.net/).

Evaluating a method to solve UCTP is performed by instantiation with a dataset. A number of such datasets are publicly available including those provided by the First International Timetabling Competition (TTComp2002),[2] the Second International Timetabling Competition (ITC-2007) [59], and researchers of [40] and [29]. These datasets provide a common framework for meaningful comparative evaluations of their methods. The datasets vary in terms of size (e.g., number of rooms, courses, and room types) and constraints.

In TTComp2002, methods were evaluated against 23 postenrolment datasets similar in complexity and size. Not only the methods were evaluated in terms of a feasible timetable with the minimum number of soft-constraint violations, but also the time allowed was limited. The best solution within the time limit was taken as the result of that method—regardless of *whether or not it was completed* [60]. For the ITC-2007, the dataset was modified to bridge the gap between existing research datasets and the real world [61] by means of adding real-world constraints. Two classes of datasets were provided: 1) curriculum based, and 2) postenrolment. Once again, the evaluation was based on the effectiveness of a particular technique in a specified time.

In [40], 60 testing datasets were constructed to stress the capability and performance of a number of efficient methods to solve hard combinatorial timetabling problems. Obtaining a feasible solution for UCTP, where only the hard constraints are satisfied, is by no means trivial. In [62], this dataset was used to evaluate the performance of simulated annealing with Kempe chain to find feasible timetables.

In [29], 11 datasets were defined to measure the performance of the MAX–MIN ant colony algorithm. The 11 datasets are grouped into five small, five medium, and one large. These datasets have been used as a *de facto* dataset for comparative evaluation by several researchers [29], [32]–[37], [47], [51]–[55], [63], [64]. With this plethora of research, the literature has provided us with a readily accessible range of comparators against which to perform any comparative evaluations. This is more useful than the time-limited results from the International Timetabling Competitions, since the objective of the research presented is to produce a feasible timetable with the least number of soft-constraint violations.

Population-based methods, such as the one presented in this paper, generally take much longer time to find the acceptable solution and, thus, will be disadvantaged if time is a factor—especially as time is not a constraint in real-world timetabling. The dataset in [40], being not as widely used, was designed to be exceptionally demanding to test the limits of the research methods. In conclusion, for an initial investigation, the Socha dataset provides the greatest range of nontime limited comparative evaluations, which is the reason why it has been selected to be the dataset against which the proposed method is evaluated.

III. University Course Timetabling Problem

A. Problem Description

A UCTP consists of assigning given events to given rooms and timeslots according to hard and soft constraints, as shown

[2]http://www.idsia.ch/Files/ttcomp2002/.

TABLE I
UCTP CONSTRAINTS

	Hard Constraints		Soft Constraints
H1	**Event Conflict**: Students must not be double booked for events.	S1	**Last timeslot of a day**: A student shall not have an event in the last slot of the day.
H2	**Room availability**: Room capacity and features must be suitable for the assigned events.	S2	**More than two events in a row**: A student shall not have more than two events in a row.
H3	**Room occupancy**: Rooms must not be double booked for events.	S3	**Sole event in a day**: A student shall not have just one event in a day.

in Table I. This problem defines three hard constraints (H1, H2, and H3) which *must* be satisfied for a feasible timetable and three soft constraints (S1, S2, and S3). The basic objective of the UCTP is to minimize the number of soft-constraint violations in a feasible timetable.

The objective function consists of the summation of the number of soft-constraint violations in the feasible timetabling solution.

B. Problem Formulation

The UCTP consists of N events, each of which is attended by a number of students and demands specific room features; a set of M rooms, $\mathcal{R} = \{r_0, r_1, \ldots, r_{M-1}\}$ with each having a specific capacity and features; a set of P timeslots[3], $\mathcal{T} = \{t_0, t_1, \ldots, t_{P-1}\}$; a set of Z room features, $\mathcal{F} = \{z_0, z_1, \ldots, z_{Z-1}\}$; and a set of V students, $\mathcal{S} = \{s_0, s_1, \ldots, s_{V-1}\}$ each attending one or more events.

The notation for UCTP formulation is shown in Table II. A timetabling solution is represented by the vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$ of events, where x_i is a map of room r_j with timeslot t_k for event i:

$$x_i = j \times P + k. \tag{1}$$

For example, if $\boldsymbol{x} = (320, \ldots)$, then event 1 is timetabled in room 7 at timeslot 5 (in the case where $P = 45$). See [28] for details.

C. Definitions

The hard and soft constraints are formally defined as follows.

Definition 1: Event i is feasibly timetabled if and only if the following conditions are met:

H1. $x_i \bmod P \neq x_j \bmod P, \quad \forall x_i, x_j \in \boldsymbol{x} \wedge c_{i,j} = 1 \wedge i \neq j$
H2. $q_{i,j} = 1, \quad j = \lfloor \frac{x_i}{P} \rfloor$
H3. $x_i \neq x_j, \quad \forall x_i, x_j \in \boldsymbol{x} \wedge i \neq j.$

Definition 1 formulates the hard constraints (H1, H2, and H3) for each x_i.

Definition 2: A timetable $\boldsymbol{x}$ is feasible if and only if $\forall x_i \in \boldsymbol{x}$, x_i is feasibly timetabled.

Definition 2 ensures that all events satisfy the hard constraints to define a feasible timetable.

The following define the three soft constraints (S1, S2, and S3).

[3]In general, $P = 45$ (as $D = 5$ working days, each with $H = 9$ timeslots).

TABLE II
NOTATIONS USED TO FORMALIZE THE UCTP

Symbols	Description
D	Number of working days per week.
H	Number of timeslots per each working day.
M	Number of rooms.
N	Number of events.
P	Number of timeslots.
V	Number of students.
Z	Number of features.
$\mathcal{F}$	Set of features, $\mathcal{F} = \{z_i \mid i = 0, \ldots, Z-1\}$.
$\mathcal{R}$	Set of rooms, $\mathcal{R} = \{r_i \mid i = 0, \ldots, M-1\}$.
$\mathcal{S}$	Set of students, $\mathcal{S} = \{s_i \mid i = 0, \ldots, V-1\}$.
$\mathcal{T}$	Set of timeslots, $\mathcal{T} = \{t_i \mid i = 0, \ldots, P-1\}$.
$\boldsymbol{x}$	A timetable solution, $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$.
b_i	Number of students attending to event i.
g_i	Capacity of room i.
x_i	The timeslot and room mapping of event i.
$a_{s,j}$	Student-day matrix element: Whether student s has only one event in a day j. $a_{s,j} = \begin{cases} 1 & \sum_{i=0}^{H-1} sa_{s,i+j\times H} = 1, \quad j \in [0, D-1], \\ 0 & \text{otherwise.} \end{cases}$
$c_{i,j}$	Conflict matrix element: whether event i and event j are in conflict with each other. $c_{i,j} = \begin{cases} 1 & \text{If } u_{k,i} = 1 \wedge u_{k,j} = 1, \quad \exists k \in \mathcal{S}, \\ 0 & \text{otherwise.} \end{cases}$
$q_{i,j}$	Event-room matrix element: whether events i and r_j are compatible in terms of size and features. $q_{i,j} = \begin{cases} 1 & \text{If } (y_{j,k} \geq w_{i,k}) \wedge (b_i \leq g_j), \quad \forall k \in \mathcal{F}, \\ 0 & \text{otherwise.} \end{cases}$
$sa_{j,k}$	Student availability matrix element: whether student s_j in timeslot t_k in the timetable $\boldsymbol{x}$. $sa_{j,k} = \begin{cases} 1 & \exists x_i \in \boldsymbol{x},\ k = x_i \bmod P \wedge u_{j,i} = 1, \\ 0 & \text{otherwise.} \end{cases}$
$u_{i,j}$	Student-event matrix element: Whether student s_j has attend to event j. $u_{i,j} = \begin{cases} 1 & \text{if student } s_i \text{ attends event } j, \\ 0 & \text{otherwise.} \end{cases}$
$w_{i,j}$	Event-feature matrix element: Whether event i requires feature z_j. $w_{i,j} = \begin{cases} 1 & \text{if event } i \text{ requires feature } z_j, \\ 0 & \text{otherwise.} \end{cases}$
$y_{i,j}$	Room-feature matrix element: Whether Room r_i contains feature z_j. $y_{i,j} = \begin{cases} 1 & \text{if room } r_i \text{ contains feature } z_j, \\ 0 & \text{otherwise.} \end{cases}$

Definition 3: Last timeslot of a day (S1). $\forall s \in \mathcal{S}$

$$f_1(\boldsymbol{x}, s) = \sum_{j=0}^{D-1} sa_{s,H\times(j+1)-1}.$$

TABLE III
OPTIMIZATION TERMS IN THE MUSICAL CONTEXT

Musical Terms		Optimisation Terms
Improvisation	$\longleftrightarrow$	Generation or Construction
Harmony	$\longleftrightarrow$	Solution vector
Musician	$\longleftrightarrow$	Decision variable
Pitch	$\longleftrightarrow$	Value
Pitch Range	$\longleftrightarrow$	Value Range
Audio-aesthetic standard	$\longleftrightarrow$	Objective function
Practice	$\longleftrightarrow$	Iteration
Pleasing harmony	$\longleftrightarrow$	(Near-) optimal solution

Definition 4: More than two events in a row (S2). $\forall s \in \mathcal{S}$

$$f_2(\boldsymbol{x}, s) = \sum_{i=0}^{D-1} \sum_{j=0}^{H-3} sa_{s,i\times H+j} \times sa_{s,i\times H+j+1} \times sa_{s,i\times H+j+2}.$$

Definition 5: Sole event in a day (S3). $\forall s \in \mathcal{S}$

$$f_3(\boldsymbol{x}, s) = \sum_{j=0}^{D-1} a_{s,j}.$$

D. Objective Function

Given the notations in Table II, the objective function to evaluate a timetabling solution $\boldsymbol{x}$ for UCTP can be formally defined as in (2). The value of $f(\boldsymbol{x})$ is the total number of soft constraints which are *not met* in a feasible timetable

$$f(\boldsymbol{x}) = \sum_{s\in\mathcal{S}} (f_1(\boldsymbol{x}, s) + f_2(\boldsymbol{x}, s) + f_3(\boldsymbol{x}, s)). \tag{2}$$

The value of $f(\boldsymbol{x})$ is referred to as the penalty value (PV) of a feasible timetable.

IV. OVERVIEW OF THE HARMONY SEARCH ALGORITHM

This section presents a detailed overview of the basics of the HSA as described in [28], [65], and [66].

A. Optimization in a Musical Context

The concepts of the HSA are described in terms of creating a pleasing harmony within a musical context. Table III shows an equivalence between optimization and musical terms of the HSA. In musical improvisation, a group of musicians improvise the pitches of their musical instruments. From repeated practice sessions, a pleasing harmony as decided by their own audio-aesthetic standard is sought. Similarly, in the optimization context, a set of decision variables is assigned with values. From repeated iterations, an optimal solution as decided by an objective function is sought.

When each musician improvises a pitch from his musical instrument, he has three options: 1) improvising a pitch from his memory; 2) modifying a pitch that exists in memory; or 3) improvising a pitch from the possible pitch range. By analogy, in the optimization context, the value of each decision variable is determined according to one of the following options: 1) assigning a value stored in memory; 2) modifying a value that exists in memory; or 3) assigning a value from its feasible range. In [11], these three options were formalized into three operators: memory consideration, pitch adjustment, and random consideration (RC).

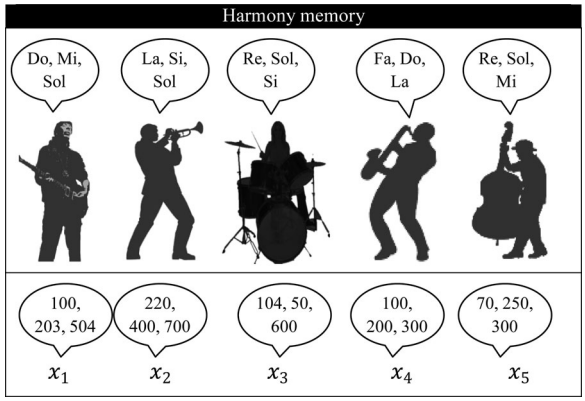

Fig. 1. HM structure.

Fig. 1 shows the harmony memory (HM) structure which is the core of the improvisation process. Consider musical instruments of five musicians on the Jazz bandstand. They have sets of preferable pitches in their memory as follows: Guitarist: $\{Do, Mi, Sol\}$, Trumpeter: $\{La, Si, Sol\}$, Drummer: $\{Re, Sol, Si\}$, Saxophonist: $\{Fa, Do, La\}$, and Double bassist: $\{Re, Sol, Mi\}$. Assume in a practice if Guitarist randomly improvises $\{Do\}$ from his memory, Trumpeter improvises $\{Sol\}$ from his memory, Drummer adjusts $\{Re\}$ from his memory to $\{Fa\}$, Saxophonist improvises $\{La\}$ from his memory, and Double bassist improvises $\{Si\}$ from the available range $\{Do, Re, Mi, Fa, Sol, Si\}$. All these pitches together create a new harmony (Do, Sol, Fa, La, Si) which is estimated by an audio-aesthetic standard. The new harmony substitutes the worst harmony that is stored in the HM, if it is better. This process is repeated until a pleasing harmony is considered to have been reached.

In terms of optimization, consider five decision variables, each of which has stored experience values in the HM as follows: x_1: $\{100, 203, 504\}$, x_2: $\{220,400, 700\}$, x_3: $\{104, 50, 600\}$, x_4: $\{100, 200, 300\}$, and x_5: $\{70, 250, 300\}$. Suppose in an iteration if x_1 is assigned 203 from its memory, x_2 is assigned 400 from its memory, x_3 is adjusted from the value 104 stored in its memory to be 180, x_4 is assigned 200 from its memory, and x_5 is assigned 320 from its feasible range $x_3 \in [0, 600]$. The new constructed solution $(203, 400, 180, 200, 320)$ is evaluated by an objective function. If the new solution is better than the worst solution in the HM, then it replaces the worst solution. This process is repeated until an optimal solution is considered to have been reached.

B. Basic Harmony Search Procedure

Algorithm 1 shows the pseudocode of the basic HSA with five main steps.

Step 1 (Initialize the problem and HSA parameters): Suppose that the discrete optimization problem is modeled as follows:

$$\min_{\boldsymbol{x}\in\mathbf{X}} f(\boldsymbol{x}) \quad \text{s.t.} \quad (g(\boldsymbol{x}) < 0) \wedge (h(\boldsymbol{x}) = 0) \tag{3}$$

Algorithm 1 The basic harmony search algorithm

STEP 1: Initialise the problem and HSA parameters.

1: Input the data instance of the optimisation problem
2: Set the HSA parameters (HMCR, PAR, NI, HMS).

STEP 2: Initialise the harmony memory.

1: Construct vectors of the harmony memory, $\mathbf{HM} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^{\text{HMS}}\}$
2: Recognize the worst vector in **HM**, $\boldsymbol{x}^{\text{worst}} \in \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^{\text{HMS}}\}$

STEP 3: Improvise a new harmony

1: $\boldsymbol{x}' = \phi$ // new harmony vector
2: **for** $i = 1, \cdots, N$ **do**
3: **if** $(U(0,1) \leq \text{HMCR})$ **then**
4: $x_i' \in \{x_i^1, x_i^2, \ldots, x_i^{\text{HMS}}\}$ {memory consideration}
5: **if** $(U(0,1) \leq \text{PAR})$ **then**
6: $x_i' = v_{i,k\pm m}$ { pitch adjustment }
7: **end if**
8: **else**
9: $x_i' \in \boldsymbol{X}_i$ { random consideration }
10: **end if**
11: **end for**

STEP 4: Update the harmony memory

1: **if** $(f(\boldsymbol{x}') < f(\boldsymbol{x}^{\text{worst}}))$ **then**
2: Include $\boldsymbol{x}'$ to the **HM**.
3: Exclude $\boldsymbol{x}^{\text{worst}}$ from **HM**.
4: **end if**

STEP 5: Check the stop criterion

1: **while** (not termination criterion is specified by NI) **do**
2: Repeat **STEP 3** and **STEP 4**
3: **end while**

where $f(\boldsymbol{x})$ is the objective function, and $\boldsymbol{x}$ is the set of each decision variable x_i. $\mathbf{X} = \{\boldsymbol{X}_i | i = 1, \ldots, N\}$ contains all the possible *discrete* values of each decision variable, i.e., $\boldsymbol{X}_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,K_i}\}$. N is the number of decision variables, and K_i is the number of possible values of decision variable x_i. $g(\boldsymbol{x})$ are inequality constraint functions, and $h(\boldsymbol{x})$ are equality constraint functions.

The parameters of the HSA required to solve the optimization problem are also specified in this step: The harmony memory consideration rate (HMCR) is used in the improvisation process to determine whether the value of a decision variable is to be selected from the solutions stored in the HM or randomly selected from the available range of possible values.

The harmony memory size (HMS) is similar to the population size in the GA. The pitch adjustment rate (PAR) decides whether the decision variables are to be adjusted to a neighboring value. The number of improvisations (NI) corresponds to the number of iterations. Note that the HMCR and PAR are the parameters that are used in the improvisation process. These parameters will be explained in more detail in the next steps.

Step 2 (Initialize the HM): The HM contains sets of solution vectors that are determined by the HMS [see (4)]. In this step, these vectors are randomly constructed and stored in the HM according to the values of their objective functions:

$$\text{HM} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_N^1 \\ x_1^2 & x_2^2 & \cdots & x_N^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{\text{HMS}} & x_2^{\text{HMS}} & \cdots & x_N^{\text{HMS}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}^1 \\ \boldsymbol{x}^2 \\ \vdots \\ \boldsymbol{x}^{\text{HMS}} \end{bmatrix}. \quad (4)$$

Step 3 (Improvise a new harmony): In this step, the HSA will construct (or *improvise*) a new harmony vector from scratch, i.e., $\boldsymbol{x}' = (x_1', x_2', \ldots, x_N')$, based on three operators: 1) memory consideration; 2) RC; and 3) pitch adjustment.

Memory consideration: In memory consideration, the value of the first decision variable x_1' is randomly assigned from the historical values, $\{x_1^1, x_1^2, \ldots, x_1^{\text{HMS}}\}$, stored in HM vectors. Values of the other decision variables, $(x_2', x_3', \ldots, x_N')$, are sequentially assigned in the same manner with the probability of (w.p.) HMCR, where HMCR $\in [0, 1]$. The work of this operator is similar to the recombination operator in other population-based methods and is a good source of exploitation [20].

Random consideration: Decision variables that are not assigned values according to memory consideration are randomly assigned according to their possible range by RC with a probability of (1−HMCR) as in

$$x_i' \leftarrow \begin{cases} x_i' \in \{x_i^1, x_i^2, \ldots, x_i^{\text{HMS}}\} & \text{w.p.} \quad \text{HMCR} \\ x_i' \in \boldsymbol{X}_i & \text{w.p.} \quad (1-\text{HMCR}). \end{cases} \quad (5)$$

RC is functionally similar to the mutation operator in the GA which is a source of global exploration in the HSA [20]. The HMCR parameter is the probability to assign one value of a decision variable, x_i', based on the historical values that are stored in the HM. For instance, if HMCR = 0.90, then it means that the value of each decision variable is assigned from historical values that are stored in the HM vectors with the probability of 0.9, and the value of each decision variable is assigned from its possible value range with the probability of 0.1.

Pitch adjustment: Every decision variable x_i' of a new harmony vector, $\boldsymbol{x}' = (x_1', x_2', x_3', \ldots, x_N')$, that has been assigned a value by memory considerations is examined for whether or not it should be pitch adjusted with the probability of PAR (PAR $\in [0, 1]$):

$$\text{Pitch adjust for } x_i'? \leftarrow \begin{cases} \text{Yes} & \text{w.p.} \quad \text{PAR} \\ \text{No} & \text{w.p.} \quad (1-\text{PAR}). \end{cases} \quad (6)$$

A PAR of 0.10 means that the HSA modifies the existing values of decision variables that are assigned by memory consideration with a probability of (PAR × HMCR), while the other values of decision variables that are assigned by memory consideration do not change. If the pitch adjustment decision for x_i' is Yes, the value of x_i' is modified to its neighboring value as follows:

$$x_i'(k) = v_{i,k\pm m} \quad (7)$$

where x_i' is assigned a value $v_{i,k}$, that is, the kth element in $\boldsymbol{X}_i$. m is the neighboring index, $m \in \mathbb{Z}$. The following summarizes

the improvisation process of step 3, which is the main mechanism to iterate toward an optimal solution:

$$x_i' \leftarrow \begin{cases} x_i' \in \{x_i^1, x_i^2, \ldots, x_i^{\mathrm{HMS}}\} & \text{w.p. HMCR} \times (1 - \text{PAR}) \\ x_i' = v_{i,k\pm m} & \text{w.p. HMCR} \times \text{PAR} \\ x_i' \in \{v_{i,1}, \ldots, v_{i,K_i}\} & \text{w.p. } 1-\text{HMCR}. \end{cases} \tag{8}$$

Instead of using the HSA operators that are expressed in (8), the value $v_{i,k}$ of the variable x_i can be alternatively expressed as a novel stochastic derivative [13]:

$$\left.\frac{\partial f}{\partial x_i}\right|_{x_i = v_{i,k}} = \frac{n(x_i = v_{i,k})}{\text{HMS}}(\text{HMCR} \times (1 - \text{PAR})) + \frac{n(x_i = v_{i,k\pm m})}{\text{HMS}}(\text{HMCR} \times \text{PAR}) + \frac{1}{n(K_i)}(1 - \text{HMCR}).$$

The stochastic derivative provides information of probabilistic inclination for the function f which has discrete variables.

Step 4 (Update the HM): If the new harmony vector, i.e., $\boldsymbol{x}' = (x_1', x_2', \ldots, x_N')$, is better than the worst harmony vector in HM, the new harmony vector replaces the worst harmony vector.

Step 5 (Check the stop criterion): Steps 3 and 4 of the HSA are repeated until the stop criterion (the maximum NI) is met. This is specified by the NI parameter.

V. Hybrid Harmony Search Algorithm

This section describes our HHSA for the UCTP. Initially, the adapted HSA for the UCTP is explained as in [28]. Then, the mechanism of how HHSA hybridizes the global-best concept of memory consideration and the HCO is discussed. An illustrative example to apply the HHSA for the UCTP is eventually given.

A. Adapting the Harmony Search Algorithm for the University Course Timetabling Problem

This section provides a brief description of the adapted harmony search for the UCTP as presented in [28]. In order to adapt the HSA to the UCTP, the objective function ($f(\boldsymbol{x})$) in (2) and the timetable representation ($\boldsymbol{x}$) are modeled. The datasets are then transformed into suitable data structures. Finally, the parameter values of the HSA (HMS, NI, HMCR, and PAR) have to be determined.

1) Initialize the Harmony Memory: Initial timetabling solutions are heuristically generated to fill the HM as in (4). It should be emphasized that in this research we only consider the feasible search space region, and thus, the initial timetables in HM must be feasible. The feasibility is preserved for the initial timetables using a combination of three heuristic methods: weighted largest degree (WLD) [67], backtracking algorithm [68], and multiswap algorithm. These methods were discussed in detail in [28].

2) Improvise a New Harmony Solution: This improvisation process is the key step where a new timetabling (*or* a new harmony) solution ($\boldsymbol{x}'$) has to be constructed from scratch based on the three operators: memory consideration, RC, and pitch adjustment. If a complete and feasible new harmony is not obtained, then a repair process, which is described in [28], has to take over. Algorithm 2 describes the pseudocode of the improvisation process for the UCTP.

Algorithm 2 Improvise a new harmony solution

```
1: x' = φ
2: for j = 1...N do
3:    i=Smallest_Positions()
4:    if ( U(0,1) ≤ HMCR) then
5:       x'_i ∈ F_i { F_i = {x_i^j | sp_{i,j} = 1 ∧ j ∈ [1, HMS]}}
6:       rnd = U(0,1)
7:       if (rnd ≤ PAR1) then
8:          Pitch adjustment Move(x'_i)
9:       else if (rnd ≤ PAR2) then
10:         Pitch adjustment Swap-location(x'_i)
11:      else if (rnd ≤ PAR3) then
12:         Pitch adjustment Swap-timeslot(x'_i)
13:      end if
14:   else
15:      x'_i ∈ C_i {C_i = {l | l is feasible for x'_i ∧ l ∈ [0, P × M − 1]}}
16:   end if
17: end for
18: repair_process(x')
```

In order to maintain the feasibility of the new harmony solution, a new feasibility matrix called "smallest position" is proposed to trace the feasibility during the improvisation step. Smallest position is a binary matrix that contains $N \times \text{HMS}$ elements ($sp_{i,j}$) which are assigned as follows:

$$sp_{i,j} \leftarrow \begin{cases} 1 & x_i^j \text{ is feasible for } x_i' \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where x_i^j is the value of x_i (which contains the room-timeslot pair for event i) in solution j that is stored in HM. At each improvisation iteration, this matrix is initialized by 1; it is also updated during the improvisation step when each event is scheduled based on memory consideration or RC, or adjusted by pitch adjustment.

Accordingly, a smallest position algorithm (see Algorithm 2, line 3) is proposed to select the events, individually, based on how difficult they are to be scheduled in the new harmony solution using the smallest position matrix.

Formally, let $\zeta_k = \sum_{j=1}^{\mathrm{HMS}} sp_{k,j}$ be the total number of feasible values for event k to be scheduled in the new harmony solution. The smallest position algorithm selects the event i as follows:

$$i = \arg \min_{k \in [1,N]} \zeta_k.$$

If there is more than one event at each iteration with the same least feasible values, the proposed algorithm selects one event depending on the WLD heuristic.

Memory consideration: Now event i, which is selected using the smallest position algorithm, is scheduled by assigning a

feasible value to x'_i selected from the HM with probability HMCR.Formally, let the set $\digamma_i = \{x_i^j | sp_{i,j} = 1 \wedge j \in [1, \text{HMS}]\}, \forall i \in [1, N]$, contain feasible values for event i to be scheduled in the new harmony solution. The value of x'_i is randomly selected from $\digamma_i$ with probability HMCR. This operator is called a "random memory consideration (RMC)" in [28].

Random Consideration (RC): Now event i, which is selected using the smallest position algorithm, is scheduled by assigning a feasible value to x'_i selected from the entire feasible range with probability (1−HMCR). Formally, let the set

$$\mathcal{C}_i = \{l | l \text{ is feasible for } x'_i \wedge l \in [0, P \times M - 1]\} \tag{10}$$

contain the feasible values for event i. $x'_i = l_i$, where $l_i \in \mathcal{C}_i$ with a probability of (1−HMCR).

Guided Pitch adjustment (GPA): The value (x'_i) of the event i selected based on memory consideration is pitch adjusted with a probability of PAR. In the case of the UCTP, this operator comprises three procedures:

$$\text{Adjust } x'_i \leftarrow \begin{cases} \text{Move} & 0 \le p \le \text{PAR1} \\ \text{Swap-location} & \text{PAR1} < p \le \text{PAR2} \\ \text{Swap-timeslot} & \text{PAR2} < p \le \text{PAR} \\ \text{do nothing} & \text{otherwise} \end{cases} \tag{11}$$

where PAR1=PAR/3, PAR1:PAR2:PAR3 is in the ratio of 1:2:3, and $p \leftarrow U(0,1)$ is a uniform distribution generating a random number between 0 and 1. For each event i scheduled with x'_i out of memory consideration, x'_i is adjusted as follows.

1) *Pitch adjustment: Move*. With the probability range of $[0, \text{PAR1}]$, x'_i moves to any *free* feasible value in the new harmony solution.
2) *Pitch adjustment: Swap-location*. With the probability range of $(\text{PAR1}, \text{PAR2}]$, x'_i exchanges its value with another in the new harmony solution.
3) *Pitch adjustment: Swap-timeslot*. With the probability range of $(\text{PAR2}, \text{PAR3}]$, x'_i is adjusted as follows: Let the set $\mathcal{A} = \{x_j | x_j \bmod P = x'_i \bmod P, \forall j \in [1, N]\}$ contain all events that are scheduled in a new harmony solution $\boldsymbol{x}'$ having the same timeslot as the event x'_i. Let the set $\mathcal{B} = \{x_b | x_b \bmod P = t_k, \forall b \in [1, N]\}$ contain all events scheduled in $\boldsymbol{x}'$ that have the same *randomly* selected timeslot t_k where $(t_k \neq x'_i \bmod P)$. Simply, $\forall x_j \in \mathcal{A}$, $x_j = \lfloor \frac{x_j}{\text{P}} \rfloor \times P + t_k$ and $\forall x_b \in \mathcal{B}$, $x_b = \lfloor \frac{x_b}{\text{P}} \rfloor \times P + x'_i \bmod P$.

In summary, the adjustment performed by any pitch adjustment procedure can only accept the nonworse move. To put it differently, any local change performed by any pitch adjustment procedure must not negatively affect the PV of the new harmony solution. This operator has been defined as "*GPA*" in [28].

B. Hybridization With Global-Best Memory Consideration

In the HHSA, the global-best concept of PSO is utilized in order to modify the selection mechanism of the RMC. The rationale behind this modification is that RMC selects the values of the events randomly, which may exclude the better values from the HM solutions and accordingly affect the convergence rate.

The global-best memory consideration (GMC) always mimics the best solutions that are stored in HM having feasible values for all events, such that

$$B^{\text{best}} = \{x_i^j | j = \arg \min_{k \text{s.t.} x_i^k \in \digamma_i} f(\boldsymbol{x}^k)\} \tag{12}$$

where $\digamma_i = \{x_i^j | sp_{i,j} = 1 \wedge j \in [1, \text{HMS}]\}, \forall i \in [1, N]$, is the set of feasible values for event i that is stored in HM solutions.

In other words, the value of the event x'_i is selected from the best solution, so far stored in HM, that has a feasible value for x'_i such that $x'_i = x_i^j$, where $x_i^j \in B^{\text{best}}$ with probability HMCR.

As a case in point, Table IV shows the timetabling solutions that are stored in HM at the kth iteration. Note that the solutions in HM are sorted in ascending order according to their objective function values.

TABLE IV
TIMETABLING SOLUTIONS STORED IN HM AT THE kTH ITERATION

Solution	x_1	x_2	x_3	x_4	...	x_{N-1}	x_N	$f(\boldsymbol{x})$
$\boldsymbol{x}^1$	14	16	19	31	...	26	22	86
$\boldsymbol{x}^2$	35	1	7	13	...	38	15	97
$\boldsymbol{x}^3$	8	10	18	31	...	42	24	99
$\boldsymbol{x}^4$	0	11	20	30	...	45	26	120
⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮	⋮
$\boldsymbol{x}^{\text{HMS}}$	11	15	34	32	...	10	23	207

Suppose that the value x'_3 of event 3 has to be selected in the new harmony $\boldsymbol{x}' = (x'_1, x'_2, x'_3, \ldots, x'_N)$ based on the GMC. Assuming that x'_3 has feasible values in the solutions $\{\boldsymbol{x}^2, \boldsymbol{x}^3, \ldots, \boldsymbol{x}^{\text{HMS}}\}$, while some other solutions do not have a feasible value for x'_3 such as $\{\boldsymbol{x}^1\}$, the feasible values of x'_3 are $\digamma_3 = \{7, 18, 20, \ldots, 34\}$, while $x_3^1 = 19$ provides no feasible value for x'_3. As such, the value x'_3 of event 3 will be scheduled in 7, such as $x'_3 = 7$ taken from the solution $\boldsymbol{x}^2$. This is because the solutions are sorted in ascending order in HM, and thus, $\boldsymbol{x}^2$ has the lowest objective function value (i.e., the best solution that has a feasible value for x'_3). Hence, the events that are scheduled in the new harmony based on the GMC will be assigned feasible values from the best possible solutions in HM.

C. Hybridization With the Hill Climbing Optimizer

The adapted HSA which is presented in [28] and described in the previous section uses pitch adjustment as a local optimizer. However, the number of local changes is limited by two parameters: N and PAR. The maximum number of local changes is N (assuming PAR =1 and all local changes are accepted). Such being the case, this process cannot be guaranteed to fine-tune the new harmony solution toward the local optimal solution.

Algorithm 3 shows how the HCO is hybridized with the adapted HSA as a new operator. The HCO fine-tunes a new harmony solution $\boldsymbol{x}'$ with a probability of "hill climbing rate (HCR)." The initial solution of the HCO is a new harmony solution $\boldsymbol{x}'$ that is generated by the original HSA operators.

The pseudocode of the HCO operator, which is called by Algorithm 3, is described in Algorithm 4. During the improvement loop in Line 2, the function $Explore(\mathcal{N}(\boldsymbol{x}'))$ navigates

Algorithm 3 HHSA calling the HCO Operator

```
Initialize(HM)
while (Not termination criterion is met) do
    x' = φ
    Improvise a new harmony solution(x') {using GMC + RC + GPA}
    if (U(0,1) < HCR) then
        Hill Climbing Optimiser(x' )
    end if
    Update(HM)
end while
```

Algorithm 4 Hill Climbing Optimiser Function

```
Input(x') {Initial solution}
while (Local optimal solution is not reached) do
    x'' = Explore(N(x'))
    if (f(x'') ≤ f(x')) then
        x' = x''
    end if
end while
```

	8:00 am 8:50 am	9:00 am 9:50 am	10:00 am 10:50 am
Monday	CPT341 (15) DK G31 (20)		CPT340 (20) DK N (20)
			CMT325(17) DK C (20)
Tuesday	CPT314(12) BT 148(15)		
		CPT317 (12) BT 148(15)	
Wednesday	CPT337(15) DK G31(20)		CMT312(13) DK C (20)
	CMT334 (11) DK N (20)		

Fig. 2. Illustrative example of a feasible timetable.

the neighboring solutions $\mathcal{N}(\boldsymbol{x}')$ of $\boldsymbol{x}'$, and moves to the first neighboring solution $\boldsymbol{x}'' \in \mathcal{N}(\boldsymbol{x}')$ which has an equal or lower PV, i.e., $f(\boldsymbol{x}'') \leq f(\boldsymbol{x}')$. This process is repeated until no further improvement is obtained.

The function $Explore(\mathcal{N}(\boldsymbol{x}'))$ navigates the search space of $\boldsymbol{x}'$ using the following neighborhood structure:

- $\mathcal{N}_1$ move an event from a feasible room-timeslot pair to another;
- $\mathcal{N}_2$ swap the room-timeslot pair of two events while the feasibility is maintained;
- $\mathcal{N}_3$ exchange three events in three separate feasible room-timeslots.

Based on the previous discussion, the HCO provides a structured method for local exploitation in the HHSA.

VI. Illustration of Applying the Hybrid Harmony Search Algorithm for the University Course Timetabling Problem

Fig. 2 shows a segment of real-world course timetable for the School of Computer Sciences at the Universiti Sains Malaysia. Each *row* represents a day, each *column* represents a time period, and each cell contains two values that represent a course code and the room allocated. The value in a cell consists of a course code with the number of students who are enrolled (e.g., Course CPT341 with 15 students who are enrolled has the value "CPT341 (15)"), and the allocated room with its capacity (e.g., lecture room DK G31 with a capacity of 20 students has the value "DK G31 (20)").

In Fig. 2, cells of the same color do not have common students and, thus, can be scheduled in the same time period (e.g., courses CPT337 and CMT334 are colored the same). As there are no hard constraints violated, this is, therefore, a feasible timetable. However, if a student was enrolled in CPT337 and CMT334, then the timetable would not be feasible, as a hard constraint (H1) would be violated. If some students enrolled in CMT334 do not have any other courses scheduled that day, then this is an example of a soft constraints (S3) that is not met. However, there is the possibility to generate another feasible timetable, where this soft constraint is not violated. If there are no other additional violations, the PV will be smaller, giving a higher quality solution.

A. Step 1. Initialize the University Course Timetabling Problem and Hybrid Harmony Search Algorithm Parameters

The parameters of the HHSA are initialized as NI = 5000, HMS = 5, HMCR = 0.99, PAR = 0.3, and HCR = 0.3. As shown in Fig. 3, the timetable in Fig. 2 has eight courses $\{x_0, x_1, \ldots, x_7\}$ each with a given number of students, 9 day-periods $\{t_0, t_1, \ldots, t_8\}$, and 4 rooms $\{r_0, r_1, r_2, r_3\}$ with given capacity and features. This is a feasible timetable which is mapped to the vector $\boldsymbol{x} = (0, 11, 20, 30, 31, 6, 15, 26)$ using (1). This vector corresponds to a row in the HM. If we assume that there are in total 120 soft-constraint violations, the PV calculated by (2) is $f(\boldsymbol{x}) = 120$.

B. Step 2. Initialize the Harmony Memory

Fig. 2 can be considered as a single feasible timetable generated by the HHSA. The remaining timetables of the HM are randomly generated as determined by the HMS as shown in Table V. It is noteworthy to point out that the timetables in HM are sorted in ascending order according to their PVs.

C. Step 3. Improvise a New Timetable

In this step, the HHSA generates a new timetable $\boldsymbol{x}' = (x'_0, x'_1, x'_2, \ldots, x'_7)$ as follows:

1) the courses of $(x'_0, x'_1, x'_2, x'_4, x'_6, x'_7)$ are assigned values (i.e., timeslot–room pairs) using the GMC [see (12)];
2) the values of courses $\{x'_2, x'_4\}$ are adjusted using GPA [see (11)];
3) the courses of $\{x'_3, x'_5\}$ are assigned values (i.e., timeslot–room pairs) using RC [see (10)].

The new timetable $\boldsymbol{x}' = (8, 11, 29, 17, 16, 33, 15, 9)$ with penalty of $f(\boldsymbol{x}') = 86$ is generated as shown in Table VI.

Recall that the GMC procedurally selects the values for each event in the new timetable from the best timetable that is stored in HM. If the best solution does not have a feasible value, the

Mapping of timetable to raw vector in HM $\boldsymbol{x}$=(0,11,20,30,31,6,15,26)				
x_i	Course code	t_k	r_j	$x_i = j \times P + k$
x_0	CPT341	t_0	r_0	0×9+0 = 0
x_1	CPT340	t_2	r_1	1×9+2 = 11
x_2	CMT325	t_2	r_2	2×9+2 = 20
x_3	CPT314	t_3	r_3	3×9+3 = 30
x_4	CPT317	t_4	r_3	3×9+4 = 31
x_5	CPT337	t_6	r_0	0×9+6 = 6
x_6	CMT334	t_6	r_1	1×9+6 = 15
x_7	CMT312	t_8	r_2	2×9+8 = 26

KEY

Rooms		
Room (r_j)	Name	Capacity
r_0	DK G31	20
r_1	DK N	20
r_2	DK C	20
r_3	BT 148	15

Timeslot (t_k)	Day-Period
t_0	Monday (8:00 – 8:50)
t_1	Monday (9:00 – 9:50)
t_2	Monday (10:00 – 19:50)
t_3	Tuesday (8:00 – 8:50)
t_4	Tuesday (9:00 – 9:50)
t_5	Tuesday (10:00 – 19:50)
t_6	Wednesday (8:00 – 8:50)
t_7	Wednesday (9:00 – 9:50)
t_8	Wednesday (10:00 – 19:50)

Fig. 3. Representation of the timetable in HM.

TABLE V
HARMONY MEMORY

	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	$f(\boldsymbol{x})$
$\boldsymbol{x}^0$	8	11	20	30	31	6	15	9	72
$\boldsymbol{x}^1$	8	10	19	31	34	11	15	22	86
$\boldsymbol{x}^2$	35	1	7	13	34	12	21	22	97
$\boldsymbol{x}^3$	8	10	18	31	34	11	5	22	99
$\boldsymbol{x}^4$	0	11	20	30	31	6	15	26	120

TABLE VI
IMPROVISATION PROCESS OF THE NEW TIMETABLE $\boldsymbol{x}'$

	GMC	GPA	RC	Possible Values	Selected	x'_i
x'_0	√	–	–	$\{8, 35, 0\}$	8	8
x'_1	√	–	–	$\{11, 10, 1\}$	11	11
x'_2	√	√	–	$\{20, 19, 7, 18\}$	$20 \longrightarrow 29$	29
x'_3	–	–	√	$\{0, 1, \ldots, 35\}$	17	17
x'_4	√	√	–	$\{34, 31\}$	$34 \longrightarrow 16$	16
x'_5	–	–	√	$\{0, 1, \ldots, 35\}$	33	33
x'_6	√	–	–	$\{15, 21, 5\}$	15	15
x'_7	√	–	–	$\{9, 22, 26\}$	9	9

GMC will select the value from the second-best solution stored in HM and so on.

Given that the HCR is met, the new timetable $\boldsymbol{x}' = (8, 11, 29, 17, 16, 33, 15, 9)$ is rapidly converging toward a local optimal solution through the HCO. The new timetable can be $\boldsymbol{x}'' = (23, 9, 0, 10, 30, 33, 25, 1)$ with $f(\boldsymbol{x}'') = 12$.

D. Step 4. Update the Harmony Memory

The new timetable $\boldsymbol{x}''$ is better than the worst timetable in HM $\boldsymbol{x}^{\text{worst}} = (0, 11, 20, 30, 31, 6, 15, 26)$ in Table V. Therefore, HM is updated as shown in Table VII.

E. Step 5. Update the Harmony Memory

Steps 3 and 4 of the HHSA are repeated until NI reaches 5000 iterations.

TABLE VII
UPDATED HM

	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	$f(\boldsymbol{x})$
$\boldsymbol{x}^0$	23	9	0	10	30	33	25	1	12
$\boldsymbol{x}^1$	8	11	20	30	31	6	15	9	72
$\boldsymbol{x}^2$	8	10	19	31	34	11	15	22	86
$\boldsymbol{x}^3$	35	1	7	13	34	12	21	22	97
$\boldsymbol{x}^4$	8	10	18	31	34	11	5	22	99

TABLE VIII
CHARACTERISTICS OF EACH CLASS OF DATASET

Class	Small	Medium	Large
Number of events	100	400	400
Number of rooms	5	10	10
Number of features	5	5	10
Number of timeslots	45	45	45
Approximate features per room	3	3	5
Percentage of the feature used	70%	80%	90%
Number of students	80	200	400
Maximum events per student	20	20	20
Maximum students per event	20	50	100

VII. EXPERIMENTAL RESULTS

The proposed HHSA method was programmed in Microsoft Visual C++ version 6.0 under Windows XP on an Intel 2 GHz Core 2 Quad processor with 2 GB of RAM. The HHSA was tested on the datasets defined by Socha *et al.* [29]. The 11 datasets comprise 5 small, 5 medium, and 1 large datasets with their characteristics summarized in Table VIII. The solution must satisfy the hard constraints and minimize the number of soft-constraint violations (see Table I). The soft-constraint violations define the PV of a solution and correspond to the objective function [see (2)].

A. Analysis of Harmony Search Algorithm Hybridization

Experiments demonstrating the effectiveness to hybridize the basic HSA are presented in this section to investigate the influence of the two hybridizing components: HCO and GMC individually. Remember that the HCO plays a significant role in fine-tuning the search space region of the new harmony toward the local optima improving the local exploitation capability of the HHSA.In addition, the GMC focuses on the best solutions in HM which provide a controlled selection mechanism leading to improving convergence rate.

1) Effect of Global-Best Memory Consideration on the Convergence of the Hybrid Harmony Search Algorithm: Experiments were conducted to compare the differences between the HHSA that has an RMC and the HHSA that has a GMC. This is to show the advantage of using GMC over RMC when both are used in the HHSA by comparing their results as shown in Table IX. As before, ten runs were conducted for each dataset, and the parameter sets were NI = 1500, HMS = 10, HMCR = 0.99, PAR = 0.6, and HCR = 0.3. The best solution for each dataset is highlighted in bold. Table IX show that in all cases the HHSA with GMC produces a better solution than HHSA with RMC.

A comparison of the convergence rate between the two (i.e., HHSA with GMC and HHSA with RMC) was conducted using the medium and large datasets. Fig. 4 plots the average PV for *all* ten solutions in the HM at each iteration for one of the ten runs. The run that is selected for this comparison was any of the ten runs. The results in this instance show a much greater convergence rate for the method that contains the GMC operator.

2) Effect of HCO on the Performance of the Hybrid Harmony Search Algorithm: Experiments that investigate the effect of various HCR values on the performance of the HHSA were conducted. Recall that HCR is the probability to use the HCO at each iteration (see Section V-C). The effect of the PVs at three HCR values, of 5%, 30%, and 80%, was investigated using the Socha dataset. This is to evaluate the influence of HCO when it is used with small, medium, and large values of HCR. Note that the other parameters were NI = 1500, HMS = 10, HMCR = 0.99, and PAR = 0.6. These values were chosen from the parameter analysis that is conducted in [28]. Table X summarizes the results of various HCR values on the PV of the solution by showing the best, worst, upper, and lower quartile, and median over ten runs. The best solution for each dataset is highlighted in bold, while the data in the table are displayed as boxplots in Fig. 5. Fig. 5(a) provides a key to the boxplot notation. The results suggest that increasing the HCR value generally improves the solutions that are obtained.

TABLE IX
COMPARISON OF THE EFFECT OF GMC OVER RMC

		HHSA with GMC	HHSA with RMC
Small1	Best	**0**	6
	Average	1.7	7
	Worst	3	8
	Std.dev	1.16	0.47
Small2	Best	**0**	6
	Average	1.2	7.5
	Worst	3	9
	Std.dev.	1.23	0.97
Small3	Best	**0**	3
	Average	1.6	4.1
	Worst	3	5
	Std.dev	0.97	0.57
Small4	Best	**0**	2
	Average	2.3	4.7
	Worst	5	7
	Std.dev.	2	1.49
Small5	Best	**0**	0
	Average	0	0.3
	Worst	0	1
	Std.dev.	0	0.48
Medium1	Best	**118**	254
	Average	137.8	267.1
	Worst	155	283
	Std.dev.	12.57	8.962
Medium2	Best	**102**	260
	Average	121.3	274.4
	Worst	138	281
	Std.dev.	11.31	6.077
Medium3	Best	**178**	335
	Average	199.5	363.3
	Worst	216	374
	Std.dev.	11.05	10.77
Medium4	Best	**112**	263
	Average	131	272
	Worst	143	281
	Std.dev.	8.68	6.73
Medium5	Best	**77**	199
	Average	86.2	227.5
	Worst	94	256
	Std.dev.	6.53	19.15
Large	Best	**435**	-
	Average	464	-
	Worst	482	-
	Std.dev.	15.75	-

B. Comparative Evaluation

The proposed method was compared with all published methods using the Socha dataset, known and available to the authors. This provided a total of 27 comparator methods, grouped according to the algorithm type (see Table XI).

Table XII shows the results of the HHSA in comparison with other comparator methods. The numbers in the table indicate the number of soft-constraint violations which define the PV. The indicator "–" shows where the method did not provide a feasible timetable (e.g., a hard constraint was not met). The numbers in bold font show the best solution that is obtained for that dataset (lowest is best). The results show that for all five small datasets, which is the simplest case, the proposed method achieves a PV of zero, as do a number of other comparative methods.

The key results, for the medium and large datasets, are summarized in bar charts. Fig. 6 is a bar chart showing the PVs of each method for the five medium datasets. For the medium dataset, out of the 28 methods that are evaluated (including HHSA), the proposed method has ranked the sixth, first, third, seventh, and first positions. The results show that the HHSA has been able to achieve comparably good results for this class of dataset.

Fig. 7 is a bar chart showing the PVs of each method for the single "large" dataset. The proposed method achieves the best solution, providing a feasible timetable with the least number

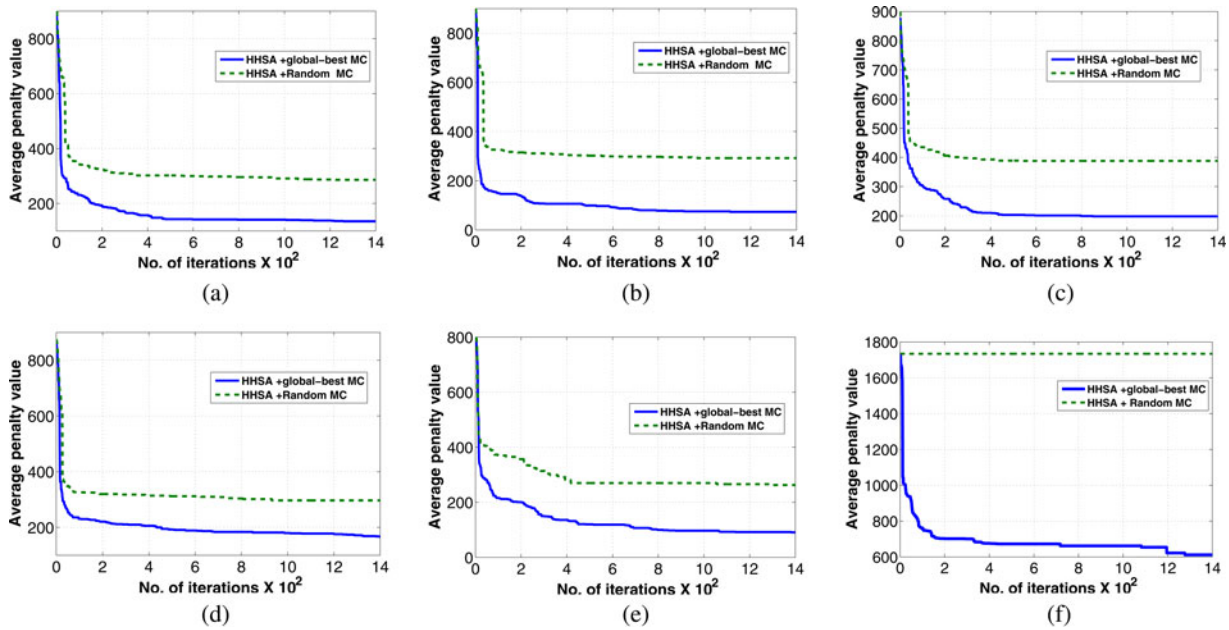

Fig. 4. Comparison of convergence rate between RMC and GMC operators. The average PV of all the HM solutions for a random run (HCR=30%, medium 1 through large datasets) is plotted against the number of iterations. The plots show the faster convergence rate of GMC over RMC. (a) Medium 1. (b) Medium 2. (c) Medium 3. (d) Medium 4. (e) Medium 5. (f) Large.

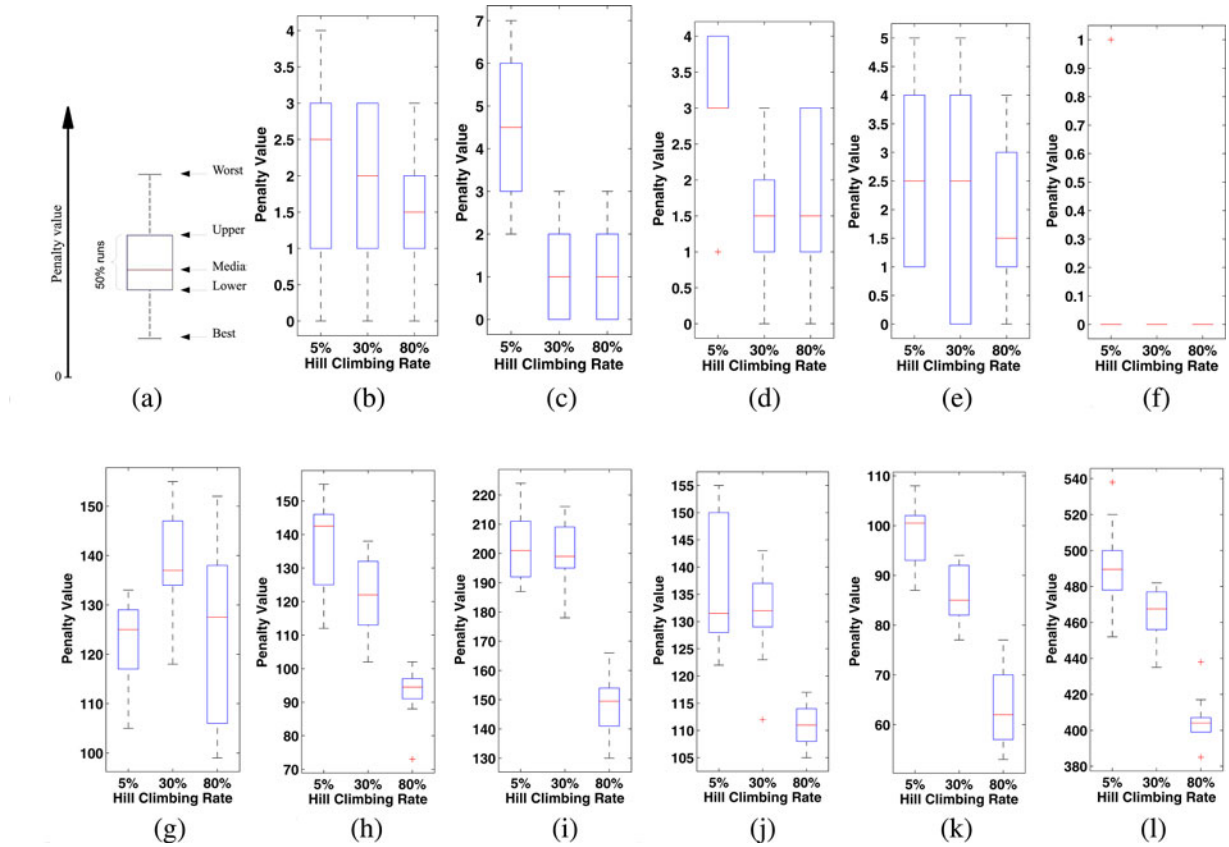

Fig. 5. Effect of various HCR values on HHSA performance. (a) Boxplot key. (b) Small 1. (c) Small 2. (d) Small 3. (e) Small 4. (f) Small 5. (g) Medium 1. (h) Medium 2. (i) Medium 3. (j) Medium 4. (k) Medium 5. (l) Large.

TABLE X
EFFECT OF VARIOUS HCR VALUES ON HHSA PERFORMANCE

DATASET		HCR 5%	HCR 30%	HCR 80
small1	Best	**0**	**0**	**0**
	Q1	1	1	1
	Median	2	2	1
	Q3	3	3	2
	Worst	4	3	3
Small2	Best	2	**0**	**0**
	Q1	3	0	1
	Median	4	1	1
	Q3	6	2	2
	Worst	7	3	3
Small3	Best	1	**0**	**0**
	Q1	3	1	1
	Median	3	2	2
	Q3	4	2	3
	Worst	4	3	3
Small 4	Best	1	**0**	**0**
	Q1	1	0	1
	Median	3	3	1
	Q3	4	4	3
	Worst	5	5	4
Small 5	Best	**0**	**0**	**0**
	Q1	0	0	0
	Median	0	0	0
	Q3	0	0	0
	Worst	1	0	0
Medium 1	best	105	118	**99**
	Q1	118.25	134.25	109
	Median	125	137	127.5
	Q3	128.5	145.25	137.75
	Worst	133	155	152
Medium 2	best	112	102	**73**
	Q1	125	113	91
	Median	142	121	93
	Q3	145	132	96
	Worst	155	138	102
Medium 3	best	187	178	**130**
	Q1	193	195	143
	Median	205	199	152
	Q3	211	209	154
	Worst	224	216	166
Medium 4	best	122	112	**105**
	Q1	128	129	108
	Median	130	131	112
	Q3	134	136	114
	Worst	155	143	117
Medium 5	best	87	77	**53**
	Q1	92	82	59
	Median	97	85	64
	Q3	102	92	70
	Worst	108	94	77
Large	best	452	435	**385**
	Q1	480.75	456.5	399
	Median	489.5	467.5	404
	Q3	498.5	475.5	407
	Worst	538	482	438

of soft-constraint violations. Note that six comparator methods were unable to obtain a feasible solution for the large dataset.

VIII. DISCUSSION

In memetic computing, the term meme refers to a unit of cultural evolution that is capable of local/individual refinement [41]. In the biological context, the gene is refined using global improvement techniques with limited local improvements. To perfect such improvement, the meme in the cultural context provides a mechanism to overcome the aforementioned limitation. The combination of biological and cultural mechanisms provides a suitable balance between generality (global improvement techniques) and problem specificity by virtue of the meme. The HSA is an EA that has a global search corresponding to the biological mechanism. It has been empowered thanks to HCO which provides local refinements corresponding to the cultural mechanism. Through hybridization, the HHSA has complemented the advantages of population-based methods and local optimizer analogues to biological and cultural mechanisms, respectively.

A. Hybridization in the Hybrid Harmony Search Algorithm

The adapted HSA for the UCTP in [28] was not able to obtain comparatively good results because of the weakness in the local exploitation due to the preference of the HSA toward exploration over exploitation. This is because pitch adjustment and RC are performed by an unguided random process. The HSA is hybridized by the HCO in order to provide a more structured local exploitation through gradient descent. This provides an improved balance between global exploration and local exploitation of the search space. This hybridization is indeed the key feature of memetic computing. However, simply hybridizing the adapted HSA with HCO did not provide a sufficient improvement because HCO interfered with the solutions in the HM, which affects the rate of convergence.

A new parameter, the HCR, determines the rate of any gradient descent. The variation on the HCR value showed that the larger the HCR, the better the solution. The boxplots in Fig. 5 show the distribution of results from ten runs for the "small 1" to "small 5" datasets. Remarkably, since all ten runs for "small 5" dataset were able to achieve the exact solution, such a dataset was excluded from distribution. Although in some cases increasing the HCR did not improve the median (e.g., for the "medium 1" and "medium 4" datasets), there was a general tendency toward an increase in the HCR to produce a better solution. Interestingly, for the most difficult large dataset, the range of results was much narrower and better with the increase of HCR values.

A second hybridizing component, which is based on PSO concepts, substituted the memory consideration operator, in order to improve the rate of convergence. PSO initializes with a population of candidate solutions called a *swarm*. Each candidate solution is called a *particle* which iteratively flies over the search space. At each iteration, each particle is attracted to the position of the best solution found previously (*local best*). The

TABLE XI
KEY TO THE COMPARATOR METHODS

Algorithm Type	No.	Key	Method	Reference
HSA based	1	HHSA	Hybrid Harmony Search Algorithm	<*proposed method*>
	2	AHSA	Adapted Harmony Search Algorithm	(2010) [63]
	3	MHSA	Modified Harmony Search Algorithm	(2010) [28]
	4	HSA-MPAR	Harmony Search Algorithm With Multi Pitch Adjusting Rate	(2010) [30]
Heuristic based	5	RRLS	Random Restart Local search	(2002) [29]
	6	MMAS	MAX-MIN Ant System	(2002) [29]
	7	THH	Tabu-search Hyper-Heuristic	(2003) [54]
	8	VNS	Variable Neighborhood Search	(2005) [32]
	9	FMHO	Fuzzy Multiple Heuristic Ordering	(2005) [33]
	10	DCFHH	Distributed Choice Function Hyperheuristics	(2005) [69]
	11	GHH	Graph-based Hyper-Heuristic	(2007) [55]
	12	RII	Randomized Iterative Improvement	(2007) [37]
	13	GD	Great Deluge	(2008) [64]
	14	NGD	Non-linear Great Deluge	(2008) [64]
	15	NGDHH-SM	Non-linear Great Deluge Hyper-Heuristic -Static Memory	(2009) [35]
	16	NGDHH-DM	Non-linear Great Deluge Hyper-Heuristic-Dynamic Memory	(2009) [35]
	17	GDTS	Great Deluge and Tabu Search	(2009) [70]
	18	PCA	Particle Collision Algorithm	(2009) [71]
	19	LARD	Late Acceptance Randomized Descent	(2010) [72]
Memetic based	20	EGD	Extended Great Deluge	(2007) [34]
	21	HEA	Hybrid Evolutionary Approach	(2007) [47]
	22	MA	Memetic Algorithm	(2008) [73]
	23	ENGD	Evolutionary Non-linear Great Deluge	(2009) [52]
	24	EMGD	Electromagnetism Mechanism Great Deluge	(2009) [36]
	25	GGA	Guided Genetic Algorithm	(2009) [74]
	26	ACS-SA	Ant Colony System with Simulated Annealing	(2009) [53]
	27	ACS-TS	Ant Colony System with Tabu Search	(2009) [53]
	28	GAGLS	Genetic Algorithms With Guided and Local Search	(2010) [51]

TABLE XII
COMPARATIVE RESULTS

Algorithm Type	No.	Key	Small 1	Small 2	Small 3	Small 4	Small 5	Medium 1	Medium 2	Medium 3	Medium 4	Medium 5	Large
HSA based	1.	HHSA(*best*)	**0**	**0**	**0**	**0**	**0**	99	**73**	130	105	**53**	**385**
	2.	AHSA(*best*)	3	2	4	3	**0**	223	216	272	202	177	–
	3.	MHSA(*best*)	**0**	**0**	**0**	**0**	**0**	168	160	176	144	71	417
	4.	HSA-MPAR(*best*)	**0**	**0**	**0**	**0**	**0**	124	117	148	132	67	424
Heuristic based	5.	RRLS *(avg.)*	8	11	8	7	5	199	202.5	–	177.5	–	–
	6.	MMAS *(avg.)*	1	3	1	1	**0**	195	184	284	164.5	219.5	851.5
	7.	THH *(best)*	1	2	**0**	1	**0**	146	173	267	169	303	1166
	8.	VNS *(best)*	**0**	**0**	**0**	**0**	**0**	317	313	375	247	292	–
	9.	FMHO *(best)*	10	9	7	17	7	243	325	249	285	132	1138
	10.	DCFHH *(best)*	1	3	1	1	**0**	182	164	250	168	222	–
	11.	GHH *(best)*	6	7	3	3	4	372	419	359	348	171	1068
	12.	RII *(best)*	**0**	**0**	**0**	**0**	**0**	242	161	265	181	151	–
	13.	GD *(best)*	17	15	24	21	5	201	190	229	154	222	1066
	14.	NGD *(best)*	3	4	6	6	**0**	140	130	189	112	141	876
	15.	NGDHH-SM *(best)*	**0**	**0**	**0**	**0**	**0**	**71**	82	137	**55**	106	777
	16.	NGDHH-DM *(best)*	**0**	**0**	**0**	**0**	**0**	88	88	**112**	84	103	915
	17.	GDTS *(best)*	**0**	**0**	**0**	**0**	**0**	78	92	135	75	68	556
	18.	PCA *(best)*	1	1	1	1	**0**	136	138	165	143	135	789
	19.	LARD *(best)*	**0**	**0**	**0**	**0**	**0**	149	132	200	138	173	855
Memetic based	20.	EGD *(best)*	**0**	**0**	**0**	**0**	**0**	80	105	139	88	88	730
	21.	HEA *(best)*	**0**	**0**	**0**	**0**	**0**	221	147	246	165	130	529
	22.	MA *(best)*	**0**	**0**	**0**	**0**	**0**	227	180	235	142	200	–
	23.	ENGD *(best)*	**0**	1	**0**	**0**	**0**	126	123	185	116	129	821
	24.	EMGD *(best)*	**0**	**0**	**0**	**0**	**0**	96	96	135	79	87	683
	25.	GGA *(best)*	**0**	**0**	**0**	**0**	**0**	240	160	242	158	124	801
	26.	ACS-SA *(best)*	**0**	**0**	**0**	**0**	**0**	117	121	158	124	134	645
	27.	ACS-TS *(best)*	**0**	**0**	**0**	**0**	**0**	150	179	183	140	152	750
	28.	GAGLS *(best)*	**0**	**0**	**0**	**0**	**0**	139	92	122	98	116	615

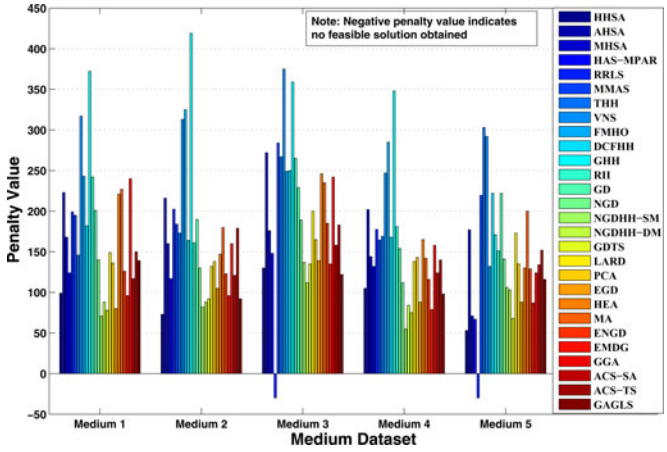

Fig. 6. PVs of comparative methods using the medium dataset.

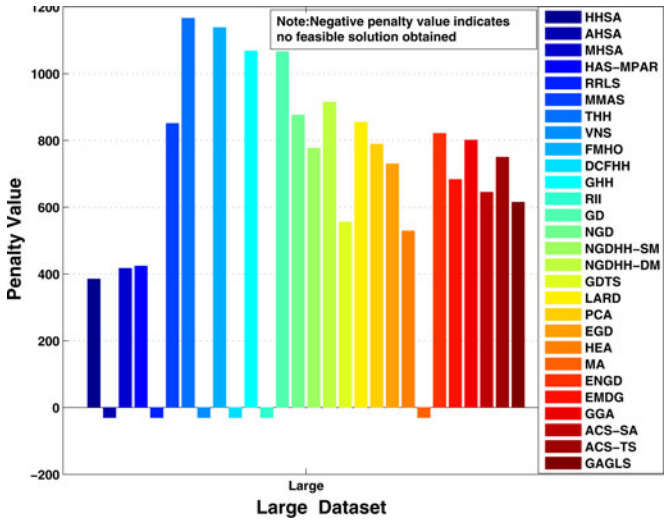

Fig. 7. PVs of comparative methods using the large dataset.

position of the best quality particle in the entire swarm is the *global best* [75].

The new operator, which is called the GMC, substituted the RMC, thus improving the rate of convergence by tracking the best solutions in HM using the global-best method. Other solutions in the HM are attracted to the global best, thereby directing the improvisation process. The difference in the rate of convergence is shown in Fig. 4 and seems significant.

B. Comparative Analysis

The HHSA was evaluated in terms of obtaining the feasible solution with the least number of soft-constraint violations—with no running time limitation. This is more realistic as running time is not a major constraint since timetabling is not conducted frequently enough to require methods to provide realtime solutions [42]. However, it may be of interest to some that the slowest run took less than 6 h to complete.

The HHSA provides the exact solutions for all five small datasets (see Table XII). This is not a hard dataset as 17 others also obtain an exact solution (MHSA, HSA-MPAR, VNS, RII, HEA, NGDHH-SM, NGDHH-SM, GDTS, LARD, EGD, HEA, MA, EMDG, GGA, ACS-SA, ACS-TS, and GAGLS). For the medium dataset, the HHSA has been able to achieve the best solution for the "medium 2" and "medium 5" datasets. For the remaining medium datasets, the proposed method is able to provide a respectable solution which is more easily visualized in Fig. 6. It is noted that the RRLS method was not able to find a feasible timetable for "medium 3" and "medium 5" datasets.

For the large dataset, the HHSA achieved the best solution (PV = 385), which is closely followed by MHSA (PV = 417) and HSA-MPAR (PV = 424). Notably, six methods were unable to find a feasible timetable (HSA, RRLS, VNS, and RII). It was noted by Burke *et al*. [55] that the "medium 5" and "large" datasets were the most taxing, as was borne out by the results. Interestingly, the HHSA has been able to achieve the best solutions for these datasets.

The best result obtained for the large dataset before applying the HSA-based methods (i.e., AHSA, MHSA, and HSA-MPAR) was achieved by HEA, which is a hybridized variable neighborhood structure (VNS) with the EA. Moreover, comparator methods that used the same concepts of memetic computing have been separately grouped to memetic based at the bottom of Table XII (i.e., EGD, HEA, MA, ENGD, EMDG, GGA, ACS-SA, ACS-TS, and GAGLS). Notably, these methods produced a high-quality solutions for all Socha datasets—especially the most complex and large ones.

It can be observed that the hybridization of the HSA as a global improvement search strategy with HCO as a local improvement search strategy, combined with auxiliary concepts as global best of PSO, provided a superior algorithm. This hybrid method stands out for its ability to strike a balance between global exploration and local exploitation in the search space. In a nutshell, *"Mixing and hybridizing is often better than purity"* [3].

IX. Conclusion and Future Work

In this paper, a memetic computing technique designed for the UCTP, which is called the HHSA, has been proposed. The HHSA was hybridized by virtue of HCO and global-best concepts of PSO within the HSA. Experimental analysis of hybridization using hill climbing as a local optimizer, in conjunction with the GMC as a convergence accelerator, showed significant improvement in performance over previous HSA-based methods contributed in [28] and [30].

The method was evaluated using the 11 datasets that were provided by Socha *et al*. [29] (5 small, 5 medium, and 1 large), thus providing 27 comparative methods. The results of the proposed method showed that it achieved a feasible timetable with no constraint violations for small datasets. For the medium datasets, the method achieved comparable results, while for the large dataset the proposed method achieved a feasible timetable with the least number of soft-constraint violations, thereby providing the best solution in comparison wit other comparative methods. This shows that hybridization between local search algorithms and population-based algorithms based on the HSA is a

promising research area that has the potential to produce good results for the most difficult problem instances.

As HHSA has been shown to be successful in finding the best solution for the most difficult datasets, future work can investigate the success of this algorithm further by testing dataset such as that defined in [39]. Other means of hybridization of the HSA, such as simulated annealing acceptance criteria to escape local optima, can also be investigated. Quite recently, an improved HSA technique that does not require parameter settings has been developed [76], [77]. This idea can be incorporated in the HHSA to simplify implementation.

ACKNOWLEDGMENT

The authors sincerely appreciate the helpful and insightful comments from the anonymous referees, which have greatly improved the clarity of the paper. The first author is grateful to a Postdoctoral Fellowship from the School of Computer Sciences, USM. They would like to thank Prof. L. C. Peng, M. A. Awadallah, and A. L. Bolaji for their valuable help and suggestions.

REFERENCES

[1] R. Lewis, "A survey of metaheuristic-based techniques for university timetabling problems," *OR Spectrum*, vol. 30, pp. 167–190, 2008.

[2] E. K. Burke, D. de Werra, and J. Kingston, "Applications to timetabling," in *Handbook of Graph Theory*, J. L. Gross and J. Yellen, Eds. London, U.K.: CRC Press, 2004, pp. 445–474.

[3] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, 2003.

[4] Y.-S. Ong, M.-H. Lim, N. Zhu, and K.-W. Wong, "Classification of adaptive memetic algorithms: A comparative study," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 1, pp. 141–152, Feb. 2006.

[5] J. Tang, M. Lim, and Y.-S. Ong, "Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems," *Soft Comput.—Fusion Found., Methodol. Appl.*, vol. 11, pp. 873–888, 2007.

[6] S. Handoko, C. K. Kwoh, and Y.-S. Ong, "Feasibility structure modeling: An effective chaperone for constrained memetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 14, no. 5, pp. 740–758, Oct. 2010.

[7] Y. S. Ong and A. Keane, "Meta-lamarckian learning in memetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 99–110, Apr. 2004.

[8] Q. H. Nguyen, Y.-S. Ong, and M. H. Lim, "A probabilistic memetic framework," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 604–623, Jun. 2009.

[9] R. Meuth, M.-H. Lim, Y.-S. Ong, and D. Wunsch, "A proposition on memes and meta-memes in computing for higher-order learning," *Memetic Comput.*, vol. 1, pp. 85–100, 2009.

[10] N. Q. Huy, O. Y. Soon, L. M. Hiot, and N. Krasnogor, "Adaptive cellular memetic algorithms," *Evol. Comput.*, vol. 17, no. 2, pp. 231–256, 2009.

[11] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[12] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Appl. Math. Comput.*, vol. 188, no. 2, pp. 1567–1579, 2007.

[13] Z. W. Geem, "Novel derivative of harmony search algorithm for discrete design variables," *Appl. Math. Comput.*, vol. 199, no. 1, pp. 223–230, 2008.

[14] G. G. Roy, P. Chakraborty, and S. Das, "Designing fractional-order piλdμ controller using differential harmony search algorithm," *Int. J. Bio-Inspired Comput.*, vol. 2, no. 5, pp. 303–309, 2010.

[15] B. K. Panigrahi, V. R. Pandi, S. Das, and Z. Cui, "Dynamic economic load dispatch with wind energy using modified harmony search," *Int. J. Bio-Inspired Comput.*, vol. 2, no. 3/4, pp. 303–309, 2010.

[16] G. Ingram and T. Zhang, "Overview of applications and developments in the harmony search algorithm," in *Music-Inspired Harmony Search Algorithm*, Z. W. Geem, Ed. Berlin, Germany: Springer-Verlag, 2009, pp. 15–37.

[17] Z. W. Geem, "Harmony search in water pump switching problem," in *Advances in Natural Computation* (Lecture Notes in Computer Science vol. 3612), K. C. L. Wang and Y. Ong, Eds. Berlin, Germany: Springer-Verlag, 2005, pp. 751–760.

[18] M. A. Al Betar, A. T. Khader, and J. J. Thomas, "A combination of metaheuristic components based on harmony search for the uncapacitated examination timetabling," presented at the Proc. 8th Int. Conf. Practice Theory Automat. Timetabling, Belfast, Northern Ireland, Aug. 2010.

[19] M. A. Al-Betar, A. T. Khader, and F. Nadi, "Selection mechanisms in memory consideration for examination timetabling with harmony search," presented at the Proc. Genetic and Evolutionary Computation Conf., Portland, OR, Jul. 2010.

[20] X.-S. Yang, "Harmony search as a metaheuristic algorithm," in *Music-Inspired Harmony Search Algorithm*, Z. W. Geem, Ed. Berlin, Germany: Springer-Verlag, 2009, pp. 1–14.

[21] M. G. H. Omran and M. Mahdavi, "Global-best harmony search," *Appl. Math. Comput.*, vol. 198, no. 2, pp. 643–656, 2008.

[22] V. R. Pandi, B. K. Panigrahi, R. C. Bansal, S. Das, and A. Mohapatra, "Economic load dispatch using hybrid swarm intelligence based harmony search algorithm," *Electr. Power Compon. Syst.*, vol. 39, no. 8, pp. 751–767, 2011.

[23] P. Chakraborty, G. G. Roy, S. Das, D. Jain, and A. Abraham, "An improved harmony search algorithm with differential mutation operator," *IFundamenta Informaticae*, vol. 95, no. 4, pp. 401–426, 2009.

[24] S. Zhao, P. N. Suganthan, and S. Das, "Dynamic multi-swarm particle swarm optimizer with sub-regional harmony search," in *Proc. IEEE Congr. Evol. Comput.*, Barcelona, Spain, 2010, pp. 1–8.

[25] O. Alia and R. Mandava, "The variants of the harmony search algorithm: An overview," *Artif. Intell. Rev.*, vol. 36, pp. 49–68, 2011.

[26] S. Das, A. Mukhopadhyay, A. Roy, A. Abraham, and B. K. Panigrahi, "Exploratory power of the harmony search algorithm: Analysis and improvements for global numerical optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 1, pp. 89–106, Feb. 2011.

[27] B. K. Panigrahi, V. R. Pandi, S. Das, and A. Abraham, "Population variance harmony search algorithm to solve optimal power flow with non-smooth cost function," in *Recent Advances In Harmony Search Algorithm* (Studies in Computational Intelligence vol. 270), Z. W. Geem, Ed. Berlin, Germany: Springer-Verlag, 2010, pp. 65–75.

[28] M. A. Al-Betar and A. T. Khader, "A harmony search algorithm for university course timetabling," *Ann. Operat. Res.*, pp. 1–29. DOI: 10.1016/j.amc.2011.11.095.

[29] K. Socha, J. Knowles, and M. Samples, "A max-min ant system for the university course timetabling problem," in *ANTS* (Lecture Notes in Computer Science, vol. 2463). Berlin, Germany: Springer-Verlag, 2002, pp. 1–13.

[30] M. A. Al-Betar, A. T. Khader, and I. Y. Liao, "A harmony search algorithm with multi-pitch adjusting rate for university course timetabling," in *Recent Advances In Harmony Search Algorithm* (Studies in Computational Intelligence vol. 270), Z. W. Geem, Ed. Berlin, Germany: Springer-Verlag, 2010, pp. 147–162.

[31] E. K. Burke and S. Petrovic, "Recent research directions in automated timetabling," *Eur. J. Operat. Res.*, vol. 140, no. 2, pp. 266–280, 2002.

[32] S. Abdullah, E. K. Burke, and B. McColum, "An investigation of variable neighbourhood search for university course timetabling," in *Proc. 2nd Multidisciplinary Int. Conf. Schedul.: Theory Appl.*, New York, G. Kendall, L. Lei, and M. Pinedo, Eds., Jul. 18–21, 2005, pp. 413–427.

[33] H. Asmuni, E. K. Burke, and J. M. Garibaldi, "Fuzzy multiple heuristic ordering for course timetabling," in *Proc. 5th United Kingdom Workshop Comput. Intell. (UKCI)*, 2005, pp. 302–309.

[34] P. McMullan, "An extended implementation of the great deluge algorithm for course timetabling," in *Proc. 7th Int. Conf. Computational Science, Part I*, 2007, pp. 538–545.

[35] J. Obit, D. Landa-Silva, D. Ouelhadj, and M. Sevaux, "Non-linear great deluge with learning mechanism for solving the course timetabling problem," in *Proc. 8th Metaheuris-tics Int. Conf.*, 2009, p. 10.

[36] H. Turabieh, S. Abdullah, and B. McCollum, "Electromagnetism-like mechanism with force decay rate great deluge for the course timetabling problem," in *Proc. Rough Sets Knowl. Technol.*, 2009, pp. 497–504.

[37] S. Abdullah, E. K. Burke, and B. McCollum, "Using a randomised iterative improvement algorithm with composite neighbourhood structures for the university course timetabling problem," in *Proc. Metaheuristic*, 2007, pp. 153–169.

[38] M. R. Malim, A. T. Khader, and A. Mustafa, "Artificial immune algorithms for university timetabling," in *Proc. 6th Int. Conf. Pract. Theory Automated Timetabling*, Brno, Czech Republic, Aug. 2006, pp. 234–245.
[39] R. Lewis and B. Paechter, "New crossover operators for timetabling with evolutionary algorithms," in *Proc. 5th Int. Conf. Recent Adv. Soft Comput.*, Nottingham, England, 2004, pp. 189–194.
[40] R. Lewis and B. Paechter, "Application of the grouping genetic algorithm to university course timetabling," in *Evolutionary Computation in Combinatorial Optimization (EvoCop)* (Lecture Notes in Computer Science vol. 3448), G. Raidl and J. Gottlieb, Eds. Berlin, Germany: Springer-Verlag, 2005, pp. 144–153.
[41] Y.-S. Ong, M. Lim, and X. Chen, "Memetic computation—Past, present & future[research frontier]," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 24–31, May 2010.
[42] R. Qu, E. K. Burke, B. McCollum, L. T. G. Merlot, and S. Y. Lee, "A survey of search methodologies and automated system development for examination timetabling," *J. Schedul.*, vol. 12, no. 1, pp. 55–89, 2009.
[43] M. Fesanghary, M. Mahdavi, M. Minary-Jolandan, and Y. Alizadeh, "Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems," *Comput. Methods Appl. Mech. Eng.*, vol. 197, no. 33–40, pp. 3080–3091, 2008.
[44] E. K. Burke and J. D. Landa Silva, "The design of memetic algorithms for scheduling and timetabling problems," in *Recent Advances in Memetic Algorithms and Related Search Technologies* (Studies in Fuzziness and Soft Computing vol. 166). New York: Springer-Verlag, 2004, pp. 289–312.
[45] E. K. Burke, J. P. Newall, and R. F. Weare, "A memetic algorithm for university exam timetabling," in *Proc. 1st Int. Conf. Practice and Theory of Automated Timetabling (PATAT1995)* (Lecture Notes in Computer Science vol. 1153), E. Burke and P. Ross, Eds. Berlin, Germany: Springer-Verlag, 1996, pp. 241–250.
[46] E. K. Burke and J. P. Newall, "A multistage evolutionary algorithm for the timetable problem," *IEEE Trans. Evol. Comput.*, vol. 3, no. 1, pp. 63–74, Apr. 1999.
[47] S. Abdullah, E. K. Burke, and B. McCollum, "A hybrid evolutionary approach to the university course timetabling problem," in *Proc. IEEE Congr. Evol. Comput.*, Singapore, Sep. 2007, pp. 1764–1768.
[48] O. Rossi-Doria and B. Paechter, "A memetic algorithm for university course timetabling," presented at the Combinatorial Optimisation (CO'2004), School of Computing, Napier Univ., Scotland, 2004.
[49] S. Abdullah and H. Turabieh, "Generating university course timetable using genetic algorithm and local search," in *Proc. 3rd Int. Conf. Converg. Hybrid Inf. Technol.*, 2008, pp. 254–260.
[50] S. Jat and S. Yang, "A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling," *J. Schedul.*, vol. 14, no. 6, pp. 617–637, 2010. DOI:10.1007/s10951-010-0202-0.
[51] S. Yang and S. N. Jat, "Genetic algorithms with guided and local search strategies for university course timetabling," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 41, no. 1, pp. 93–106, Jan. 2010.
[52] D. Landa-Silva and J. H. Obit, "Evolutionary non-linear great deluge for university course timetabling," in *Hybrid Artificial Intelligence Systems* Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence vol. 5572), E. Corchado, X. Wu, E. Oja, E. Hristozov, and T. Jedlovcnik, Eds. Berlin, Germany: Springer-Verlag, 2009, pp. 269–276.
[53] M. Ayob and G. Jaradat, "Hybrid ant colony systems for course timetabling problems," in *Proc. 2nd Conf. Data Mining Optimization*, Oct. 27–28, 2009, pp. 120–126.
[54] E. K. Burke, G. Kendall, and E. Soubeiga, "A tabu-search hyperheuristic for timetabling and rostering," *J. Heurist.*, vol. 9, no. 6, pp. 451–470, 2003.
[55] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu, "A graph-based hyper-heuristic for educational timetabling problems," *Eur. J. Oper. Res.*, vol. 176, no. 1, pp. 177–192, 2007.
[56] M. W. Carter and G. Laporte, "Recent developments in practical course timetabling," in *The Practice and Theory of Automated Timetabling* (Lecture Notes in Computer Science vol. 1408), B. E. K and C. M, Eds. Berlin, Germany: Springer-Verlag, 1997, pp. 3–19.
[57] Z. W. Geem, "Particle-swarm harmony search for water network design," *Eng. Optim.*, vol. 41, no. 4, pp. 297–311, 2009.
[58] O. Rossi-Doria, C. Blum, J. Knowles, M. Sampels, K. Socha, and B. Paechter, "A local search for the timetabling problem," in *Proc. 4th Int. Conf. Pract. Theory Automat. Timetabling*, Gent, Belgium, 2002, pp. 124–127.
[59] B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. Parkes, L. Di Gaspero, R. Qu, and E. K. Burke, "Setting the research agenda in automated timetabling: The second international timetabling competition," *Informs J. Comput.*, vol. 22, no. 1, pp. 120–130, Jan. 2010. DOI:10.1287/ijoc.1090.0320.
[60] M. Chiarandini, M. Birattari, K. Socha, and O. Rossi-Doria, "An effective hybrid algorithm for university course timetabling," *J. Schedul.*, vol. 9, no. 5, pp. 403–432, 2006.
[61] B. McCollum, "University timetabling: Bridging the gap between research and practice," in *Proc. 5th Int. Conf. Practice and Theory of Automated Timetabling*, 2006, pp. 15–35.
[62] M. Tuga, R. Berretta, and A. Mendes, "A hybrid simulated annealing with Kempe chain neighborhood for the university timetabling problem," in *Proc. 6th IEEE/ACIS Int. Conf. Comput. Inf. Sci.*, 2007, pp. 400–405.
[63] M. Al-Betar, A. Khader, and T. Gani, "A harmony search algorithm for university course timetabling," presented at the 7th Int. Conf. Practice Theory Automat. Timetabling, Montreal, Canada, Aug. 2008.
[64] D. Landa-Silva and J. H. Obit, "Great deluge with non-linear decay rate for solving course timetabling problems," in *Proc. 4th Int. IEEE Conf. Intell. Syst.*, Sep. 2008, pp. 8.11–8.18.
[65] K. S. Lee and Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice," *Comput. Methods Appl. Mech. Eng.*, vol. 194, no. 36–38, pp. 3902–3933, 2005.
[66] K. S. Lee and Z. W. Geem, "A new structural optimization method based on the harmony search algorithm," *Comput. Struct.*, vol. 82, no. 9–10, pp. 781–798, 2004.
[67] T. Arani and J. A. Lofti, "A three phased approach to final exam scheduling," *IIE Trans.*, vol. 21, no. 4, pp. 86–96, 1989.
[68] M. W. Carter, G. Laporte, and S. Y. Lee, "Examination timetabling: Algorithmic strategies and applications," *J. Operat. Res. Soc.*, vol. 74, pp. 373–383, 1996.
[69] A. Gaw, P. Rattadilok, and R. S. Kwan, "Distributed choice function hyperheuristics for timetabling and scheduling," in *Practice and Theory of Automated Timetabling V* (Lecture Notes in Computer Science vol. 3616), E. K. Burke and M. Trick, Eds. Berlin, Germany: Springer-Verlag, 2005, pp. 51–67.
[70] S. Abdullah, K. Shaker, B. McCollum, and P. McMullan, "Construction of course timetables based on great deluge and tabu search," presented at the Metaheuristics Int. Conf., VIII Meteheuristic, pp. 13–16, Hamburg, Jul. 2009.
[71] A. Abuhamdah and M. Ayob, "Experimental result of particle collision algorithm for solving course timetabling problems," *Int. J. Comput. Sci. Network Security*, vol. 9, no. 9, pp. 134–142, 2009.
[72] A. Abuhamdah, "Experimental result of late acceptance randomized descent algorithm for solving course timetabling problems," *Int. J. Comput. Sci. Netw. Security*, vol. 10, no. 1, pp. 192–200, 2010.
[73] W. S. Jang, H. I. Kang, and B. H. Lee, "Hybrid simplex-harmony search method for optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, Hong Kong, Jun. 1–6, 2008, pp. 4157–4164.
[74] S. N. Jat and S. Yang, "A guided search genetic algorithm for the university course timetabling problem," in *Proc. 4th Multidiscipl. Conf. Schedul.: Theory Appl.*, Dublin, Ireland, Aug. 10–12, 2009, pp. 180–191.
[75] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.
[76] Z. W. Geem and K.-B. Sim, "Parameter-setting-free harmony search algorithm," *Appl. Math. Comput.*, vol. 217, no. 8, pp. 3881–3889, 2010.
[77] O. Hasançebi, F. Erdal, and M. P. Saka, "Adaptive harmony search method for structural optimization," *ASCE J. Struct. Eng.*, vol. 136, no. 4, pp. 419–431, 2010.

Mohammed Azmi Al-Betar received the B.Sc. and M.Sc. degrees from the Department of Computer Science, Yarmouk University, Irbid, Jordan, in 2001 and 2003, respectively, and the Ph.D. degree from the School of Computer Sciences, University Sains Malaysia (USM), Pulau Pinang, Malaysia, in October 2010.

He is currently an Assistant Professor with the Department of Computer Science, Al-Zaytoonah University of Jordan, Amman, Jordan, and a Postdoctoral Research Fellow with the School of Computer Sciences, USM. He has authored a number of publications in international journals, conferences, and book chapters. His research interests are mainly directed to metaheuristic optimization methods and hard combinatorial optimization problems including scheduling and timetabling.

Ahamad Tajudin Khader (M'09) received the B.Sc. and M.Sc. degrees in mathematics from Ohio University, Athens, in 1982 and 1983, respectively, and the Ph.D. degree in computer science from the University of Strathclyde, Glasgow, U.K., in 1993.

He is currently a Professor and Deputy Dean with the School of Computer Sciences, Universiti Sains Malaysia, Pulau Pinang, Malaysia. He has authored many publications in international journals, conferences, and book chapters. His research interest mainly focuses on optimization, scheduling, and timetabling. In particular, he is interested in evolutionary and metaheuristic algorithms. He is attracted to search in general.

Dr. Khader is a member of the Association of Computing Machinery, Special Interest Group for Genetic and Evolutionary Computation, and Computational Intelligence Society.

Munir Zaman received the Graduate degree in computer science from the University of Manchester, Manchester, U.K., in 1988, the M.Sc. degree in computer integrated manufacturing from Cranfield University, Bedford, U.K., in 1990, and the Doctorate degree in mobile robotic localization from the University of Surrey, Surrey, U.K., in 2006.

He was a CAD Analyst Programmer with British Aerospace (Military Aircraft Division), which he followed with an M.Sc. degree. He then became a Scientist with the UK Ministry of Defence where he worked in operations research, C3I, and unmanned ground vehicles. During this time, he authored over a dozen technical reports. He then returned to academia and worked toward the Doctorate degree. He is currently a Senior Lecturer with the School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia. His research interests include computer vision, sensor synchronization, and operations research.

Dr. Zaman is a Chartered Member of the British Computer Society (CEng, MBCS).