A hybrid harmony search for university course timetabling

Mohammed Azmi Al-Betar • Ahamad Tajudin Khader

Abstract. Combinations of metaheuristics to solve highly constrained combinatorial optimisation problems have recently become a promising research area. Some combinations emerge from integrating evolutionary algorithms and local based algorithms to strike the right balance between global exploration of the search region and local exploitation of promising regions. This study presents hybridization between harmony search algorithms and hill climbing optimisers to tackle the university course timetabling problem. The solution is essentially a harmony search algorithm. The hill climbing optimiser is responsible for improving the new harmony vector, obtained from harmony search, with a certain probability. Furthermore, inspired by the practical swarm optimisation, we modified the memory consideration operator to let it improvise rather than always working in the same way. Our algorithm converges significantly faster than does the classical harmony search algorithm. We evaluate our hybrid algorithm with a standard benchmark introduced for UCTP. Our results show that our algorithm can find higher quality solutions than can previous work.

1 Introduction

Timetabling problems are complex combinatorial optimisation problems classified as NP-hard [1]. The general requirement for timetabling problems “is to assign a set of entities (such as tasks, public events, vehicles, or people) to a limited number of resources over time, in such a way as to meet a set of pre-defined schedule requirements”[2]. Such problems appear in many forms such as school, examination, or course timetabling [3, 4, 2, 5]; employee timetabling [6]; transport timetabling [7]; and nurse rostering [8].

This paper addresses the university course timetabling problem (UCTP), which includes assigning sets of events to particular rooms and time slots on a weekly basis according to constraints. There are two kinds of timetabling constraints: hard and soft [3]. A timetabling
solution must satisfy hard constraints in order to be feasible, but need not necessarily satisfy all soft constraints. In other words, soft constraints may be violated, and the quality of a timetable solution is optimised by satisfying soft constraints. The most common soft constraints reflect the preferences of students and teachers.

The UCTP has attracted attention from researchers in artificial intelligence and operational research for quite a long time. Many approaches have been proposed to tackle this problem. Because timetabling is similar to graph colouring problems (graph vertices correspond to timetable events, and colours correspond to time periods) [9, 10], the earliest approaches depended on graph colouring heuristics (that is, largest degree, largest enrolment, largest weighted degree, or largest saturating degree); these heuristics build a timetable by assigning events to valid time periods and rooms one by one according to a particular order. Modern approaches often use these heuristics to find a feasible timetable [11, 12], and also use them as low-level heuristics in hyper heuristics [13, 14]. Furthermore, they serve as a fuzzy inference rule in fuzzy multiple ordering [12].

Researchers have successfully applied metaheuristics in both local search and population-based approaches to the UCTP. Local search approaches (iterated local search [15], tabu search [16], simulated annealing [17], and others) make sequences of local changes to the initial (or current) solution guided by an objective function until they reach a locally optimal solution. The key to the success of local approaches is the definition of neighbourhood structures, which make more of the search space reachable. Population-based approaches, also called evolutionary algorithms, have also been successfully applied to the UCTP. Examples of such approaches include genetic algorithms [18], ant colony optimization [15], and harmony search [19]. Population-based approaches typically use recombination and randomisation rules, which mix different components of solutions (or individuals) with the hope of finding a ‘good’ mixture in the new solution(s). The following articles survey previous metaheuristics for the UCTP [3, 4, 2, 20, 5].

Hybridization of metaheuristics is an efficient solution for the UCTP. In a recent comprehensive survey for timetabling problems, Qu et al. [21] suggest that “There are many research directions generated by considering the hybridization of meta-heuristic methods particularly between population-based methods and other approaches”. Hybridization between genetic algorithms and local algorithms or the so-called memetic algorithm is a successful hybrid approach that tackled university timetabling problems [22]. Recently, Abdullah et al. [11] have tackled the UCTP using the hybrid evolutionary algorithm. The system of Abdullah et al. combines variable neighbourhood algorithms and genetic algorithms, and it obtained the best known result for UCTP problem instances at the time of publishing.

The harmony search algorithm (HSA), developed by Geem et al. [23] is a new metaheuristic population-based algorithm that imitates musical performance. HSA combines the key components of population-based and local based algorithms in a simple optimisation model [23, 24].

The HSA has strong components (operators) that can efficiently and effectively explore the search space. It combines some components of population-based algorithms (that is, memory consideration corresponding to recombination and random consideration corresponding to mutation), which are responsible for global improvement, and some components of local search algorithms (that is, pitch adjustment corresponding to neighbourhood structures), which are responsible for local improvement.

Our previous work adapted the HSA to the UCTP [19]. The pitch adjustment operator is designed to work similarly to neighbourhood structures. At each run, this operator makes a
limited number of local changes to the new improvised solution, called “new harmony”, based on the pitch adjustment rate (PAR) and harmony memory consideration rate (HMCR). These local changes move the new harmony solution to the neighbouring solution as a ‘random walk’ acceptance criterion without considering the objective function value. Therefore, the HSA cannot be fine-tune the search space region of new harmony to find a local optimal solution at this region. In this study, we introduce a Hybrid Harmony Search Algorithm (HHSA) to be applied to the UCTP. The hill climbing optimizer (HCO) is incorporated as a new operator of the HSA to explore the search space region of new harmony thoroughly to fine-tune this region. In addition, we propose an idea stemming from practical swarm algorithms, modified by a memory consideration operator, to lower the selection pressure of this operator. Our method dramatically improves the convergence of HSA. In the HHSA, the essential application of the HSA, the HCO serves as a new operator, controlled by a parameter we call ‘Hill Climbing Rate (HCR)’. Furthermore, inspired by practical swarm optimisation, we modified the functionality of the memory consideration operator to increase its improvisational ability.

This paper is organized as follows: Section 2 discusses the university course timetabling problem, and Section 3 explains the HSA. Section 4 presents the HSA for the UCTP and our hybridization strategy. Section 5 discusses our experimental results and compares them to the previous literature. In the final section, we present conclusions and future directions for our algorithm.

2 University course timetabling problem

2.1 Problem Description

Napier University course timetabling in Edinburgh was considered as an application problem by a Metaheuristics Network1 (MN). The MN described the context of this problem as assigning certain events to suitable rooms and timeslots according to hard and soft constraints; a solution must satisfy hard constraints, while it should minimise violations of soft constraints.

The UCTP benchmark we consider was constructed by Socha et al. [15] who proposed eleven data instances using Paechter's generator2 to mimic the same Napier University course timetabling. These data instances are grouped into three classes: five small, five medium, and one large, and they include three hard constraints:

H1. Students must not be double-booked for events.
H2. Room sizes and features must be suitable for assigned events.
H3. Rooms must not be double booked for events.

And three soft constraints:

S1. A student shall not have a class in the last slot of the day.
S2. A student shall not have more than two classes in a row.
S3. A student shall not have a single class on one day.

The main objective of this context is to satisfy the hard constraints and to minimise the violation of soft constraints.

---

2 Ben Paechter is a Professor in the School of Computing at Napier University, UK and a member of Metaheuristics Network. His official home page is “http://www.dcs.napier.ac.uk/~benp/”. (04 Feb 2008).
2.2 Problem formulation

In order to illustrate the bridge between the UCTP and harmony search algorithm, it is useful to discuss UCTP mathematically to overview the set of parameters and decision variables. Indeed, the formulation, presented here, could not be solved by exact approaches since the number of decision variables would be large. In the following subsections, the sets, parameters and decision variables are expressed before the problem formulation is introduced.

2.2.1 Sets and Parameters descriptions

- A set \( E = \{1, \ldots, \bar{E}\} \) of events, each of which contains certain students and needs certain features.
- A set \( R = \{1, \ldots, \bar{R}\} \) of rooms, each of which has a seat capacity and its own features.
- A set \( S = \{1, \ldots, \bar{S}\} \) of students, each of whom enrols in some events.
- A set \( F = \{1, \ldots, \bar{F}\} \) of features, such as overhead projectors or special whiteboards.
- A set \( P = \{1, \ldots, \bar{P}\} \) of timeslots where \( p = 45 \) (5 days with 9 periods on each day).
- A set \( D = \{1, \ldots, \bar{D}\} \) of days, where each day has nine periods.
- Ordered subsets \( P_d \) of \( P \) corresponding to a period in a day \( d \). where \( p = 45 \) (5 days with 9 periods on each day).
- An ordered subset \( L_d \) that contains the last periods of each day.
- \( \bar{E}, \bar{F}, \bar{S}, \bar{R}, \bar{P} \) are the number of events, rooms, students, features, and timeslots respectively.

- \( s_r^e \) is the size of room \( r \), \( r \in R \).
- \( s_e^s \) is the number of students enrolled in event \( e \), \( e \in E \).
- \( w_{r,f} \) is 1 if event \( e \) requires feature \( f \), 0 otherwise. \( e \in E \) and \( f \in F \).
- \( y_{r,f} \) is 1 if room \( r \) contains feature \( f \), 0 otherwise. \( r \in R \) and \( f \in F \).
- \( t_s^e \) is 1 if student \( s \) is enrolled in event \( e \), 0 otherwise. \( s \in S \) and \( e \in E \).

2.2.2 Decision variables

- \( x \) are binary decision variables indexed by events, rooms and timeslots. Their value \( x_{e,r,p} \) is 1 if and only if event \( e \) occurred in room \( r \) and time period \( p \), \( e \in E, r \in R \) and \( p \in P \).
• $C_{i}^{ld}$ (Last period of day) are decision variables indexed by students; their value indicating the number of violations of soft constraint S1 by student $s \in S$.

• $C_{i}^{2l}$ (More than two events in a row) are decision variables indexed by students; their value indicates the number of violations of soft constraint S2 by student $s \in S$.

• $C_{i}^{sd}$ (Single class in a day) are decision variables indexed by students; their value indicate to the number of violations of soft constraint S3 by student $s \in S$.

• $z_{s,d}$ are binary decision variables indexed by student and day; their value indicates that the student $s$ has a single class in a day $d$. $s \in S$ and $d \in D$.

2.2.3 Formulation

The objective function of the obtained solution can be described as follows:

Minimize \( \sum_{s \in S} \left( C_{i}^{ld} + C_{i}^{2l} + C_{i}^{sd} \right) \) \hspace{1cm} (1)

The objective function is described in Eq. (1), $C_{i}^{ld}$, $C_{i}^{2l}$, and $C_{i}^{sd}$ consecutively describe the violations of the soft constraints S1, S2, S3 made against the will of each student $s$. When each violation occurs in the solution, it will be penalized by 1.

The hard constraints are described by Eqs. (2-6); these constraints are mandatory for any feasible solution.

\[
\forall e \in E \quad \sum_{r \in R} \sum_{p \in P} x_{r,p} = 1 \hspace{1cm} (2)
\]

\[
\forall p \in P, \forall s \in S \quad \sum_{e \in E} \sum_{r \in R} t_{e,r} x_{e,r,p} \leq 1 \hspace{1cm} (3)
\]

\[
\forall r \in R, \forall p \in P \quad \sum_{e \in E} s_{r} x_{e,r,p} \leq s_{r}^{d} \hspace{1cm} (4)
\]

\[
\forall f \in F, \forall r \in R, \forall p \in P \quad \sum_{e \in E} w_{f,r} y_{f,r} x_{e,r,p} = 1 \hspace{1cm} (5)
\]

\[
\forall r \in R, \forall p \in P \quad \sum_{e \in E} x_{e,r,p} \leq 1 \hspace{1cm} (6)
\]

Eq. (2) describes the implicit constraint, not mentioned in Section 2.1, which meant that timetable solution must be complete and each event must be presented once. Eq. (3) describes H1, which means that no student may be double-booked for events. Eqs. (4, 5) describe H2, which reflects room allocation with suitable capacity and features for events. Eq. (6) reflects H3, which explains that no room may be double-booked for events.

\[
\forall s \in S \quad C_{i}^{ld} = \sum_{e \in E} \sum_{r \in R} \sum_{q \in Q} t_{e,r} x_{e,r,q} \hspace{1cm} (7)
\]

\[
\forall s \in S \quad C_{i}^{2l} = \sum_{e \in E} \sum_{r \in R} \sum_{p \in P} \sum_{q \in Q} \sum_{m \in M} t_{e,r} x_{e,r,p} x_{e,r,q} x_{e,r,m} \hspace{1cm} (8)
\]

\[
\forall s \in S, \forall d \in D, \forall z = \sum_{s \in S} \sum_{d \in D} z_{s,d} = \begin{cases} 1 & \sum_{s \in S} \sum_{r \in R} \sum_{p \in P} t_{e,r} x_{e,r,p} = 1 \\ 0 & \text{otherwise} \end{cases} \hspace{1cm} (9)
\]

\[
\forall s \in S \quad C_{i}^{sd} = \sum_{e \in D} z_{r,s} \hspace{1cm} (10)
\]
The decision variables $C_{sC}^{l_{dp}}$, $C_{sC}^{l_{2}}$, $C_{sC}^{l_{1}}$ reflect the violation of soft constraints S1, S2, and S3 consecutively. These soft constraints are formalised in Eq. (7), which describes the soft constraint S1, Eq. (8) describes the soft constraint S2, and Eq. (10) describes the soft constraint S3. Eq. (9) is necessary for describing S3, which penalises students who have only attended a single event in a day by 1, while Eq. (10) calculates all violations of any students for all days.

3 Harmony search algorithm

Current metaheuristics originate from natural phenomena and mimic real situations. Examples include physical annealing in simulated annealing, ant behaviour in ant systems, vertebrate immune systems in artificial immune systems, human neural system in artificial neural networks, human memory in tabu search, and evolution in genetic algorithms. The HSA is a new metaheuristic algorithm developed by Geem et al. [23] to imitate the natural phenomenon of musical performance, in which musicians jointly tune the pitches of their instruments to find a euphonious harmony. This intense and prolonged musical process leads them to the perfect (or Nirvana) state.

The HSA is a scalable stochastic search mechanism, simple in its concept, and has few parameters, needing no derivational information at the initial stage [24]. It can find a trade-off between global improvement of search space and local improvement of the new harmony. Furthermore, HSA works at the component level rather than the individual level, which considers components (or decision variables) of the stored solutions (or harmony memory vectors) as essential elements to generate a new solution called the new harmony. The HSA has been successfully applied to several optimisation problems [25].

Algorithm 1 describes the basic HSA, which has five steps, as follows [23]:

- **Step 1.** Initialize the HSA and optimisation problem parameters.
- **Step 2.** Initialize the Harmony Memory (HM).
- **Step 3.** Improvise a new harmony.
- **Step 4.** Update the harmony memory.
- **Step 5.** Check the stopping criterion.

These steps will be explained similarly to Lee and Geem [24] in the next five subsections:

3.1 Initialize the HSA and optimisation problem parameters

In step 1, the optimisation problem can be specified as follows:

$$\min \{ f(x) | x \in X \} \quad \text{Subject to} \quad g(x) \leq 0 \quad \text{and} \quad h(x) = 0$$

(11)

Where $f(x)$ is the objective function; $x$ is the set of each decision variable $x_i$ that is, $i \in \{ 1 \ldots N \}$; $X$ is the possible range of values for each decision variables, that is, $X_i = \{ x_i(1), x_i(2), \ldots , x_i(K) \}$. $N$ is the number of decision variables, and $K$ is the number of possible values of decision variables. $g(x)$ is inequality constraint function and $h(x)$ is equality constraint function.

Furthermore, the parameters of the HSA required to solve the optimisation problem are also specified in this step: Harmony Memory Consideration Rate (HMCR), Harmony Memory Size (HMS) (that is, equivalent to population size), Pitch Adjustment Rate (PAR), and Number of Improvisations (NI) (that is, the maximum number of generations). The HSA parameters will be explained in more detail in the next steps.
### Algorithm 1 The framework of a basic HSA

1. **Input**: the data instance $P$ of the optimisation problem and the parameters for HSA (HMCR, PAR, NI, HMS).
2. Initialize-HM $\{x_1^{HMS}, x_2^{HMS}, \ldots, x_N^{HMS}\}$
3. Recognize $x_{x_{\text{worst}}} \in \{x_1^{HMS}, x_2^{HMS}, \ldots, x_N^{HMS}\}$.
4. **while** not termination criterion specified by NI **do**
5. $x_{\text{NEW}} \leftarrow \emptyset$ // new harmony vector
6. **for** $j = 1, \ldots, N$ do // $N$ is the number of decision variables.
7. **if** $U(0,1) \leq \text{HMCR}$ then
8. begin
9. $x_j^{\text{NEW}} \in \{x_j^{HMS}, \ldots, x_j^{HMS}\}$ // memory consideration
10. **if** $U(0,1) \leq \text{PAR}$ then // pitch adjustment
11. $x_j^{\text{NEW}} \in N(x_j^{\text{NEW}})$ // $N(x_j^{\text{NEW}})$ is the neighbouring values of variable $x_j^{\text{NEW}}$
12. end
13. else // random consideration
14. $x_j^{\text{NEW}} \in X_j$
15. end if
16. **end for**
17. **if** $f(x_{\text{NEW}}) \leq f(x_{\text{worst}})$ then
18. $x_{\text{worst}} = x_{\text{NEW}}$
19. **end if**
20. **end while**
21. **output**: the best solution obtained so far.

#### 3.2 Initialize the harmony memory

The harmony memory (HM) is a vector of solutions with size HMS as shown in Eq. (12). In this step, those solutions are randomly constructed and decreasingly filled to HM based on the values of the objective function.

$$
\text{HM} = \begin{bmatrix}
x_1^1 & x_2^1 & \cdots & x_N^1 & f(x_1^1) \\
x_1^2 & x_2^2 & \cdots & x_N^2 & f(x_2^2) \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
x_1^{HMS} & x_2^{HMS} & \cdots & x_N^{HMS} & f(x_N^{HMS})
\end{bmatrix}
$$  \hspace{1cm} (12)

#### 3.3 Improvise a new harmony

In this step, the HSA generates (improvises) a new harmony vector, $x_{\text{NEW}} = (x_1^{\text{NEW}}, x_2^{\text{NEW}}, \ldots, x_N^{\text{NEW}})$, based on three operators: (1) memory consideration, (2) random consideration, and (3) pitch adjustment.

In the memory consideration, the decision variables of the new harmony randomly inherited the historical value stored in the HM with probability HMCR. Restating, the value of decision variable $(x_1^{\text{NEW}})$ is chosen from $\{x_1^1, x_2^1, x_3^1, \ldots, x_N^{HMS}\}$ that stored in HM, the next decision variable $(x_2^{\text{NEW}})$ is chosen from $\{x_1^1, x_2^1, x_3^1, \ldots, x_N^{HMS}\}$, the other decision variables, $(x_3^{\text{NEW}}, x_4^{\text{NEW}}, x_5^{\text{NEW}}, \ldots)$, are chosen consecutively in the same way with probability HMCR where
(0 ≤ HMCR ≤ 1). Other values that are not chosen depending on memory considerations with probability (1-HMCR) are selected according to their possible range by random consideration as in Eq. (13).

\[
x_i^{\text{NEW}} = \begin{cases} 
  x_i^{\text{NEW}} & \text{with probability } HMCR \\
  x_i & \text{with probability } 1 - HMCR 
\end{cases}
\] (13)

The HMCR parameter is the probability of selecting one value of the decision variable, \(x_i^{\text{NEW}}\), based on historical values stored in the HM. For instance, if (HMCR = 0.90), that indicates that the probability of selecting the value of decision variable from historic value in the HM with the probability is 90%, and the value of decision variable is selected from its possible range with a probability of 10%.

All decision variables, \(x_i^{\text{NEW}} = \{x_1^{\text{NEW}}, x_2^{\text{NEW}}, x_3^{\text{NEW}}, \ldots, x_N^{\text{NEW}}\}\), chosen by memory considerations are examined to be pitch-adjusted with the probability \(PAR\). Pitch adjustment makes a sequence of local changes (pitch adjustments) on the new harmony with probability \(PAR\) where \(0 \leq PAR \leq 1\). The size of these local changes is selected as in Eq. (14).

The pitch adjusting decision for \(x_i^{\text{NEW}}\) is:

\[
x_i^{\text{NEW}} = \begin{cases} 
  \text{Yes} & \text{with probability } PAR \\
  \text{No} & \text{with probability } 1 - PAR 
\end{cases}
\] (14)

The values of decision variables not obtained by memory consideration with probability of \((1 - PAR)\) are not changed. The decision variable, suppose it is \(x_i^{\text{NEW}}(k)\), that is, the \(k\)th element (or value) in \(x_i^{\text{NEW}}\), that examined to be examined for pitch adjusting, is chosen as a neighbouring value with probability \((PAR \times HMCR)\). For example, if HMCR=90% and PAR=20%, the probability of selecting the neighbouring value of any decision variable is 18%.

If the pitch adjusting decision for \(x_i^{\text{NEW}}\) is Yes, the pitch-adjusted value of \(x_i^{\text{NEW}}(k)\) is:

\[
x_i^{\text{NEW}} = x_i^{\text{NEW}}(k + m) \quad \text{where } m \text{ is the neighbouring index, } m \in \{-2, -1, 0, 1, 2, \ldots\}
\] (15)

The HMCR parameters help the HSA to find globally improved solutions, and the PAR parameters help it to find locally improved solutions [26].

3.4 Update the harmony memory

If the new harmony vector, \(x^{\text{NEW}} = \{x_1^{\text{NEW}}, x_2^{\text{NEW}}, x_3^{\text{NEW}}, \ldots, x_N^{\text{NEW}}\}\), has a better objective function value than the worst harmony stored in the HM, then it is included in the HM and the worst harmony vector is excluded from the HM.

3.5 Check the stopping criterion

Steps 3 and 4, presented in subsections 3.3 and 3.4, are repeated until the stop criterion (maximum number of improvisations) is met. This is determined by the NI parameter.
4 A hybrid harmony search algorithm for the UCTP

This section describes our HHSA for UCTP. First, we briefly describe HSA as presented in our previous work on UCTP; then, we illustrate the combination of HSA, HCO, and PSO. Figure 1 gives pseudo-code for each step of HHSA.

4.1 The HSA for UCTP

The HSA represents the UCTP as a decision variable $x$ indexed by rooms and time slots, where $x_{r,p}$ says whether a room contains event $e$ in room $r$ and timeslot $p$ or -1 if it is empty. $r \in R$ and $p \in P$; See Eq. (16). The value of parameters $\overline{\sigma}$ indicates the number of rooms, and $\overline{p}$ is the number of time slots.

$$x = \begin{bmatrix}
x_{0,0} & x_{0,1} & \cdots & x_{0,\overline{p}} \\
x_{1,0} & x_{1,1} & \cdots & x_{1,\overline{p}} \\
x_{2,0} & x_{2,1} & \cdots & x_{2,\overline{p}} \\
\vdots & \vdots & \ddots & \vdots \\
x_{r,0} & x_{r,1} & \cdots & x_{r,\overline{p}}
\end{bmatrix}$$  \hspace{1cm} (16)

Step 1 specifies the UCTP parameters and possible range for each decision variable (that is, event). The objective function, $f(x)$, which describes the violations of soft constraints $S_1$, $S_2$, and $S_3$, is defined in this step. The parameters of HSA (HMCR, PAR, HMS, and NI) are also selected in the same step.

In step 2, the random feasible timetables, $\{x^1, x^2, \ldots, x^{\text{HMS}}\}$, are generated according to the HMS. These timetables will be decreasingly stored to the HM according to the objective function. See Eq. (17). The feasibility for all HM members is maintained with a method that combines largest weighted degree (LWD), backtracking algorithms, and the MultiSwap algorithm (for more details, see the pseudo-code in step 2 of Figure 1).

$$HM = \begin{bmatrix}
x_{0,0}^1 & x_{0,1}^1 & \cdots & x_{0,\overline{p}}^1 \\
x_{1,0}^1 & x_{1,1}^1 & \cdots & x_{1,\overline{p}}^1 \\
x_{2,0}^1 & x_{2,1}^1 & \cdots & x_{2,\overline{p}}^1 \\
\vdots & \vdots & \ddots & \vdots \\
x_{r,0}^1 & x_{r,1}^1 & \cdots & x_{r,\overline{p}}^1
\end{bmatrix} \sim \begin{bmatrix}
x_{0,0}^{\text{HMS}} & x_{0,1}^{\text{HMS}} & \cdots & x_{0,\overline{p}}^{\text{HMS}} \\
x_{1,0}^{\text{HMS}} & x_{1,1}^{\text{HMS}} & \cdots & x_{1,\overline{p}}^{\text{HMS}} \\
x_{2,0}^{\text{HMS}} & x_{2,1}^{\text{HMS}} & \cdots & x_{2,\overline{p}}^{\text{HMS}} \\
\vdots & \vdots & \ddots & \vdots \\
x_{r,0}^{\text{HMS}} & x_{r,1}^{\text{HMS}} & \cdots & x_{r,\overline{p}}^{\text{HMS}}
\end{bmatrix}$$  \hspace{1cm} (17)

In step 3, the new harmony (see Eq. (18)) is improvised based on memory consideration, random consideration, and pitch adjustment operators; if the complete new harmony is not obtained, then the repair strategy has to take over.

$$x^{\text{NEW}} = \begin{bmatrix}
x_{0,0}^{\text{NEW}} & x_{0,1}^{\text{NEW}} & \cdots & x_{0,\overline{p}}^{\text{NEW}} \\
x_{1,0}^{\text{NEW}} & x_{1,1}^{\text{NEW}} & \cdots & x_{1,\overline{p}}^{\text{NEW}} \\
x_{2,0}^{\text{NEW}} & x_{2,1}^{\text{NEW}} & \cdots & x_{2,\overline{p}}^{\text{NEW}} \\
\vdots & \vdots & \ddots & \vdots \\
x_{r,0}^{\text{NEW}} & x_{r,1}^{\text{NEW}} & \cdots & x_{r,\overline{p}}^{\text{NEW}}
\end{bmatrix}$$  \hspace{1cm} (18)

In memory consideration, the value of the decision variable $x_{0,0}^{\text{NEW}}$ should be randomly selected from $x_{0,0}^{\text{NEW}} \in \{x_{0,0}^1, x_{0,0}^2, \ldots, x_{0,0}^{\text{HMS}}\}$, the values of other decision variables,
\( \left( x_{0,1}^{\text{NEW}}, x_{0,2}^{\text{NEW}}, x_{0,3}^{\text{NEW}}, \ldots, x_{0,0}^{\text{NEW}}, \ldots, x_{0,0}^{\text{NEW}}, \ldots, x_{0,0}^{\text{NEW}} \right) \), according to the basic HSA, should be selected sequentially in the same way [23, 24, 27].

In the UCTP case, this sequential process is not useful because, in our case, the feasibility for new harmony should be maintained (that is, we worked in feasible search space regions). The main difficulty of the UCTP lies in the attempt to find valid locations for each event. Indeed, each event has a specific number of valid locations in the HM members to be scheduled in the new harmony. For this reason, the selection of the new event, \( x_{i,j}^{\text{NEW}} \), to be scheduled in new harmony should be based on available locations for this event in the HM members.

For UCTP, we propose the smallest position algorithm to select the appreciate event, \( x_{i,j}^{\text{NEW}} \), to be scheduled in a new harmony with probability HMCR. In this algorithm, the event \( e \in E \) is ordered iteratively based on the minimum valid locations available in the HM members. Let set \( Q = \{ Q_1, Q_2, \ldots, Q_T \} \) be the HM members that have a valid location for event \( e \) to be scheduled in new harmony. The event \( e \) with the minimal number of valid places will be scheduled first. The valid location for event \( e \) will be randomly chosen from \( Q \) such that \( x_e^{\text{NEW}} \in \{ x_e^{\text{NEW}}, x_e^{\text{NEW}}, \ldots, x_e^{\text{NEW}} \} \) with probability HMCR, where \( \{ x_e^{\text{NEW}}, x_e^{\text{NEW}}, \ldots, x_e^{\text{NEW}} \} \) and \( K \) is the total number of HM members that have a valid location for event \( e \).

In random consideration, the event \( e \) selected according to the smallest position algorithm is scheduled to the new harmony based on the possible available locations in the new harmony rather than in HM members with probability (1-HMCR).

Indeed, in certain situations the two above operators cannot find a feasible timetable (this happens in some medium and large data instances). In this case, the algorithm initiates a repair process through one-level backtracking.

In the pitch adjustment operator, every decision variable (event), \( x_{i,j}^{\text{NEW}} \), obtained by the memory consideration is examined to determine whether it should be pitch-adjusted with probability PAR. Moreover, for the UTCP, we design two pitch adjusting procedures, each one of which reflects a neighbourhood structure. Each pitch adjusting procedure is controlled by a particular PAR value as follows:

1- PAR1- PAR1 denotes the rate of local changes in the new harmony by moving an examined event \( e \) to a different location while preserving feasibility. More formally, let an event \( e \) be assigned to feasible timetable \( x_e^{\text{NEW}} \) in location \( k, x_e^{\text{NEW}} = k \), based on memory consideration. Let location \( k' \) be randomly selected. Location \( k' \) is free and suitable for event \( e \). The event \( e \) is moved to another location \( k' \) such that \( x_e^{\text{NEW}} = k' \).

2- PAR2- It denotes the rate of local changes in the new harmony by swapping the location of the examined event \( e \) and another event \( e' \) while preserving feasibility. More formally, let an examined event \( e \) be assigned to feasible timetable \( x_e^{\text{NEW}} \) in the location \( k, x_e^{\text{NEW}} = k \), based on memory consideration. Let an event \( e' \) assigned to location \( k', x_e^{\text{NEW}} = k' \), be randomly selected. Note that both locations \( k \) and \( k' \) are suitable for both events \( e \) and \( e' \). The locations of event \( e \) and event \( e' \) are swapped such that \( x_e^{\text{NEW}} = k' \) and \( x_{e'}^{\text{NEW}} = k \).

These local changes of new harmony do a random walk without adopting any acceptance criteria like first improvement or best improvement. As such, this operator emphasises exploration rather than exploitation, which should be set to a small value.
Step 4 calculates the objective function for new harmony; if the objective function value of new harmony is better than the objective function of worst harmony in HM, then the new harmony is included in HM and the worst solution is excluded from the HM.

In step 5, steps 3 and 4 are repeated until the stop criterion determined by NI is satisfied.

4.2 Hybridization

This section discusses the hybridisation strategy between HSA and hill climbing optimiser (HCO), called the hybrid harmony search algorithm (HHSAs). We then explain the need to hybridise the HHSAs with Practical Swarm Optimisation (PSO) concepts to increase the convergence rate. Section 4.2.1 discusses the HHSAs and section 4.2.2 discusses hybridisation between HHSAs and PSO.

4.2.1 Hybridization with HCO

The HSA has three key components that generate a new harmony in every run. Memory and random considerations are the two components responsible for global improvement: memory consideration is a source of exploitation, and random considerations are a source of exploration [28]. The pitch adjustment component is the third component of HSA, which is responsible for local improvement. The size of the local improvements this component makes is determined mainly by the number of decision variables of each problem. The size is specified by the probability of HMCR and PAR: each decision variable that is selected according to memory consideration with probability HMCR is examined for potential pitch adjustment with probability PAR. Thus, the probability that each variable will be pitch-adjusted is HMCR × PAR. As such, the amount of local improvement in the new harmony does not guarantee that the new harmony converges to the local optimal solution in each run. Furthermore, these local improvements are done randomly, which means that not all local changes might be responsible for the improvement in the new harmony.

Recently, Fesanghary et al. [29] have proposed a new hybridisation strategy for the harmony search algorithm to solve engineering optimization problems. The main motivations of this hybrid algorithm are to improve the solution quality and lower the computational cost of HSA. Their work incorporates sequential quadratic programming into the step 4 of the HSA to improve the quality of the new harmony vector and other vectors in HM. This incorporation process, as Fesanghary et al. explain, needs careful attention when the HMS is chosen for the problem, especially considering that the local optimiser is computationally expensive. Finally, Fesanghary et al. suggest that the applicability of hybrid harmony search can be found by using a few harmony memory vectors.

In fact, the HHSAs presented in this study is similar to the memetic algorithm (MA), which is used mostly for timetabling problems by incorporating hill-climbing into genetic algorithm (GA) operators [22, 30]. One possible way to incorporate hill-climbing is ‘Elaborate Encodings and Operators’ [22], in which GA and the local search operator work in a feasible search space region. The main restriction to this strategy is that the exploratory ability of MA may be decreased.

In this study, we combine the hill climbing optimiser (HCO) with the HSA as a new operator (see Figure 1, step 3) which is controlled by a new parameter called ‘hill climbing rate (HCR)’. The HCR is the rate of using the HCO to improve the new harmony vector relative to the number of improvisations (NI). The HCO starts with a feasible new harmony generated by the original HSA operators. In each run, the HCO explores the neighbourhood structures of the new harmony and moves to another neighbouring solution that has a better or equivalent objective function value. Indeed, we use the best improvement and side walk acceptance criteria to guide the HCO, which means that the HCO accepts only the best local change (the change with the best objective function value) among all possible local changes. Side
walk means the current solution walk to its neighbouring solution without changing the objective function. We consider five different neighbourhood structures, as follows:

- N1- Move one event from a timeslot to another empty one;
- N2- Swap two events in two different timeslots;
- N3- Exchange three events in three separate timeslots;
- N4- Move one event from a room to another empty one in the same timeslot;
- N5- Swap two events in two different rooms in the same timeslot.

The first three neighbourhood structures directly affect the solution cost, because they are directly linked to time slots and events. This means that the local changes that are linked to the interaction between time slots and events affect the soft constraints S1, S2, and S3, especially by penalising the events with relation to the time slot. The other two remaining neighbourhood structures do not affect the solution cost since they execute the same time slot process which is mainly useful in the side walk. These two neighbourhood structures very usefully reveal more options to the HCO in the coming iteration, because events may acquire a room with a better size and features.

4.2.2 Hybridization with PSO

Memory consideration is the most vital operator in HSA. Most of the decision variables in the new harmony are selected from the other vectors stored in HM. In the timetabling case, the decision variables (or events) are, by nature, highly correlated, which means that the random selection of the value of the event from any HM vector might influence other events in the new harmony, and this in turn might violate some soft constraints. Therefore, the HSA has a low probability of generating a good-quality new harmony when the value of each single event is different in the HM vectors, which may get stuck in a bottleneck. Actually, in HSA, this limitation has little or no effect on the convergence behaviour of HSA while, in the proposed HHSA, such a limitation does practically impede the effect of the speed of convergence. Here, the question below should be explicitly answered:

Why does memory consideration work well in HSA while it does not work well in HHSA?

To the best of our knowledge, the vectors stored in HM on HSA smoothly converge to the same search space region during the generation process, in which any combination (or mixture) done by memory consideration to generate a new harmony leads to the same search space region, and the functionality of this operator is thus efficient. Keep in mind that random consideration and pitch adjustment are the source of exploration to other search space regions. In contrast, the vectors stored in HM in the proposed HHSA belong to different search space regions that have a different structure than that used by HCO. Thus, the functionality of the memory consideration is lost in most cases where diversity is too high.

To address this limitation arising from the memory consideration of HHSA, we need another selection criterion that reduces the selection pressure of memory consideration. Therefore, the new harmony cost in most iterations can be improved.

In this study, we modify the memory consideration to use ideas from Particle Swarm Optimization (PSO)[31]as an auxiliary rule to select a promising value of the decision variables from the vectors stored in HM. This auxiliary concept mimics the best harmony among the HM vectors to construct the new harmony. Figure 1, step 3, presents the pseudo-code for the PSO concepts. The memory consideration selects the value(or location) of the decision variables primarily from the best harmony stored in harmony memory, if it is not feasible; the values of the decision variables are selected from any valid value stored in any HM vectors.

Similar to our work presented in this section, Omran and Mahdavi [32] proposed a new variation of HSA called ‘global best harmony search’. In their work, the pitch adjustment, which was modified to mimic the best so-far solution obtained in HM, is similar as the PSO concepts.
To better understand our motivation for introducing our hybrid algorithm, recall two terms from metaheuristic algorithms: exploration and exploitation. Any effective and robust metaheuristic algorithm must be based on two search techniques to find a global optimal solution: exploration to reach not-yet-visited regions in the search space when it is necessary, and exploitation to make use of the previous visited search space regions to yield a high quality solution. These terms contradict each other. The suitable balance between both exploration and exploitation must be achieved in any metaheuristics to find high quality solutions.

Any successful metaheuristic (local or population-based) has components responsible for exploring the search space efficiently and effectively and exploitation to fine-tune the regions of search space that were already visited. This is achieved by the objective function consideration. The more the metaheuristic component observes the objective function of previous state already visited during the search, the more the search drifts toward exploitation. In contrast, the more the same component observes randomness, the more the search is drawn to exploration [33]. Briefly, the pure hill climbing algorithm has an exploitation component with no exploration power [34].

The convergence rate of any metaheuristic relies on the balance between exploration and exploitation during the search. In general, local search metaheuristics are very effective in exploring the single search space region (exploitation) and finding the local optimal solution in this region (exploitation), yet they fail to explore the entire search space and may get stuck in a locally optimal solution (lack of exploration). On the other hand, the component power of population-based algorithms is very powerful in visiting multiple search space regions at the same time (exploration), yet they cannot fine-tune promising regions to find a globally optimal solution (exploitation) in each region. As pointed out earlier, recent surveys of metaheuristics direct the new researcher to turn to hybridization between population-based and local search algorithms. For example, Blum and Roli [33] write that “… 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.”

---

5 As an example for the metaheuristics component [33]: In a population based algorithm like a genetic algorithm or ant colony system, the ‘recombination operator’ works from the argument that a better offspring is obtained by a collection of good pieces from its parents. This means that the higher the recombination rate (that is, the crossover rate) is increased, the more the offspring is like its parent. As such, the recombination operator is useful in exploitation. The opposite case takes place if it decreases. Another example of a metaheuristic is the acceptance criterion in simulated annealing, which depends on the value of temperature and the objective function. The high temperature that means the chance of accepting the uphill move increases (the amount of exploration increases). The opposite happens if it decreases. A final example of metaheuristics is the tabu list in tabu search, in which the more the size of the tabu list (that is, tabu tenure) increases, the more the diversification power increases.
Step 1. Initialize parameters

1. Initialize the problem parameters
   - \( f(x) \): objective function of timetable.
   - \( X \): set of decision variables (events)
   - \( X_i \): set of possible range for each decision variable in our study, all valid places of each event

2. Initialize HHSA parameters
   - HMCR: Harmony memory consideration rate.
   - PAR: Pitch Adjusting Rate.
   - NI: Number of Improvisations.
   - HMS: Harmony Memory Size
   - HCR: Hill Climbing Rate

Step 2. Initialize the harmony memory

1. for \( k = 1 \) to HMS do
2. begin
3. generate \( x^{NEW} \)
4. end
5. end for

Step 3. Improve a new harmony

1. for \( i = 1 \) to \( n_e \) do // \( n_e \) is the number of events
2. begin
3. if \( U(0,1) < \text{HMCR} \) then
4. begin
5. if \( x^{BEST}_{\text{NEW}} \) is valid then
6. \( x^{NEW} = x^{NEW}_{\text{BEST}} \)
7. else
8. \( x^{NEW} \in \{x^1_{\text{NEW}}, x^2_{\text{NEW}}, ..., x^{K}_{\text{NEW}}\} \)
9. end
10. if \( U(0,1) < \text{PAR1} \) then
11. begin
12. Pitch_Adjustment_Move(\( x^{NEW} \))
13. end
14. else
15. begin
16. random consideration
17. end
18. end
19. end
20. end for

Step 4. Update the harmony memory

1. calculate \( f(x^{NEW}) \)
2. if \( x^{NEW} \) is complete do
3. begin
4. if \( f(x^{NEW}) \) is better than the worst \( f(x^i) \) then
5. begin
6. include \( x^{NEW} \) from HM
7. end
8. end
9. end

Step 5. Check the stopping criterion

1. Repeat
2. step 3 and step 4
3. Until NI is satisfied
4. Done

Figure 1. The steps and pseudo-code of our hybrid algorithm.
5 Experimental results and analysis

We now turn our attention to evaluate our hybrid methods. Section 5.1 discusses the problem instances, and Section 5.2 presents the comparison results of our hybrid method and other presented in the previous study that use the same problem instances. Section 5.3 analyses different individual and hybrid algorithms proposed in our study. Finally, we illustrate an empirical study of the performance of our hybrid method on a different HCR in Section 5.4.

We ran the experiments on an Intel 2 GHz Core 2 Quad processor with 2 GB of RAM. We implemented our method in Microsoft Visual C++ version 6.0 under Windows XP.

5.1 The problem instances

The UCTP data used in the experiments in this study are freely available at http://iridia.ulb.ac.be/~msampels/tt.data/, prepared by Socha et al. [15]. For the purposes of our study, we call them the ‘Socha benchmark’. The 11 problem instances, which are grouped into five small problem instances, five medium problem instances and one large problem instance, have different levels of complexity and various sizes, as shown in Table 1. The solution to all problem instances must satisfy the hard constraints stated in Section 2.1. Furthermore, the solution cost is measured by the defined soft constraints violation.

<table>
<thead>
<tr>
<th>Class</th>
<th>Small</th>
<th>Medium</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of events</td>
<td>100</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>number of rooms</td>
<td>5</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>number of features</td>
<td>5</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>number of students</td>
<td>80</td>
<td>200</td>
<td>400</td>
</tr>
<tr>
<td>number of timeslots</td>
<td>45</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>approximate feature per room</td>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>percentage of feature use</td>
<td>70</td>
<td>80</td>
<td>90</td>
</tr>
<tr>
<td>maximum number of events per student</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>maximum number of students per event</td>
<td>20</td>
<td>50</td>
<td>100</td>
</tr>
</tbody>
</table>

5.1 Comparison with previous work

This section compares the best results obtained by HHSA to those applied by other methods in the literature to the same Socha benchmarks which are listed the Table 2 as follows:

- **RRLS** – Random Restart Local search [15].
- **MMAS** – MAX-MIN Ant System [15].
- **THH** – Tabu-search Hyper Heuristic [13].
- **VNS** – Variable Neighbourhood search [35].
- **FMHO** – Fuzzy Multiple Heuristic Ordering [12].
- **GHH** – Graph-based Hyper-Heuristic [14].
- **RII** – Randomised Iterative Improvement [36].
- **HSA** – Harmony Search Algorithm [19].
- **GD** – Great Deluge [37].
- **GDNLDR** – Great Deluge with Non-Linear Decay Rate [37].
- **HHSA** – The proposed Hybrid Harmony Search Algorithm.
We compare the obtained results of HHSA with those of previous methods in Table 2. The best results reported among all methods are highlighted. The comparisons are meant to show the ability of HHSA to find high quality solutions to the UCTP. In short, the HHSA outperformed the previous methods in four out of five medium problem instances. The results of HHSA also shared the same best known results with RII, HEA, and some results introduced by MMSA, THH, VNS, and GDNLDR for small problem instances. In addition, HHSA reports the second-best result in the large problem instance.

Table 2
A comparison of results on the small / medium/large data instances with previous methods.

<table>
<thead>
<tr>
<th></th>
<th>Small 1</th>
<th>Small 2</th>
<th>Small 3</th>
<th>Small 4</th>
<th>Small 5</th>
<th>Medium 1</th>
<th>Medium 2</th>
<th>Medium 3</th>
<th>Medium 4</th>
<th>Medium 5</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>RRLS Avg.</td>
<td>8</td>
<td>11</td>
<td>8</td>
<td>7</td>
<td>5</td>
<td>199</td>
<td>202.5</td>
<td>-</td>
<td>-</td>
<td>177.5</td>
<td>-</td>
</tr>
<tr>
<td>MMAS Avg.</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>195</td>
<td>184</td>
<td>284</td>
<td>164.5</td>
<td>219.5</td>
<td>851.5</td>
</tr>
<tr>
<td>THH Best</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>146</td>
<td>173</td>
<td>267</td>
<td>169</td>
<td>303</td>
<td>1166</td>
</tr>
<tr>
<td>VNS Best</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>317</td>
<td>313</td>
<td>375</td>
<td>247</td>
<td>292</td>
<td></td>
</tr>
<tr>
<td>FMHO Best</td>
<td>10</td>
<td>9</td>
<td>7</td>
<td>17</td>
<td>7</td>
<td>243</td>
<td>325</td>
<td>249</td>
<td>285</td>
<td>132</td>
<td>1138</td>
</tr>
<tr>
<td>GHH Best</td>
<td>6</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>372</td>
<td>419</td>
<td>359</td>
<td>348</td>
<td>171</td>
<td>1068</td>
</tr>
<tr>
<td>RII Avg.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>242</td>
<td>161</td>
<td>265</td>
<td>181</td>
<td>151</td>
<td></td>
</tr>
<tr>
<td>HEA Best</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>221</td>
<td>147</td>
<td>246</td>
<td>165</td>
<td>130 529</td>
<td></td>
</tr>
<tr>
<td>HSA Best</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>316</td>
<td>243</td>
<td>255</td>
<td>235</td>
<td>215</td>
<td></td>
</tr>
<tr>
<td>GD Best</td>
<td>17</td>
<td>15</td>
<td>24</td>
<td>21</td>
<td>5</td>
<td>201</td>
<td>190</td>
<td>229</td>
<td>154</td>
<td>222</td>
<td>1066</td>
</tr>
<tr>
<td>GDNLDR Best</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>140</td>
<td>130</td>
<td>189</td>
<td>112</td>
<td>141</td>
<td>876</td>
</tr>
<tr>
<td>HHSA Best</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
<td>102</td>
<td>135</td>
<td>122</td>
<td>123 671</td>
<td></td>
</tr>
</tbody>
</table>

We are particularly interested in comparing our results with HEA, which is the closest approach to our HHSA. The HEA hybridised variable neighbourhood structure with genetic algorithm (memetic algorithm), and our proposed algorithm also combined HCO with the harmony search algorithm. Both methods incorporate local search into population-based algorithms. Clearly, combining local search algorithms with population-based algorithms produced better results than others that were solely based either on local search or population-based algorithms separately in most cases. Note that any method can obtain good results by considering the right balance between global and local improvements incorporating both population and local search algorithms.

5.3 Performance of the proposed hybrid method

This section discusses the performance of five different methods proposed by us on the 11 Socha benchmarks. Specifically, these are the HHSA with PSO, HHSA without PSO, HSA with PSO, HSA without PSO, and HCO. The same program is used for all of these methods (in the second method and fourth one, the PSO codes are deactivated in the program), and the parameter configuration for each method is presented in Table 3. The first two methods are tested in the same parameter configurations to measure the effectiveness of the PSO on HHSA. Furthermore, the effectiveness of PSO upon the HSA is also studied by the experiments done in the third and fourth methods, which run in the same parameter configuration. Finally, we experiment with the HCO with more relaxed computational time than that used in HHSA.

Table 4 shows the comparison results obtained by the five proposed methods. The best results obtained are highlighted. Each method runs ten times for each Socha benchmark. The
best cost, the worst cost, and the approximate computational time among all runs are reported. We also report the average and standard deviation of the best results obtained for each of the ten runs. The results demonstrate that the HHSA with PSO can find higher quality solutions for all Socha benchmarks than those of others presented in this study.

### Table 3
The parameters used for different methods

<table>
<thead>
<tr>
<th>Parameters</th>
<th>HHSA with PSO</th>
<th>HHSA without PSO</th>
<th>HSA with PSO</th>
<th>HSA without PSO</th>
<th>HCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>NI</td>
<td>1.5×10^3</td>
<td>1.5×10^3</td>
<td>e×5</td>
<td>e×5</td>
<td>1</td>
</tr>
<tr>
<td>HMS</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>HMCR</td>
<td>98%</td>
<td>98%</td>
<td>98%</td>
<td>98%</td>
<td>100%</td>
</tr>
<tr>
<td>PAR1</td>
<td>2%</td>
<td>2%</td>
<td>2%</td>
<td>2%</td>
<td>0%</td>
</tr>
<tr>
<td>PAR2</td>
<td>4%</td>
<td>4%</td>
<td>4%</td>
<td>4%</td>
<td>0%</td>
</tr>
<tr>
<td>HCI*</td>
<td>2×10^3</td>
<td>2×10^3</td>
<td>0</td>
<td>0</td>
<td>e×5</td>
</tr>
<tr>
<td>HCR</td>
<td>30%</td>
<td>30%</td>
<td>0%</td>
<td>0%</td>
<td>100%</td>
</tr>
</tbody>
</table>

* HCI (the number of iteration of HCO).

Note that the numbers of iterations for both hybrid methods (HHSA with PSO and HHSA without PSO) are the same, but the results are significantly different, which demonstrates the effect of the PSO on HHSA convergence behaviour. See Figure 2 as an example of the effect of the PSO on HHSA, where the two trends are the average cost of HM vectors in each iteration. This means that the HHSA without PSO required more computational resources because it had to do additional iterations to be able to find the same results obtained by HHSA with PSO. For the large problem instance, the PSO helps the HHSA to find a feasible solution in each run.

Furthermore, we study the impact of PSO on the convergence behaviour of HSA. Here, we do ten runs for each Socha benchmark using two methods: the HSA with PSO and the HSA without PSO. Our results show that the PSO has not affected the ability of HSA to converge to a minimal solution, while it helps the HSA find a feasible solution for large problem instances.

Finally, we can observe that hybridisation between HSA as a global improvement search strategy and HCO as a local improvement search strategy, with auxiliary concepts from PSO, had a better solution cost than HSA without PSO and HCO, when each of them is individually running. This hybrid method stands out for its ability to strike a balance between global improvement and local improvement in a parallel optimisation environment.

![Figure 2](image_url)  
*Figure 2.* The impact of PSO on the HHSA for medium 1 problem instance.
Table 4
The performance of different variations of the proposed methods on each of the 11 Socha benchmarks. (Note that h means hours and m means minutes).

<table>
<thead>
<tr>
<th></th>
<th>HHS with PSO</th>
<th>HHS without PSO</th>
<th>HSA with PSO</th>
<th>HSA without PSO</th>
<th>HCO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best</td>
<td>Average</td>
<td>Worst</td>
<td>Std. dev.</td>
<td>Time</td>
</tr>
<tr>
<td>small1</td>
<td>Best</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>1.7</td>
<td>7.4</td>
<td>9.9</td>
<td>10.44</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>3</td>
<td>9</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Std. dev.</td>
<td>1.159</td>
<td>0.96</td>
<td>2.02</td>
<td>2.06</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>18.14h</td>
<td>17.04m</td>
<td>5.11m</td>
<td>4.83m</td>
</tr>
<tr>
<td>Small2</td>
<td>Best</td>
<td>0</td>
<td>6</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>1.3</td>
<td>7.7</td>
<td>12.1</td>
<td>10.9</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>3</td>
<td>9</td>
<td>16</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Std. dev.</td>
<td>1.33</td>
<td>0.94</td>
<td>2.18</td>
<td>1.79</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>32.45m</td>
<td>28.51m</td>
<td>4.55m</td>
<td>4.6m</td>
</tr>
<tr>
<td>Small3</td>
<td>Best</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>2.2</td>
<td>4.4</td>
<td>9.2</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>4</td>
<td>12</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Std. dev.</td>
<td>1.31</td>
<td>0.69</td>
<td>1.68</td>
<td>1.82</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>1.03h</td>
<td>47.56m</td>
<td>20.7m</td>
<td>20.53m</td>
</tr>
<tr>
<td>Small4</td>
<td>Best</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>2.77</td>
<td>5.1</td>
<td>7.1</td>
<td>5.8</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Std. dev.</td>
<td>1.71</td>
<td>1.19</td>
<td>1.44</td>
<td>1.98</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>38.7m</td>
<td>40.33m</td>
<td>12.19m</td>
<td>9.59m</td>
</tr>
<tr>
<td>Small5</td>
<td>Best</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>0</td>
<td>0.4</td>
<td>5.4</td>
<td>3.7</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>0</td>
<td>2</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Std. dev.</td>
<td>0.69</td>
<td>1.71</td>
<td>1.88</td>
<td>1.68</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>12.44m</td>
<td>9.47m</td>
<td>25.10m</td>
<td>29.5m</td>
</tr>
<tr>
<td>Medium1</td>
<td>Best</td>
<td>118</td>
<td>254</td>
<td>282</td>
<td>308</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>137.8</td>
<td>267.1</td>
<td>305.3</td>
<td>320.6</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>155</td>
<td>283</td>
<td>321</td>
<td>336</td>
</tr>
<tr>
<td></td>
<td>Std. dev.</td>
<td>12.56</td>
<td>8.96</td>
<td>11.17</td>
<td>9.93</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>2.06h</td>
<td>1.56h</td>
<td>25.57m</td>
<td>22.27m</td>
</tr>
<tr>
<td>Medium2</td>
<td>Best</td>
<td>106</td>
<td>270</td>
<td>336</td>
<td>282</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>126.9</td>
<td>275.7</td>
<td>351.7</td>
<td>300.1</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>144</td>
<td>284</td>
<td>366</td>
<td>317</td>
</tr>
<tr>
<td></td>
<td>Std. dev.</td>
<td>13.57</td>
<td>4.423</td>
<td>11.16</td>
<td>12.05</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>2.03h</td>
<td>2.14h</td>
<td>21.4m</td>
<td>20.53m</td>
</tr>
<tr>
<td>Medium3</td>
<td>Best</td>
<td>187</td>
<td>345</td>
<td>290</td>
<td>312</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>201.7</td>
<td>365.6</td>
<td>316.6</td>
<td>340.4</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>224</td>
<td>379</td>
<td>328</td>
<td>352</td>
</tr>
<tr>
<td></td>
<td>Std. dev.</td>
<td>12.15</td>
<td>9.21</td>
<td>11.87</td>
<td>13.47</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>1.54h</td>
<td>1.35h</td>
<td>22.47m</td>
<td>24.23m</td>
</tr>
<tr>
<td>Medium4</td>
<td>Best</td>
<td>130</td>
<td>255</td>
<td>245</td>
<td>251</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>145.2</td>
<td>271.44</td>
<td>266.4</td>
<td>271.9</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>162</td>
<td>283</td>
<td>298</td>
<td>284</td>
</tr>
<tr>
<td></td>
<td>Std. dev.</td>
<td>11.24</td>
<td>8.94</td>
<td>16.948</td>
<td>9.960</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>1.65h</td>
<td>2.23h</td>
<td>50.51m</td>
<td>34.73m</td>
</tr>
<tr>
<td>Medium5</td>
<td>Best</td>
<td>155</td>
<td>199</td>
<td>240</td>
<td>266</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>180.3</td>
<td>227.5</td>
<td>273.2</td>
<td>292.7</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>207</td>
<td>256</td>
<td>306</td>
<td>306</td>
</tr>
<tr>
<td></td>
<td>Std. dev.</td>
<td>17.61</td>
<td>19.14</td>
<td>22.97</td>
<td>12.23</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>4.6h</td>
<td>4.5h</td>
<td>35.93m</td>
<td>31.58m</td>
</tr>
<tr>
<td>Large</td>
<td>Best</td>
<td>706</td>
<td>-</td>
<td>734</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>771.5</td>
<td>-</td>
<td>849</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>889</td>
<td>-</td>
<td>925</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Std. dev.</td>
<td>64.22</td>
<td>-</td>
<td>58.83</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>2.36h</td>
<td>-</td>
<td>46 m</td>
<td>-</td>
</tr>
</tbody>
</table>
5.4 Empirical study of the effect of HCR on convergence rate of HHSA

In this section, we aim to evaluate our hybrid method with PSO on different HCR settings, bearing in mind that the same parameter settings stated in Table 3 are used with different HCR settings. As we explained earlier, HCR is the probability of using the HCO in our hybrid method at each iteration, that is, NI. For example, if HCR=50%, this means that the HHSA uses the HCO in each iteration with probability of 50%. Values of HCR variations are settings of three scenarios: too small being selected 5%, medium selected 30% and too large selected 80%. These variations of HCR ran ten times for each Socha problem instance. The best cost, worst cost, and the computational time are recorded in Table 5. The average and standard deviation indicate the average and standard deviation of the best values among the HM vectors obtained from each single run. The best result between HCR variants is highlighted.

From the results recorded in Table 5, we can see that the best results of all Socha benchmarks are obtained from a high setting of HCR, which means that using HCO with a high HCR improves our algorithm's performance. The computational time is significantly different in the three variants, where increasing the HCR leads to increasing computation time with a significant improvement in the solution quality.

A further effect of the variant HCR on our hybrid method can be seen in Figure 3 which demonstrates statistical evidence that the HHSA is very effective in obtaining a high quality solution and that the HCR has a strong effect on the behaviour of HHSA. Box-plots in Figure 2 demonstrate the approximation distribution of cost values of the HCR variants for each Socha problem instance. The box shows the cost values between first and third quartiles, while the bold line inside the box denotes the median of obtained costs. The lower and upper cost values are indicated by whiskers. The asterisks denote highly extreme cost values.

6 Conclusions and future work

In this paper, we presented HHSA to tackle UCTP. We have incorporated HCO with harmony search algorithms as a new operator to improve the quality of new harmony in each run with probability HCR. This idea stems from an analogy with MA to find a trade-off between the locally improved solution from HCO and the globally improved solution from HSA. The results demonstrate that our algorithm can find a high-quality solution with a reasonable computational time, compared to previous work.

Having been inspired by PSO concepts, we modified the memory consideration operator to mimic the best harmony so far found that is stored in harmony memory. This modification influences the behaviour of the proposed HHSA, being able to configure a good-quality new harmony that might be further improved by HCO.

In short, our HHSA with PSO concepts could find the best results for four out of five medium problem instances prepared by Socha et al. It also shared the best results with some past literature research on small problem instances, while the second best result for large problem instances is obtained. This shows that hybridisation between local search algorithms and population-based algorithms is promising in this research area and can introduce high-quality solutions.

In fact, our previous work and the contribution in this paper work on a feasible region of search space. The limitation mainly implies the difficulty of producing a feasible solution in each iteration, which also affects running time. Our future aim is to study the hybrid harmony search algorithm over the entire search space (feasible and infeasible regions) to diversify searching even more efficiently.

As a recommendation, we propose that the hybrid harmony and harmony search algorithm to the UCTP problem has broken new ground and thus can be used with great efficiency in the optimisation process for combining global and local components in one algorithm, which can then be used in other scheduling problems like examination timetabling and school timetabling.
Table 5
The performance of the proposed methods using different HCR setting. (Note that h means hours and m means minutes).

<table>
<thead>
<tr>
<th>HHSA</th>
<th>HCR ≤5%</th>
<th>HCR ≤30%</th>
<th>HCR ≤80%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small1</td>
<td>Best</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>2.1</td>
<td>1.7</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Std.dev.</td>
<td>1.28</td>
<td>1.159</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>9.87 m</td>
<td>18.14 m</td>
</tr>
<tr>
<td>Small2</td>
<td>Best</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>5.1</td>
<td>1.3</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Std.dev.</td>
<td>1.72</td>
<td>1.33</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>9.66m</td>
<td>32.45m</td>
</tr>
<tr>
<td>Small3</td>
<td>Best</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>3.4</td>
<td>2.2</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Std.dev.</td>
<td>1.42</td>
<td>1.31</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>21.3m</td>
<td>1.03 h</td>
</tr>
<tr>
<td>Small4</td>
<td>Best</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>2.8</td>
<td>2.77</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Std.dev.</td>
<td>1.61</td>
<td>1.71</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>15.3m</td>
<td>38.7 m</td>
</tr>
<tr>
<td>Small5</td>
<td>Best</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Std.dev.</td>
<td>0.31</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>7.3 m</td>
<td>12.44 m</td>
</tr>
<tr>
<td>Medium1</td>
<td>Best</td>
<td>106</td>
<td>118</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>140.4</td>
<td>137.8</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>156</td>
<td>155</td>
</tr>
<tr>
<td></td>
<td>Std.dev.</td>
<td>14.36</td>
<td>12.56</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>1.4 h</td>
<td>2.06 h</td>
</tr>
<tr>
<td>Medium2</td>
<td>Best</td>
<td>121</td>
<td>106</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>139.3</td>
<td>126.9</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>163</td>
<td>144</td>
</tr>
<tr>
<td></td>
<td>Std.dev.</td>
<td>13.94</td>
<td>13.57</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>1.37 h</td>
<td>2.03 h</td>
</tr>
<tr>
<td>Medium3</td>
<td>Best</td>
<td>178</td>
<td>187</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>199.5</td>
<td>201.7</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>216</td>
<td>224</td>
</tr>
<tr>
<td></td>
<td>Std.dev.</td>
<td>11.04</td>
<td>12.15</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>1.4 h</td>
<td>2.54 h</td>
</tr>
<tr>
<td>Medium4</td>
<td>Best</td>
<td>124</td>
<td>130</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>139.2</td>
<td>145.2</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>163</td>
<td>162</td>
</tr>
<tr>
<td></td>
<td>Std.dev.</td>
<td>14.38</td>
<td>11.24</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>55.57m</td>
<td>1.65 h</td>
</tr>
<tr>
<td>Medium5</td>
<td>Best</td>
<td>163</td>
<td>155</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>194.3</td>
<td>180.3</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>244</td>
<td>207</td>
</tr>
<tr>
<td></td>
<td>Std.dev.</td>
<td>21.89</td>
<td>17.61</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>2.19 h</td>
<td>4.6 h</td>
</tr>
<tr>
<td>Large</td>
<td>Best</td>
<td>678</td>
<td>706</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>822.1</td>
<td>771.5</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>963</td>
<td>889</td>
</tr>
<tr>
<td></td>
<td>Std.dev.</td>
<td>87.85</td>
<td>64.22</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>1.62h</td>
<td>2.36h</td>
</tr>
</tbody>
</table>
Figure 3. The result of the proposed hybrid method with variant HCR where HCR= 5%, HCR=30%, and HCR=80%.
References


