Applied Soft Computing 35 (2015) 726–739

Contents lists available at ScienceDirect

Applied Soft Computing

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

A hybrid artificial bee colony for a nurse rostering problem

Mohammed A. Awadallah[a,*], Asaju La'aro Bolaji[b], Mohammed Azmi Al-Betar[c]

[a] Department of Computer Science, Al-Aqsa University, P.O. Box 4051, Gaza, Palestine
[b] Department of Computer Science, University of Ilorin, P.M.B. 1515, Ilorin, Nigeria
[c] Department of Information Technology, Al-Huson University College, Al-Balqa Applied University, P.O. Box 50, Al-Huson, Irbid, Jordan

ARTICLE INFO

Article history:
Received 24 February 2015
Received in revised form 20 June 2015
Accepted 2 July 2015
Available online 17 July 2015

Keywords:
Metaheuristics
Rostering
Artificial bee colony
Nurse rostering problem
Hill climbing

ABSTRACT

The nurse rostering problem (NRP) is a combinatorial optimization problem tackled by assigning a set of shifts to a set of nurses, each has specific skills and work contract, to a predefined rostering period according to a set constraints. The metaheuristics are the most successful methods for tackling this problem. This paper proposes a metaheuristic technique called a hybrid artificial bee colony (HABC) for NRP. In HABC, the process of the employed bee operator is replaced with the hill climbing optimizer (HCO) to empower its exploitation capability and the usage of HCO is controlled by hill climbing rate (*HCR*) parameter. The performance of the proposed HABC is evaluated using the standard dataset published in the first international nurse rostering competition 2010 (INRC2010). This dataset consists of 69 instances which reflect this problem in many real-world cases that are varied in size and complexity. The experimental results of studying the effect of HCO using different value of *HCR* show that the HCO has a great impact on the performance of HABC. In addition, a comparative evaluation of HABC is carried out against other eleven methods that worked on INRC2010 dataset. The comparative results show that the proposed algorithm achieved two new best results for two problem instances, 35 best published results out of 69 instances as achieved by other comparative methods, and comparable results in the remaining instances of INRC2010 dataset.

1. Introduction

Nurse rostering problem (NRP) involves assigning a set of nurses with different working skills and working contracts to a set of shifts of different types, taking into account a set of *hard* and *soft* constraints. The hard constraint satisfaction is mandatory in any roster to be *feasible* while the violation of the soft constraints is allowable but it should be minimize as much as possible. Normally, the satisfaction of some soft constraints might be lead to violate others. Therefore, in most cases, finding a global optimal solution is almost impossible [1]. Since NRP is among the most prevalent problems that are being tackle in the domain of automated timetabling and operations research, researchers have proposed several algorithmic techniques over the last few decades [2–4]. Some of these techniques are the mathematical programming, graph-based heuristics, metaheuristic-based method and hyper-heuristics.

In particular, the metaheuristic-based method is normally divided into local-based search methods such as Tabu Search [5,6], Simulated Annealing [7], Variable Neighbourhood Structure [8–10] and population-based methods which include Ant Colony optimization algorithm [11], Electromagnetic algorithm [12], Genetic Algorithm [13–15], Scatter Search [16]. Note that the major concern of local search-based methods is to exploit the problem search space whereas population-based methods focus more on the exploration [17]. *Exploration* is the process of visiting the not-yet-visited search space regions, while *exploitation* is the process of makes use of accumulative search [18]. Due to the nature of the NRP search space, the researchers in this domain introduced a more robust solution method of solving this problem, named hybrid approaches. This hybrid approaches involve hybridizing the local-based search method within the population-based method in order to strike a balance between exploration and exploitation [19]. In these approaches, the local search-based methods work as a local exploiter to find the local optima at each region of the search, while the population-based methods work as a global explorer to diversify the search. The main advantage obtained from the usage of the hybrid method is that the search space of the solutions in the population is reduced to the subspace of local optima. In [20],

* Corresponding author. Tel.: +970 599686748.
E-mail addresses: ma.awadallah@alaqsa.edu.ps (M.A. Awadallah), lbasaju@unilorin.edu.ng (A.L. Bolaji), mohbetar@bau.edu.jo (M.A. Al-Betar).

http://dx.doi.org/10.1016/j.asoc.2015.07.004
1568-4946/© 2015 Elsevier B.V. All rights reserved.

the genetic algorithm (GA) is combined with Simulated Annealing hyper-heuristic as a local search procedure to tackle this problem. In another study, the NRP is tackled with the hybridization of the steepest descent improvement within GA [21]. Similarity, the GA is integrated with hill climbing optimizer (HCO) to solve this problem [22].

In the recent times, a new swarm intelligence population-based metaheuristic algorithm based on the foraging behaviour of honey bees, namely artificial bee colony algorithm (ABC), has been initially proposed by Karaboga in [23]. ABC is exemplified as a simple idea which is very easy to implement with only a few parameters and high degree computational efficiency. Based on these advantages, ABC has successfully been used to tackle numerous optimization problems [24,25]. In particular, it has been successfully been adapted and hybridized with local search-based methods to empower its exploitation capability. Bolaji et al. [26,27] proposed a hybrid approach combining ABC with the HCO for the university timetabling problems. In another research, the ABC is hybridized with a greedy heuristic and a local search for the quadratic knapsack problem [28]. Furthermore, the tabu search algorithm is combined with three operators of ABC for solving the flexible job shop scheduling problems [29].

Therefore, this paper presents a new hybrid algorithm that hybridized HCO within the operators of the ABC for NRP and henceforth called HABC. In HABC, the functionality of the employed bee operator is replaced by HCO to empower its exploitation properties and the usage of HCO is controlled by hill climbing rate (*HCR*) parameter. The performance of the proposed method is evaluated using the standard dataset published in the first international nurse rostering competition (INRC2010) [30]. Experimental results and comparisons demonstrate the effectiveness of the proposed algorithm for NRP.

In Section 2 the description of the NRP is presented, while the presentation of the original ABC is given in Section 3, this is followed by the proposed approach as described with detailed information in Section 4. Next, the experimental results are presented in Section 5. Finally, conclusions are considered in Section 6.

Table 1
Description of hard and soft constraints defined in INRC2010.

Symbol	The constraint
H_1	All demanded shifts during the scheduling period must be met.
H_2	A nurse cannot work more than one shift per day.
S_1	Maximum and minimum number of shifts should be assigned to a nurse during the scheduling period.
S_2	Maximum and minimum number of consecutive working days.
S_3	Maximum and minimum number of consecutive free days.
S_4	Maximum number of consecutive working weekends.
S_5	Maximum number of working weekends in four weeks.
S_6	Number of days off after a series of night shifts.
S_7	Assign complete weekends. (i.e., if a nurse has to work only on some days of the weekend then penalty occurs).
S_8	Assign identical complete weekends. (i.e., Assignments of different shift types to the same nurse during a weekend are penalized).
S_9	Number of free days after a series of night shifts.
S_{10}	Requested Day off/on. (i.e., if a nurse prefers to work or not to work on specific days of the week not respected then penalty occurs).
S_{11}	Requested Shift off/on. (i.e., similar to the previous but now for specific shifts on certain days).
S_{12}	Alternative skill. (i.e., If an assignment of a nurse to a shift type requiring a skill that he/she does not have occurs, then the roster is penalized accordingly).
S_{13}	Unwanted patterns. (i.e., an unwanted pattern is a sequence of assignments that is not in the preferences of a nurse based on his/her contract).

2. Problem description

The NRP involves the allocation of specific number of nurses to a set of shift types for a predefined rostering period. Shift type is the term used for representing a time frame for which a nurse with specific skill is needed. The solution roster is subjected to two types of constraints: *hard* and *soft*. The problem considered in this research includes two hard constraints (i.e., H_1, H_2 as shown in Table 1) that must be satisfied. The two hard constraints including that the roster should provide a specific number of nurses for each shift and a nurse can work at only one shift per day. The other type of constraints (i.e., S_1–S_{13} as shown in Table 1) are considered as soft and in which their violations must be minimized as much as possible. The roster is *feasible* if it satisfies all hard constraints and its *quality* is determined by calculating the penalty of violating the soft constraints. The basic objective is to find a feasible roster with a good enough quality.

Formally, the problem instances include a set $\mathcal{N} = \{n_0, n_1, \ldots, n_{m-1}\}$ of nurses, each has a specific skill from the set of skill categories $\mathcal{K} = \{k_0, k_1, \ldots, k_{q-1}\}$. Each nurse has a specific work contract from the set of contracts $\mathcal{C} = \{c_0, c_1, \ldots, c_{w-1}\}$. A set $\mathcal{D} = \{d_0, d_1, \ldots, d_{b-1}\}$ of days represent the rostering period, and each day split to a set $\mathcal{S} = \{s_0, s_1, \ldots, s_{r-1}\}$ of shift types. It should be noted that the mathematical formulation of the hard and soft constraints can be seen in [31,32].

The nurse roster is evaluated using the objective function formalized in Eq. (1) that adds up the penalty of soft constraint violations in a feasible roster.

$$\min f(\boldsymbol{x}) = \sum_{s=1}^{13} c_s . g_s(\boldsymbol{x}). \tag{1}$$

Note that s refers to the index of the soft constraint, c_s refers to the penalty weight for the violation of the soft constraint s, and $g_s(\boldsymbol{x})$ is the total number of violations of the soft constraint s in solution roster x.

Fig. 1 shows illustrative example of a feasible nurse roster, an instance of allocating a six nurses to a set of four shifts over 28 days of rostering period. Each row represents a schedule of each nurse, while each column represents a day. The content in a cell represents the shift type allocated to a nurse. It should be noted that each nurse allocated to one shift at most per day, where the number of shifts assigned to each nurse during the rostering period should be within the range of shifts agreed in his/her contract.

The problem instances in the INRC2010 dataset were classified into three tracks: sprint, medium, and long instances that are varying in size and complexity. Each track of this competition is grouped into four types in terms of their publication time in the competition: early, late, hidden, and hint. The sprint track instances are divided into 10 early, 10 hidden, 10 late, and 3 hint. However, the medium and long tracks are divided into 5 early, 5 hidden, 5 late, and 3 hint.

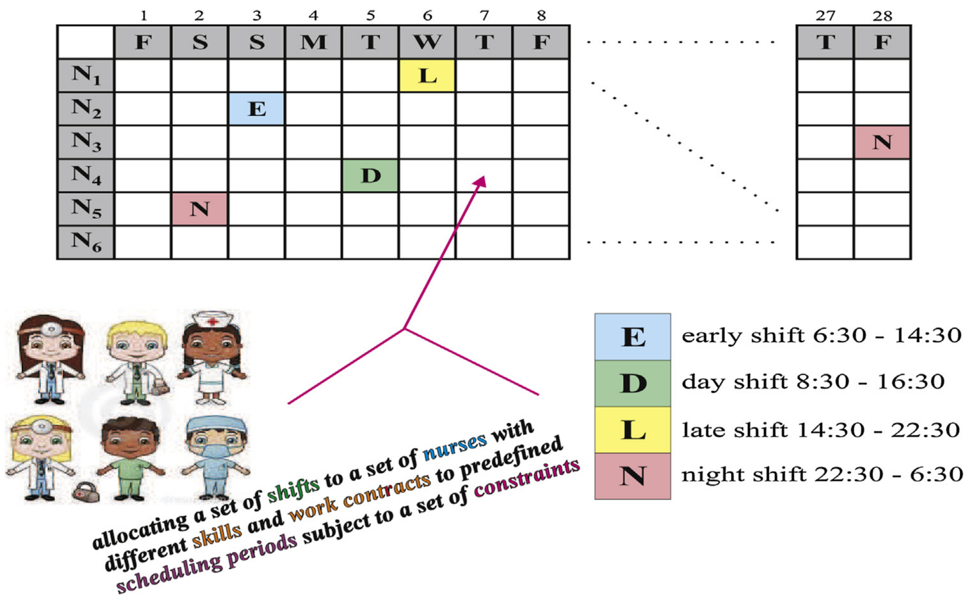

1 2 3 4 5 6 7 8 27 28
F S S M T W T F T F
N1 N2 N3 N4 N5 N6
L E D N N
E early shift 6:30 - 14:30
D day shift 8:30 - 16:30
L late shift 14:30 - 22:30
N night shift 22:30 - 6:30
allocating a set of shifts to a set of nurses with different skills and work contracts to predefined scheduling periods subject to a set of constraints

Fig. 1. Illustrative example of a feasible nurse roster.

A brief summary of the characteristics of the INRC2010 dataset is fully given in Table 2. This table includes the number of nurses, the nurse skills, the shift types, the work contracts, the number of unwanted patterns, and the existence of day-off and shifts-off nurse preferences.

For example, *Long _ early*05 instance refers to a ward of hospital with 49 nurses. These nurses are grouped into two types of skills: *nurse* and *head-nurse*, and three types of work contracts. Each working day is divided into five types of shifts (i.e., day, night, early, late, and day-head). Furthermore, three types of unwanted patterns should be avoided during the construction process of roster. In contrast, the day-off and shift-off nurse preferences should be respected as much as possible in the roster.

2.1. Previous methods

The NRP has been widely studied by operational research and artificial intelligence research communities during the past five decades [2–4]. Several methods which have been proposed to solve the NRP using the INRC2010 dataset shall be briefly review as follows:

Valouxis et al. [33], the winner of the challenge, utilized the Integer Programming with local search procedures to solve NRP using INRC2010 dataset. Their solution method consists of two phases: In the first phase, the different nurses are allocated to the different working days, while in the second phase the shifts are assigned to the nurses. For medium and long track instances, they employed three local search procedures to enhance the exploitation capability of their method in the first phase. These three local search procedure are (i) rescheduling one day, (ii) rescheduling two days, and (iii) reshuffling the shifts among nurses.

Bilgin et al. [34], combined hyper-heuristic with a greedy shuffle heuristic for tackling the problem. The hyper-heuristic is initially adapted to generate a feasible roster that satisfied the soft constraints as much as possible. Then the greedy shuffle heuristic is used for further improvements. In addition, the authors provided the computational results of Integer Linear Programming using IBM CPLEX.

In another study, Burke and Curtois [35], adapted ejection chain-based method called Variable Depth Search (VDS) and branch and price method separately for solving the problem. The experimental results show that the branch and price algorithm outperforms the VDS especially for medium and long track instances.

Nonobe [36], modeled NRP as Constraint Optimization Problem, and then used the “COP solver” based on tabu search algorithm. It should be noted that their method had previously been used to solve other timetabling problems [37].

Similarly, the adaptation of tabu search is investigated by Lü and Hao [38] to solve the NRP. The proposed solution method consist of two phases: (i) random heuristic is used to construct a feasible roster by allocating different nurses to the different shifts randomly. And (ii) the tabu search algorithm is used to improve the feasible roster locally with the aid of two neighbourhood structures (i.e., move and swap).

Santos et al. [39], presented Integer Programming technique to tackle the problem by decomposing the problem into subproblems. The violations of these subproblems are handled as must as possible to speed up the improvement of feasible solutions. Later on, the Variable Neighborhood Descent (VND) is used for further improvements. Note that the local search procedures used is similar to the one employed in [33].

The hybridization of harmony search algorithm with HCO is presented by Awadallah et al. [40]. The HCO is combined within the harmony search algorithm to improve the solution locally till the local optima is obtained. It is noteworthy that some of the local search procedures in the HCO were used for the first time in the nurse rostering domain. Finally, the harmony search hyper-heuristic is proposed by Anwar et al. [41] to solve this problem with promising results.

As shown in the literature, none of these methods has been able to achieved exact solution due to the nature of the NRP search space. Clearly, the research in this domain is still active and presentation of a new method maybe able to achieve better results.

3. Artificial bee colony algorithm

Artificial bee colony algorithm proposed in [23], is recently emerged as one of most popular nature-inspired algorithms which

Table 2
The characteristics of the INRC2010 dataset.

Problem instance	Nurses	Skills	Shifts	Contracts	Unwanted patterns	Day off	Shift off
*Sprint*_*early*01, *Sprint*_*early*02, *Sprint*_*early*03, *Sprint*_*early*04, *Sprint*_*early*05, *Sprint*_*early*06, *Sprint*_*early*07, *Sprint*_*early*08, *Sprint*_*early*09, *Sprint*_*early*10.	10	1	4	4	3	√	√
*Sprint*_*hidden*01, *Sprint*_*hidden*02, *Sprint*_*hidden*06, *Sprint*_*hidden*07.	10	1	3	3	4	√	√
*Sprint*_*hidden*03, *Sprint*_*hidden*04, *Sprint*_*hidden*05, *Sprint*_*hidden*08, *Sprint*_*hidden*09, *Sprint*_*hidden*10, *Sprint*_*late*01, *Sprint*_*late*03, *Sprint*_*late*04, *Sprint*_*late*05, *Sprint*_*hint*01, *Sprint*_*hint*03.	10	1	4	3	8	√	√
*Sprint*_*late*02.	10	1	3	3	4	√	√
*Sprint*_*late*06, *Sprint*_*late*07, *Sprint*_*late*10.	10	1	4	3	0	√	√
*Sprint*_*late*08, *Sprint*_*late*09.	10	1	4	3	0	*X*	*X*
*Sprint*_*hint*02.	10	1	4	3	0	√	√
*Medium*_*early*01, *Medium*_*early*02, *Medium*_*early*03, *Medium*_*early*04, *Medium*_*early*05.	31	1	4	4	0	√	√
*Medium*_*Hiiden*01, *Medium*_*Hiiden*02, *Medium*_*Hiiden*03, *Medium*_*Hiiden*04, *Medium*_*Hiiden*05	30	2	5	4	9	*X*	*X*
*Medium*_*late*01, *Medium*_*hint*01, *Medium*_*hint*03.	30	1	4	4	7	√	√
*Medium*_*late*02, *Medium*_*late*04.	30	1	4	3	7	√	√
*Medium*_*late*04, *Medium*_*hint*02.	30	1	4	4	0	√	√
*Medium*_*late*05.	30	2	5	4	7	√	√
*Long*_*early*01, *Long*_*early*02, *Long*_*early*03, *Long*_*early*04, *Long*_*early*05.	49	2	5	3	3	√	√
*Long*_*hidden*01, *Long*_*hidden*02, *Long*_*hidden*03, *Long*_*hidden*04, *Long*_*late*01, *Long*_*late*03, *Long*_*hint*01.	50	2	5	3	9	*X*	*X*
*Long*_*hidden*05, *Long*_*late*02.	50	2	5	4	9	*X*	*X*
*Long*_*late*04.	50	2	5	5	9	*X*	*X*
*Long*_*hint*02, *Long*_*hint*03.	50	2	5	3	7	*X*	*X*

has bee utilized for tackling practical optimization problems [24,25]. This algorithm carry out a search by evolving a population of food sources (i.e. solutions) with the aid of nondeterministic operators (e.g., employed, onlooker and scout), by iteratively improving the solutions using these operators until some criterion of convergence has been achieved. Studies have shown that ABC algorithm has offer significant advantages such as simplicity and ease of implementation over other traditional algorithms when applied to tackle many practical optimization problems [24,25,42]. The colony of ABC comprises three groups of bees: employed bees, onlookers and scouts. A bee on the dance floor waiting for dance behaviour of employed bee in order to choose a desired food source is referred to as onlooker while the one sharing the information of the food source visited by it before on the dance area is called employed bee. The other category of bee that carries out random search for discovering new sources is called scout bee. The position of a food source is given as a possible position of the solution to the optimization problem, furthermore, the nectar quantity of a food source corresponds to the fitness (i.e., quality) of the associated food source.

In ABC algorithm, the first part of the colony comprises employed bees whereas the second part contains the onlookers. The number of the employed bees or the onlooker bees is equal to the number of food sources (i.e. the solutions) in the food source memory (i.e., population). At the initial stage, the ABC generates initial population of food source randomly where the population of the solutions is subjected to repeat search processes with the aid of three operators (i.e. employed, onlooker and scout bees). An employed bee modifies the position of the solution in her memory based on the local information and calculates the nectar quality of the new solution. Provided that, the quality of the new solution is better than that of the previous one, then it memorizes the position of the new solution. If not, the position of the previous one remains in its memory. After the search process is completed by all employed bees. Then the nectar and position information of all food sources are shared with the onlooker bee on the dance area. An onlooker bee calculates the nectar information taken from all employed bees and selects a solution with higher probability. Using similar strategy as in the case of the employed bee, the onlooker modifies the position of the solution in her memory and evaluates the nectar quality of the solution. If better than that of the previous one, the new position is memorized by the bee and forgets the old one. A new solution is generated randomly by a scout bee to replace a food source whose nectar quality is abandoned by the onlooker bees. The search process of the ABC is given in Algorithm 1 as follows:

Algorithm 1. Schematic pseudo-code of ABC procedure

```
Initialization of the ABC and problem parameters
Initialization of the food source memory
repeat
   Send the employed bees to the food sources.
   Send the onlooker bees to select a food source.
   Send the scout bees to search possible new food sources.
   Memorize the best food source.
until(termination criterion are met)
```

The description of the detailed search process of the ABC algorithm is given in [25]. It is worthy of mentioning that other version of ABC have been recently proposed in which the scout bee phase of the old version of ABC which was based on generations is modified to count fitness evaluations consumed [43]. However, the research in this study is based on the old version of ABC.

Note that ABC algorithm as a nature-inspired algorithm has some similarity with ant colony optimization because both algorithms exchange information on the quality of their solutions through a memory structure [44]. However, ant colony system shares their information through the pheromone whereas in ABC, the employed bee communicates their information with other bees through the use of waggle dance in the dance floor [44].

4. Proposed algorithm

This section provides the detailed descriptions of hybridization of HCO within employed bees operator of the ABC algorithm for NRP. First of all, the description of original ABC as adapted for the NRP shall be discussed, this is followed by hybridization of ABC algorithm.

4.1. Artificial bee colony for NRP

In this section, the ABC algorithm is adapted for tackling the NRP in which different neighbourhood structures are integrated with the operators of the ABC to cope with the nature of NRP search space.

In ABC, any NRP solution (i.e. roster) is represented as a vector of allocations $\boldsymbol{x} = (x^1, x^2, \ldots, x^N)$, of which each allocation x^i takes three values (nurse number, day number, shift number). For instance, let $\boldsymbol{x} = ((1, 2, 1), (1, 3, 2), \ldots, (5, 5, 4))$ be a feasible roster. Then, this roster is interpreted by the ABC algorithm as follows: the allocation $x^1 = (1, 2, 1)$ means nurse n_1 is assigned to shift s_1 at day d_2. The second allocation $x^2 = (1, 3, 2)$ means to nurse n_1 assigned to shift s_2 at day d_3, and so on. The six main procedural steps of ABC algorithm for tackling NRP are described bellow and Algorithm 2 shows the pseudo-code.

Algorithm 2. ABC algorithm for NRP

```
STEP I    Initialize the parameters of NRP and HABC
          1: Set the NRP parameters drawn from the INRC2010 dataset.
          2: Set the HABC parameters ($SN$, $MCN$, $limit$).
          3: Define the roster representation and utilize the objective function.
STEP II   Initialization of the Food Source Memory
          1: for $s = 1$ to $SN$ do
          2:    $\mathbf{x}_s \leftarrow$ heuristic ordering strategy
          3:    $f_s = f(\mathbf{x}_s)$
          4: end for
          5: Sorts($\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{SN}$),   where $f_1 \le f_2 \le \cdots \le f_{SN}$
STEP III  Send the Employed Bees to the Food Sources
          1: for $s = 1$ to $SN$ do
          2:    $r = rand(1, 4)$
          3:    $\mathbf{x}'_s = N_r(\mathbf{x}_s)$   $N_1$ = MNS(), $N_2$ = SNS(), $N_3$ = SPS(), $N_4$ = TRM()
          4:    if ($f(\mathbf{x}'_s) < f_s$) then
          5:       $\mathbf{x}_s = \mathbf{x}'_s$
          6:       $f_s = f(\mathbf{x}'_s)$
          7:       $trial_s = 0$
          8:    else
          9:       $trial_s$++
          10:   end if
          11: end for
STEP IV   Send the Onlooker Bees to the Food Sources
          1: $r = rand(0, 1)$
          2: $found = false$
          3: $i = 1$, $sum\_prob$=0
          4: while($i \le SN$ AND NOT ($found$)) do
          5:    $sum\_prob = sum\_prob + p_i$
          6:    if ($sum\_prob \ge r$) then
          7:       $s = i$
          8:       $found = true$
          9:    else
          10:      $i$++
          11:   end if
          12: end while
          13: $r = rand(1, 4)$
          14: $\mathbf{x}'_s = N_r(\mathbf{x}_s)$
          15: if ($f(\mathbf{x}'_s) < f_s$) then
          16:    $\mathbf{x}_s = \mathbf{x}'_s$
          17:    $f_s = f(\mathbf{x}'_s)$
          18:    $trial_s = 0$
          19: else
          20:    $trial_s$++
          21: end if
STEP V    Send the Scout to Search for Possible New Food Sources
          1: $s = rand(1, SN)$
          2: if ($trial_s \ge limit$) then
          3:    $\mathbf{x}_s$ = random($\mathbf{x}_s$)
          4:    $f_s = f(\mathbf{x}_s)$
          5:    $trial_s$=0
          6: end if
STEP VI   Stopping condition
          1: while ($time \le MCN$) do
          2:    Repeat STEP III to STEP V
          3: end while
```

STEP I: *Initialization of ABC and NRP parameters.* In this step, the three control parameters of the ABC algorithm which are needed for tackling the NRP are initialized. These parameters include solution number (*SN*) which is the number of solutions (i.e., food sources) in the population and similar to the population size in GA; maximum cycle number (*MCN*) that corresponds to the maximum number of iterations; and *limit* that is responsible for the abandonment of solution, whenever it is not improved for certain number of iterations and basically it is utilized to diversify the search. Similarly, the NRP parameters are initiated by extracting them from the problem instances of the INRC2010 dataset such as the set of nurses, the set of skill categories, the set of shift types, the scheduling period, the set of work contracts, matrix of weekly nurse demand, matrices of nurses preferences (i.e., shift-off, shift-on, day-off, and day-on), and eventually the set of unwanted patterns. The job specification such as maximum/minimum number of assignments over rostering period, maximum/minimum number of consecutive working days, maximum/minimum number of consecutive free days, the number of days-off after a series of night shifts, maximum number

Table 3
Ordering of shifts.

Shift	Weekly nurses demand							Sum of demand	Shift ordering
	M	T	W	T	F	S	S		
D	3	3	3	3	3	2	2	19	4
L	2	2	2	2	2	1	1	12	2
E	2	2	2	2	2	2	2	14	3
N	1	1	1	1	1	1	1	7	1

of consecutive working weekends, and maximum working weekend in four weeks. It should be noted that contents of the problem instances are modified to number format for the initialization of the NRP parameters.

STEP II: *Initialization of the food source memory*. The food source memory (FSM) is the memory allocation spaces that comprises set of feasible food source (i.e., rosters) as determined by *SN* as shown in Eq. (2). In this step, the set of feasible rosters are generated using the heuristic ordering strategy and stored in ascending order in FSM with respect to the objective function values that is $f(\mathbf{x}_1) \le f(\mathbf{x}_2) \le \cdots \le f(\mathbf{x}_{SN})$. The heuristic ordering is utilized to sorts the different shifts in ascending order based on the level of difficulty. Note that the required nurses per week (i.e., weekly nurses demand) for the different shifts is used as a difficulty factor. The shift with the lesser required nurses is the most difficult to be assigned and thus the nurses with shift are allocated first before the other. Later, the nurses are assigned to the second lesser weekly nurses demand till the last shift.

$$\mathbf{FSM} = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^N \\ x_2^1 & x_2^2 & \cdots & x_2^N \\ \vdots & \vdots & \ddots & \vdots \\ x_{SN}^1 & x_{SN}^2 & \cdots & x_{SN}^N \end{bmatrix} \begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_{SN}) \end{bmatrix} \qquad (2)$$

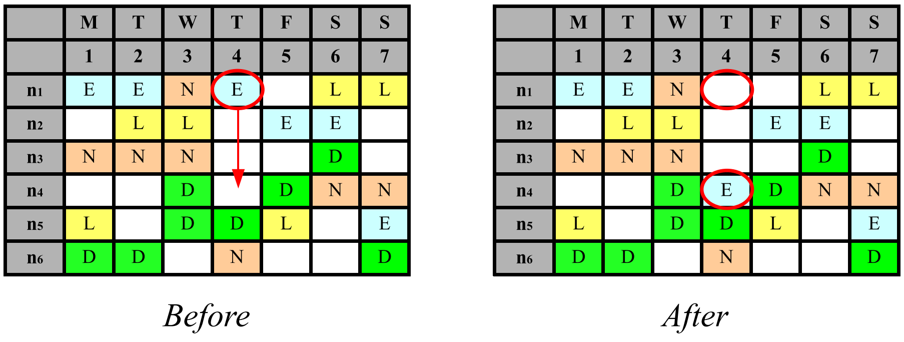
Table 3 provides an illustrative example of different shifts arrangement using the heuristic ordering strategy. In this table, the summation of the weekly nurse demand for each shift is available in the second-to-last column. The shift with the lesser value in the sum of demand column is ranked first by the heuristic ordering strategy (see the last column in Table 3) and so on. In the construction process, the shift with the prior rank is assigned to the different nurses first, followed by the shift with the second rank, until the shift with the last rank is assigned. It should be noted that the shift is eligible to assign for any nurse if the hard constraints are preserved.

STEP III: *Send the employed bees to the food sources.* In this step, the set of feasible rosters are sequentially selected by the employed bee operator from the FSM in which each roster are exploited using the set of neighbourhood structures in order to produce a new set of neighbouring rosters (i.e., new food sources). The neighbourhood structures used by employed bee operator are:

- **Move Neighbourhood Structure** (i.e., MNS()): This neighbourhood is used to move the shift of current nurse to another nurse selected randomly while maintaining feasibility. Fig. 2 shows an example of this procedure, where the early shift *E* is released from nurse n_1 and reassigned to the nurse n_4.
- **Swap Neighbourhood Structure** (i.e., SNS()): In this neighbourhood, the shift allocated to a specific nurse is exchanged with the shift of another nurse while maintaining feasibility. Note that both nurses are assign to two different shifts on the same day. An example of this procedure is given in Fig. 3, where the night shift *N* of nurse n_1 is exchanged with the early shift *E* of the nurse n_3.
- **Swap Pattern of Shifts** (i.e., SPS()): This neighbourhood is used to exchange a pattern of shifts among two nurses, if the feasibility is maintained. Fig. 4 shows an example of this procedure, where the two consecutive shifts for nurse n_3 are exchanged with another two consecutive shifts of the nurse n_5.
- **Token Ring Move** (i.e., TRM()): If the shift of a specific nurse is violated the soft constraint S_7 (i.e. partial weekend), then this shift is moved to another nurse who has incomplete weekend. Furthermore, If the complete weekend is not identical (i.e. soft constraint S_8), then the shift is exchanged with another nurse who has shift on the same day. Fig. 5 shows an example of this procedure, where the early shift *E* of nurse n_5 is moved to the nurse n_3 to solve the violations of the soft constraint S_7 (i.e. complete weekend). Furthermore, the early shift *E* of nurse n_3 on Sunday is exchanged with the day shift *D* of the nurse n_1 on the same day to solve the violation of the soft constraint S_8 (i.e. identical weekend).

The quality of each new roster $f(\boldsymbol{x}'_s)$ is calculated. If it is better than the quality of the old roster $f(\boldsymbol{x}_s)$, then it replaces the old roster in FSM. The detailed of this process is given in Algorithm 2.

STEP IV: *Send the Onlooker Bees to the Food Sources.* Subsequent to the completion exploitation of the employed bee phase, the employed bees share the information of exploited food sources (i.e. rosters) with onlooker bee. The onlooker bee decides to follow certain employed bee and exploits their corresponding food source randomly using the set of neighbourhood structures



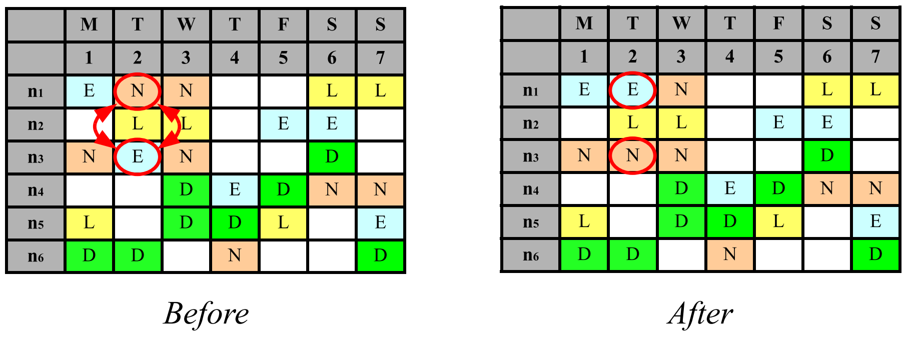
	M	T	W	T	F	S	S
	1	2	3	4	5	6	7
n1	E	E	N	E		L	L
n2		L	L		E	E	
n3	N	N	N			D	
n4			D		D	N	N
n5	L		D	D	L		E
n6	D	D		N			D

Before

	M	T	W	T	F	S	S
	1	2	3	4	5	6	7
n1	E	E	N			L	L
n2		L	L		E	E	
n3	N	N	N			D	
n4			D	E	D	N	N
n5	L		D	D	L		E
n6	D	D		N			D

After

Fig. 2. An example of MSN().



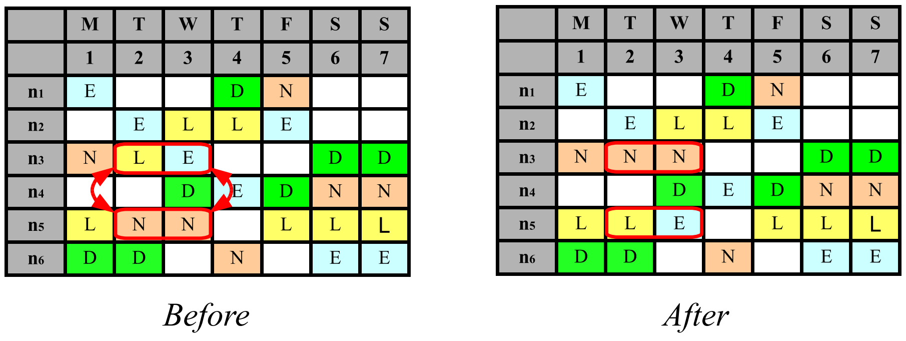
Before

	M	T	W	T	F	S	S
	1	2	3	4	5	6	7
n1	E	N	N			L	L
n2		L	L		E	E	
n3	N	E	N			D	
n4			D	E	D	N	N
n5	L		D	D	L		E
n6	D	D		N			D

After

	M	T	W	T	F	S	S
	1	2	3	4	5	6	7
n1	E	E	N			L	L
n2		L	L		E	E	
n3	N	N	N			D	
n4			D	E	D	N	N
n5	L		D	D	L		E
n6	D	D		N			D

Fig. 3. An example of SNS( ).



Before

	M	T	W	T	F	S	S
	1	2	3	4	5	6	7
n1	E			D	N		
n2		E	L	L	E		
n3	N	L	E			D	D
n4			D	E	D	N	N
n5	L	N	N		L	L	L
n6	D	D		N		E	E

After

	M	T	W	T	F	S	S
	1	2	3	4	5	6	7
n1	E			D	N		
n2		E	L	L	E		
n3	N	N	N			D	D
n4			D	E	D	N	N
n5	L	L	E		L	L	L
n6	D	D		N		E	E

Fig. 4. An example of SPS( ).

discussed above based on proportional selection probability as shown in Eq. (3)

$$p_i = \frac{f(\boldsymbol{x}_i)}{\sum_{k=1}^{SN} f(\boldsymbol{x}_k)} \tag{3}$$

Note that the $\sum_{i=1}^{SN} p_i$ is unity.

Thus, the roster with higher selection probability may be selected and adjusted to its neighbourhood using the same strategy as the employed bee. The quality of the new roster is calculated and if it is better, then it replaces the old one in FSM.

STEP V: *Send the Scout to Search for Possible New Food Sources.* Owing to continuous exploitation, some food sources may finally be exhausted in which they might be abandoned by its corresponding employed bee. Thus, the associated employed bee turns to a scout bee and explores the roster search space randomly for a possible new food source to replace the abandoned one. Memorize the quality of the best roster ($f(\boldsymbol{x}_{best}) = f(\boldsymbol{x}_1)$) found so far in FSM.

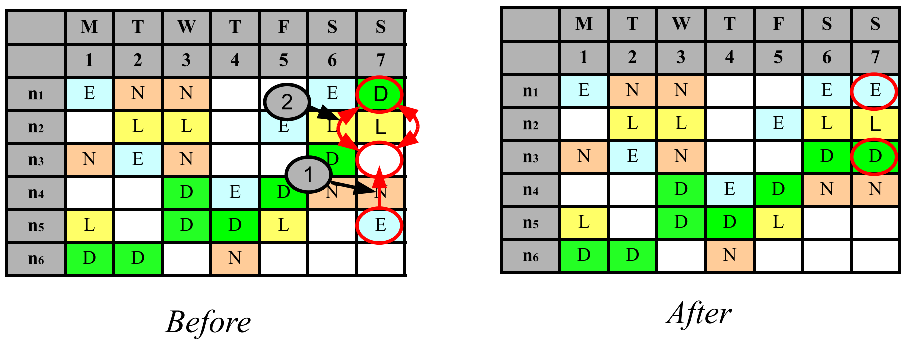
STEP IV: *Stopping condition.* Repeat steps III to V until a stop condition is achieved which is originally determined by *MCN* value.

4.2. Hybrid artificial bee colony for NRP

In this section, the proposed hybrid algorithm based on integration of HCO within the employed operator of the ABC to solve NRP is presented. Firstly, in next section, the description of HCO is briefly given, which is followed by its integration with the employed bee operator of the original ABC.

4.2.1. Hill climbing optimizer

Hill climbing optimizer is one of the simplest search techniques which belongs to the category of local search-based method. It is most intuitive type of search which evaluates a set of possible solutions in the neighbourhood associated with a move. The process of HCO begins with a random generated solution and its quality is evaluated. The generated solution is iteratively improved by making small changes and its quality is re-evaluated. The old solution is replaced with the new one if its has a better or equal quality. This process is repeated until a termination criterion is reached. In this paper, the proposed hybrid technique which based on integration of HCO within the employed bee operator of ABC takes initiative from the memetic algorithms that have been mostly



Before

	M	T	W	T	F	S	S
	1	2	3	4	5	6	7
n1	E	N	N			E	D
n2		L	L		E	L	L
n3	N	E	N			D	
n4			D	E	D	N	N
n5	L		D	D	L		E
n6	D	D		N			

After

	M	T	W	T	F	S	S
	1	2	3	4	5	6	7
n1	E	N	N			E	E
n2		L	L		E	L	L
n3	N	E	N			D	D
n4			D	E	D	N	N
n5	L		D	D	L		
n6	D	D		N			

Fig. 5. An example of TRM( ).

been employed in tackling numerous combinatorial optimization problems. Literature have shown that hybridization between local search-based methods like HCO with population-based methods such as ABC algorithm could be effective, though it may be computational expensive.

4.2.2. Hybridization ABC with hill climbing optimizer

In this paper, the hybridization of ABC algorithm with HCO is proposed. The integration involves replacing the process of the employed bee operator by HCO in order to optimize the rosters (i.e. food sources) locally to the maximum limit. The concept of this hybridization is adapted from our previous works on the ABC to the university timetabling problem [26] in which of the employed bee operator is replaced by the HCO in its operation to enhance the search intensification. Note that the feasibility of the solution search space is taken into consideration during the search process.

The HABC begins with a set of feasible rosters stored in FSM at initial stage (see STEP II of Algorithm 2). Iteratively, each employed bee utilizes the HCO to find the local optima of some rosters stored in FSM controlled by the hill climbing rater (*HCR*) parameter. The *HCR* with higher value leads to higher exploitation capability and thus the computational time increased. It is noteworthy that the HCO randomly utilize the four neighbourhood structures (i.e., MNS(), SNS(), SPS(), and TRM()) discussed in Section 4.1 to optimize the rosters in FSM. Algorithm 3 shows the new process of the employed bee operator in HABC, where this pseudo-code is replaced the old one in Algorithm 2. The other steps of the ABC are used as discussed in the original version of ABC presented in Section 4.1.

Algorithm 3. Employed bee operator of the HABC

```
1: for $s = 1$ to $SN$ for
2:    if ($U(0, 1) \le HCR$) then
3:       while (Local optima is not reached) do
4:          $r = rand(1, 4)$
5:          $\mathbf{x}'_s = N_r(\mathbf{x}_s)$   $N_1$ = MNS(), $N_2$ = SNS(), $N_3$ = SPS(), $N_4$ = TRM()
6:          if ($f(\mathbf{x}'_s) < f_s$) then
7:             $\mathbf{x}_s = \mathbf{x}'_s$
8:             $f_s = f(\mathbf{x}'_s)$
9:          end if
10:      end while
11:   end if
12: end for
```

5. Computational results and discussions

In this section, the performance of the proposed HABC for NRP is evaluated using the INRC2010 dataset released by Haspeslagh et al. [30] which is available at INRC2010 website.[1] A brief summary of the characteristics of the INRC2010 dataset is fully given in Table 2.

It is worthy to note that the proposed method is programmed using Microsoft Visual C++ version 6.0 under windows Vista. The experiments presented here ran on an Intel Machine with Core^{TM}processor 2.66 GHz, and 4 GB RAM.

5.1. Experimental design

In order to study the influence of hybridizing the HCO within the ABC in the proposed HABC, three varying values of the *HCR* parameter are used. Table 4 includes the parameter settings for the proposed HABC using three experimental cases (i.e., $Case_1$ to $Case_3$). It should be noted that the value of the *SN* parameter is fixed, where this value is adopted from Awadallah et al. [31]. Awadallah et al. [31] proposed other population-based metaheuristic method for NRP using the same INRC2010 dataset. Furthermore, the value

Table 4
The three experimental cases of the HABC.

Case	*SN*	*limit*	*MCN*	*HCR*
$Case_1$	10	100	INRC2010 time rules	0.1
$Case_2$	10	100	INRC2010 time rules	0.3
$Case_3$	10	100	INRC2010 time rules	0.5

of the second parameter *limit* is fixed, where this value is adopted from [26].

The value of the third parameter *MCN* is changeable according to the computational time rules of the INCR2010. During the search process, if the competition time rules are met, then the best results achieved are memorized, while the search process is stopped. It is worthy to note that the INRC2010 time rules are 10 sec. for spring track, 10 min for medium track, and 10 h for long track.

The HCO is employed within the ABC to fine-tune the search space region of the roster toward the local optima by enhancing its local exploitation capability. The experimental case that achieved the best results among the three $Case_1$ to $Case_3$, is further studied where the HCO is not triggered in order to show the effectiveness of the hybridizing HCO within the employed bee operator of the ABC in $Case_4$. In this case, the value of the *HCR* parameter is zero, where the other parameter settings are similar to the best scenario among $Case_1$ to $Case_3$.

5.2. Experimental results

The experimental results of the four cases are summarized in Tables 5 and 6, where all these results are achieved within the competition time rules. It is noteworthy the numbers in these tables represent the penalty values of 10 runs (lowest is best), which are computed by the objective function (i.e., see Eq. (1)). For each problem instance on each experimented case, the best results, the mean, and the standard deviation (std.) over 10 runs are recorded. The best results on each problem instance is highlighted in bold.

It is apparent from Table 5 that the performance of HABC is improved as the value of *HCR* increases. Experimentally, $Case_1$ obtained the best results in 13 out of 69 instances, $Case_2$ achieved the best results in 19 out of 69 instances, while $Case_3$ obtained the best results in 66 out of 69 instances.

It can be noted that the stopping condition for the three cases of the HABC is the time rules of the INRC2010. Thus leads to fairness among the three cases [45,46], where the higher value of *HCR*, leading to higher number of calling the HCO at the higher number of iterations during the search and thus leading to a high rate of exploitation and computational time.

It is worthy of mentioning that when the value of *HCR* is bigger than 0.50, then the HABC achieved almost all the same results of the three cases (i.e., $Case_1$ to $Case_3$). Based on the HABC experimental results, the proposed HABC has high capability of exploiting the solution search space in different ways, thus guided the search towards better results.

In order to show the effectiveness of the hybridizing HCO within ABC, we study the ABC without triggering the HCO in $Case_4$. It should be noted that $Case_3$ and $Case_4$ having the same parameter setting with one difference in *HCR* parameter, where *HCR* is zero in $Case_4$ and *HCR* is 0.50 in $Case_3$. Experimentally, the results achieved by $Case_4$ are the worst in comparison with $Case_3$, where it obtained the best results in all instances of INRC2010 dataset.

Fig. 6 plots the best results in the population at each iteration for the proposed HABC using different *HCR* values (i.e. $Case_1$ (*HCR*=0.10), $Case_2$ (*HCR*=0.30), $Case_3$ (*HCR*=0.50) and $Case_4$ (*HCR*=0)) when exploring the search space for *Long*01 instance. The selected run is chosen randomly from the ten runs experimented for each cases. The colored lines in this plot shows the correlation between

[1] https://www.kuleuven-kulak.be/nrpcompetition.

Table 5
Effect of various *HCR* values on behavior of HABC using INRC2010 dataset.

Problem instance	*Case*$_1$			*Case*$_2$			*Case*$_3$		
	Best	Mean	Standard deviation	Best	Mean	Standard deviation	Best	Mean	Standard deviation
*Sprint_Early*01	57	57.6	0.5	**56**	57.4	1.0	**56**	56.8	1.0
*Sprint_Early*02	**58**	59	0.8	59	59.1	0.3	**58**	58.8	0.6
*Sprint_Early*03	52	53.5	1.4	52	53	0.9	**51**	52	0.9
*Sprint_Early*04	62	62.5	0.7	61	62	1.1	**59**	60.6	1.7
*Sprint_Early*05	59	59.3	0.9	**58**	58.4	0.5	**58**	58.9	0.9
*Sprint_Early*06	**54**	55.3	1.1	55	55.6	0.5	**54**	54.8	0.8
*Sprint_Early*07	58	58.6	0.7	57	58.5	1.4	**56**	58.8	1.9
*Sprint_Early*08	**56**	57	1.2	**56**	56.9	0.7	**56**	56.8	0.6
*Sprint_Early*09	**55**	56.9	1.4	**55**	56.7	1.4	**55**	55.8	0.8
*Sprint_Early*10	53	53.6	1.0	53	54.5	1.4	**52**	52.9	0.9
*Sprint_Hidden*01	36	37.5	1.2	35	38.9	3.6	**32**	35	2.0
*Sprint_Hidden*02	**32**	37.1	3.8	34	36	1.8	**32**	35	2.5
*Sprint_Hidden*03	64	70.8	4.8	66	69.1	3.3	**62**	65.1	2.9
*Sprint_Hidden*04	68	71.8	2.9	69	71.1	1.2	**66**	69.4	2.9
*Sprint_Hidden*05	60	62.1	1.4	60	61.6	1.3	**59**	60.1	0.9
*Sprint_Hidden*06	135	154.3	18.4	134	151.8	18.7	**130**	147.8	17.7
*Sprint_Hidden*07	161	176.7	11.9	160	173.9	15.2	**153**	170.7	12.0
*Sprint_Hidden*08	206	224.8	11.7	214	223.6	9.6	**204**	227.4	13.0
*Sprint_Hidden*09	**338**	354	12.2	343	347	6.9	**338**	353	12.6
*Sprint_Hidden*10	**306**	327.6	19.8	310	323.8	9.5	**306**	307.6	3.4
*Sprint_Late*01	40	43.5	4.1	41	42.3	1.3	**37**	41.4	2.9
*Sprint_Late*02	44	47.5	3.0	44	46.4	2.4	**42**	44.4	2.1
*Sprint_Late*03	52	52.4	0.7	50	52.6	1.6	**48**	52.1	2.3
*Sprint_Late*04	82	85.6	2.8	**73**	83.6	7.9	**73**	77.9	3.9
*Sprint_Late*05	46	47.6	1.3	46	48.7	2.2	**44**	46.1	1.9
*Sprint_Late*06	43	45.8	2.3	**42**	43.8	1.3	**42**	43.9	1.3
*Sprint_Late*07	**44**	48.9	6.0	47	49.7	2.7	46	51.1	3.6
*Sprint_Late*08	**17**	21.3	4.9	**17**	18.8	3.3	**17**	18.6	1.9
*Sprint_Late*09	**17**	23.9	5.6	**17**	18.4	1.9	**17**	18.8	2.3
*Sprint_Late*10	45	49.9	5.0	46	49.6	3.7	**43**	44.4	1.6
*Sprint_Hint*01	76	83.8	5.2	**75**	83	6.4	**75**	79.1	3.7
*Sprint_Hint*02	49	54.1	4.1	51	54.7	4.2	**46**	47.9	2.5
*Sprint_Hint*03	53	61.2	5.6	**50**	58.1	9.0	**50**	57.9	7.5
*Medium_Early*01	247	248.8	1.8	**245**	248	2.4	**245**	246.7	2.5
*Medium_Early*02	246	248.7	2.5	246	247.6	1.6	**245**	247	2.6
*Medium_Early*03	245	246.6	1.6	**242**	245	1.5	**242**	244.1	1.6
*Medium_Early*04	244	245.9	1.8	244	245.6	1.2	**240**	243.8	2.7
*Medium_Early*05	311	311.4	0.5	309	311.2	2.1	**308**	309.8	2.0
*Medium_Hidden*01	173	183.9	10.0	**155**	179.9	17.1	158	173.9	17.3
*Medium_Hidden*02	273	290.3	16.4	**254**	278.2	21.8	**254**	268.8	9.2
*Medium_Hidden*03	58	64.1	7.1	**54**	61.9	6.4	**54**	60.4	5.2
*Medium_Hidden*04	101	102.5	1.8	96	105.5	7.1	**94**	97.6	3.2
*Medium_Hidden*05	193	208.8	16.7	185	192.9	11.8	**177**	196.8	13.3
*Medium_Late*01	183	194	9.5	**174**	186.5	9.7	**174**	181.6	7.6
*Medium_Late*02	35	40.7	4.5	36	40.6	3.5	**31**	36.4	6.0
*Medium_Late*03	44	46.5	4.2	**38**	44.8	6.5	**38**	43	4.4
*Medium_Late*04	**48**	51.5	4.2	52	54.7	2.4	**48**	53.5	3.3
*Medium_Late*05	148	164.1	15.1	143	149.4	6.7	**134**	141.6	5.0
*Medium_Hint*01	**48**	56.7	6.1	52	57.6	4.2	52	56.3	6.1
*Medium_Hint*02	**94**	115.9	14.2	104	108.8	6.8	**94**	104	8.8
*Medium_Hint*03	149	162	7.6	146	157.6	9.9	**140**	150.8	9.6
*Long_Early*01	202	204.1	2.5	203	207.6	3.2	**197**	203.2	3.9
*Long_Early*02	235	238.6	3.3	235	237.2	1.8	**229**	233.8	2.8
*Long_Early*03	241	242.4	1.3	**240**	241.8	1.9	**240**	240.5	1.6
*Long_Early*04	309	312.3	1.8	308	310.3	2.2	**303**	307	2.3
*Long_Early*05	290	291.7	2.1	288	291.2	2.6	**284**	287.9	2.3
*Long_Hidden*01	432	450	21.5	422	435.7	9.2	**400**	428.7	12.9
*Long_Hidden*02	119	131.4	7.6	124	128	4.8	**117**	126.6	6.2
*Long_Hidden*03	58	62.1	3.4	**51**	59.5	7.0	**51**	57.7	4.4
*Long_Hidden*04	42	48.9	6.2	38	46.6	6.7	**29**	40.7	8.4
*Long_Hidden*05	71	78	6.4	66	74.9	6.4	**56**	68.7	11.5
*Long_Late*01	269	285.8	12.8	263	282	14.4	**257**	264.1	7.3
*Long_Late*02	266	290.7	14.4	269	274.1	4.4	**263**	271.4	7.3
*Long_Late*03	270	290.4	16.0	269	279.7	9.2	**262**	276.5	14.5
*Long_Late*04	291	302.8	9.5	262	281.6	15.0	**261**	269.2	7.9
*Long_Late*05	121	133.8	9.6	122	129.5	10.5	**102**	114.9	11.4
*Long_Hint*01	48	51.5	3.2	46	53.4	6.7	**42**	46.8	3.4
*Long_Hint*02	37	40.5	3.4	33	36.6	3.4	**30**	35.4	5.8
*Long_Hint*03	104	120.1	13.0	100	122	18.7	**83**	104.5	15.7

Table 6
Effect of HCO on the behavior of ABC using INRC2010 dataset.

Problem instance	ABC			HABC
	Best	Mean	Standard deviation	Best
*Sprint _Early*01	62	63.9	1.9	**56**
*Sprint _Early*02	64	65.3	1.1	**58**
*Sprint _Early*03	58	61.6	3.9	**51**
*Sprint _Early*04	66	67.9	1.8	**59**
*Sprint _Early*05	63	63.6	0.5	**58**
*Sprint _Early*06	58	59.7	2.0	**54**
*Sprint _Early*07	61	62	0.8	**56**
*Sprint _Early*08	58	62	3.7	**56**
*Sprint _Early*09	58	60.4	2.2	**55**
*Sprint _Early*10	57	59.7	2.2	**52**
*Sprint _Hidden*01	44	46.2	1.3	**32**
*Sprint _Hidden*02	38	42.8	5.0	**32**
*Sprint _Hidden*03	71	74	3.1	**62**
*Sprint _Hidden*04	76	77.6	1.8	**66**
*Sprint _Hidden*05	65	68.4	3.0	**59**
*Sprint _Hidden*06	161	182.6	15.8	**130**
*Sprint _Hidden*07	178	193	18.1	**153**
*Sprint _Hidden*08	245	252.3	7.5	**204**
*Sprint _Hidden*09	371	379.6	9.3	**338**
*Sprint _Hidden*10	327	344.7	19.1	**306**
*Sprint _Late*01	49	51.9	3.2	**37**
*Sprint _Late*02	52	53.5	1.6	**42**
*Sprint _Late*03	56	58.5	3.0	**48**
*Sprint _Late*04	89	100.5	6.6	**73**
*Sprint _Late*05	53	53.8	1.0	**44**
*Sprint _Late*06	47	48.1	1.5	**42**
*Sprint _Late*07	52	57.8	5.5	**44**
*Sprint _Late*08	**17**	23.8	8.3	**17**
*Sprint _Late*09	**17**	26.2	5.4	**17**
*Sprint _Late*10	56	57.5	1.6	**43**
*Sprint _Hint*01	85	93.5	6.6	**75**
*Sprint _Hint*02	57	59.9	2.6	**46**
*Sprint _Hint*03	74	77.6	7.1	**50**
*Medium _Early*01	260	266.9	5.3	**245**
*Medium _Early*02	261	267	4.8	**245**
*Medium _Early*03	259	267.4	6.2	**242**
*Medium _Early*04	257	264.7	8.6	**240**
*Medium _Early*05	329	333.6	6.5	**308**
*Medium _Hidden*01	188	200.7	7.7	**158**
*Medium _Hidden*02	284	298.1	10.9	**254**
*Medium _Hidden*03	64	68	3.3	**54**
*Medium _Hidden*04	100	105.4	4.1	**94**
*Medium _Hidden*05	201	211.1	11.6	**177**
*Medium _Late*01	206	223.4	11.6	**174**
*Medium _Late*02	52	54.3	2.9	**31**
*Medium _Late*03	70	72.6	2.1	**38**
*Medium _Late*04	65	70.8	4.3	**48**
*Medium _Late*05	178	192	11.9	**134**
*Medium _Hint*01	69	77.2	7.9	**48**
*Medium _Hint*02	141	151.3	10.8	**94**
*Medium _Hint*03	187	225.6	42.5	**140**
*Long _Early*01	242	247.2	3.8	**197**
*Long _Early*02	277	284.1	5.2	**229**
*Long _Early*03	269	277.4	5.0	**240**
*Long _Early*04	337	346.6	8.3	**303**
*Long _Early*05	327	332.1	5.3	**284**
*Long _Hidden*01	445	457	11.2	**400**
*Long _Hidden*02	130	132.4	2.1	**117**
*Long _Hidden*03	59	62.4	3.5	**51**
*Long _Hidden*04	47	50.9	2.8	**29**
*Long _Hidden*05	76	81.5	4.0	**56**
*Long _Late*01	288	296.9	6.5	**257**
*Long _Late*02	293	311.5	30.7	**263**
*Long _Late*03	306	311.1	5.8	**262**
*Long _Late*04	303	313.7	9.6	**261**
*Long _Late*05	141	151.5	9.6	**102**
*Long _Hint*01	52	56.1	2.4	**42**
*Long _Hint*02	39	44.7	8.8	**30**
*Long _Hint*03	116	122.6	7.4	**83**

Table 7
Wilcoxon signed-rank statistical test used for every pair of experimental cases of HABC on INRC2010 dataset.

Problem instance	$Case_1$–$Case_2$		$Case_1$–$Case_3$		$Case_1$–$Case_4$		$Case_2$–$Case_3$		$Case_2$–$Case_4$		$Case_3$–$Case_4$	
	Dif	ρ	*Dif*	ρ	*Dif*	ρ	*Dif*	ρ	*Dif*	ρ	*Dif*	ρ
Sprint_early01	−0.5916	0.5541	−1.8204	0.0687	−2.8031	0.0051*	−1.4703	0.1415	−2.8031	0.0051*	−2.8031	0.0051*
Sprint_early02	−0.3381	0.7353	−0.7338	0.4631	−2.8031	0.0051*	−1.2136	0.2249	−2.8031	0.0051*	−2.8031	0.0051*
Sprint_early03	−1.0483	0.2945	−2.5205	0.0117	−2.8031	0.0051*	−2.2014	0.0277	−2.8031	0.0051*	−2.8031	0.0051*
Sprint_early04	−1.4003	0.1614	−2.3664	0.0180	−2.8031	0.0051*	−2.3694	0.0178*	−2.8031	0.0051*	−2.8031	0.0051*
Sprint_early05	−2.2014	0.0277	−0.8402	0.4008	−2.8031	0.0051*	−1.3628	0.1730	−2.8031	0.0051*	−2.8031	0.0051*
Sprint_early06	−0.9435	0.3454	−1.3522	0.1763	−2.8031	0.0051*	−2.0226	0.0431	−2.8031	0.0051*	−2.8031	0.0051*
Sprint_early07	−0.3058	0.7598	−0.3501	0.7263	−2.8031	0.0051*	−0.2962	0.7671	−2.8031	0.0051*	−2.6656	0.0077*
Sprint_early08	−0.1690	0.8658	−0.6290	0.5294	−2.6502	0.0080*	−0.3381	0.7353	−2.5205	0.0117*	−2.8031	0.0051*
Sprint_early09	−0.3381	0.7353	−1.8363	0.0663	−2.5205	0.0117*	−1.9439	0.0519	−2.6502	0.0080*	−2.8031	0.0051*
Sprint_early10	−1.5213	0.1282	−1.2603	0.2076	−2.8031	0.0051*	−2.2404	0.0251*	−2.8031	0.0051*	−2.8031	0.0051*
Sprint_hidden01	−1.1902	0.2340	−2.2404	0.0251	−2.8031	0.0051*	−2.1325	0.0330*	−2.6656	0.0077*	−2.8031	0.0051*
Sprint_hidden02	−0.6625	0.5076	−1.4703	0.1415	−2.2424	0.0249*	−1.1202	0.2626	−2.8031	0.0051*	−2.8031	0.0051*
Sprint_hidden03	−0.8664	0.3863	−2.3102	0.0209	−1.4780	0.1394	−2.1915	0.0284*	−2.1405	0.0323*	−2.8031	0.0051*
Sprint_hidden04	−0.5331	0.5940	−1.6586	0.0972	−2.6656	0.0077*	−1.4780	0.1394	−2.8031	0.0051*	−2.8031	0.0051*
Sprint_hidden05	−0.9102	0.3627	−2.5205	0.0117	−2.8031	0.0051*	−2.3102	0.0209*	−2.8031	0.0051*	−2.8031	0.0051*
Sprint_hidden06	−0.1529	0.8785	−1.1722	0.2411	−2.4973	0.0125*	−0.5606	0.5751	−2.7011	0.0069*	−2.4973	0.0125*
Sprint_hidden07	−0.4587	0.6465	−0.8885	0.3743	−2.6502	0.0080*	−0.1529	0.8785	−1.9876	0.0469	−2.1004	0.0357*
Sprint_hidden08	−0.1529	0.8785	−0.3554	0.7223	−2.8031	0.0051*	−0.5096	0.6103	−2.8031	0.0051*	−2.8031	0.0051*
Sprint_hidden09	−1.6309	0.1029	−0.1777	0.8590	−2.7011	0.0069*	−1.1722	0.2411	−2.8031	0.0051*	−2.8031	0.0051*
Sprint_hidden10	−0.2548	0.7989	−2.2509	0.0244*	−1.5799	0.1141	−2.8031	0.0051*	−2.3102	0.0209*	−2.8031	0.0051*
Sprint_late01	−0.6516	0.5147	−1.2439	0.2135	−2.8031	0.0051*	−1.1902	0.2340	−2.8031	0.0051*	−2.8031	0.0051*
Sprint_late02	−0.9683	0.3329	−2.1974	0.0280*	−2.6656	0.0077*	−1.9876	0.0469	−2.8031	0.0051*	−2.8031	0.0051*
Sprint_late03	−0.2962	0.7671	−0.0592	0.9528	−2.8031	0.0051*	−0.4201	0.6744	−2.8031	0.0051*	−2.8031	0.0051*
Sprint_late04	−1.0070	0.3139	−2.8031	0.0051*	−2.6656	0.0077*	−2.3102	0.0209*	−2.8031	0.0051*	−2.8031	0.0051*
Sprint_late05	−1.4809	0.1386	−2.1405	0.0323*	−2.8031	0.0051*	−2.3694	0.0178*	−2.8031	0.0051*	−2.8031	0.0051*
Sprint_late06	−1.6803	0.0929	−1.7504	0.0801	−1.8955	0.0580	−0.2097	0.8339	−2.8031	0.0051*	−2.8031	0.0051*
Sprint_late07	−0.3568	0.7213	−1.3251	0.1851	−2.8031	0.0051*	−0.7135	0.4755	−2.6656	0.0077*	−2.5205	0.0117*
Sprint_late08	−1.3522	0.1763	−1.9604	0.0500	−0.2962	0.7671	−0.1400	0.8886	−1.5993	0.1097	−2.1004	0.0357*
Sprint_late09	−2.1915	0.0284*	−2.3664	0.0180*	−0.5606	0.5751	−0.5601	0.5754	−2.6656	0.0077*	−2.3105	0.0209*
Sprint_late10	−0.5096	0.6103	−2.6656	0.0077*	−2.6502	0.0080*	−2.6502	0.0080*	−2.6656	0.0077*	−2.8031	0.0051*
Sprint_hint01	−0.4077	0.6835	−1.6818	0.0926	−2.2934	0.0218*	−1.2439	0.2135	−2.1915	0.0284*	−2.8031	0.0051*
Sprint_hint02	−0.5331	0.5940	−2.5992	0.0093*	−2.6502	0.0080*	−2.6502	0.0080*	−2.4286	0.0152*	−2.8031	0.0051*
Sprint_hint03	−1.1722	0.2411	−0.9174	0.3590	−2.8031	0.0051*	−0.0700	0.9442	−2.7011	0.0069*	−2.8031	0.0051*
Medium_early01	−0.8154	0.4122	−1.8347	0.0672	−2.8031	0.0051*	−1.2603	0.2076	−2.8031	0.0051*	−2.8031	0.0051*
Medium_early02	−1.1255	0.2604	−1.2439	0.2135	−2.8031	0.0051*	−0.7001	0.4838	−2.8031	0.0051*	−2.8031	0.0051*
Medium_early03	−2.3664	0.0180*	−2.6656	0.0077*	−2.8031	0.0051*	−1.1202	0.2626	−2.8031	0.0051*	−2.8031	0.0051*
Medium_early04	−0.4201	0.6744	−1.4809	0.1386	−2.8031	0.0051*	−1.8204	0.0687	−2.8031	0.0051*	−2.8031	0.0051*
Medium_early05	−0.6516	0.5147	−2.0140	0.0440*	−2.8031	0.0051*	−1.3032	0.1925	−2.8031	0.0051*	−2.8031	0.0051*
Medium_hidden01	−0.4587	0.6455	−1.4270	0.1527	−2.5992	0.0093*	−0.7645	0.4446	−2.3953	0.0166*	−2.4973	0.0125*
Medium_hidden02	−1.4780	0.1389	−2.6502	0.0080*	−1.1722	0.2420	−0.8664	0.3863	−1.9876	0.0469*	−2.8031	0.0051*
Medium_hidden03	−1.3624	0.1731	−1.0142	0.3105	−1.5799	0.1141	−0.3568	0.7213	−2.1325	0.0330*	−2.8031	0.0051*
Medium_hidden04	−1.2439	0.2135	−2.4879	0.0129*	−1.7838	0.0751	−2.4463	0.0144*	−0.4077	0.6835	−2.1915	0.0284*
Medium_hidden05	−2.0896	0.0366*	−1.9367	0.0524	−1.0703	0.2846	−0.6625	0.5076	−2.7011	0.0069*	−2.8031	0.0051*
Medium_late01	−1.5289	0.1260	−1.9548	0.0506	−2.8031	0.0051*	−1.0703	0.2845	−2.8031	0.0051*	−2.8031	0.0051*
Medium_late02	−0.4201	0.6744	−1.8347	0.0665	−2.8031	0.0051*	−1.6586	0.0972	−2.8031	0.0051*	−2.8031	0.0051*
Medium_late03	−0.7606	0.4469	−1.6309	0.1029	−2.8031	0.0051*	−0.3568	0.7213	−2.8031	0.0051*	−2.8031	0.0051*
Medium_late04	−1.5799	0.1141	−1.1902	0.2340	−2.8031	0.0051*	−0.7645	0.4446	−2.8031	0.0051*	−2.8031	0.0051*
Medium_late05	−2.3102	0.0209*	−2.8031	0.0051*	−2.8031	0.0051*	−2.2424	0.0249	−2.8031	0.0051*	−2.8031	0.0051*
Medium_hint01	−0.4587	0.6465	−0.4146	0.6784	−2.8031	0.0051*	−0.7108	0.4772	−2.8031	0.0051*	−2.8031	0.0051*
Medium_hint02	−1.2232	0.2213	−1.9548	0.0506	−2.8031	0.0051*	−0.9683	0.3329	−2.8031	0.0051*	−2.8031	0.0051*
Medium_hint03	−1.1722	0.2411	−1.2232	0.2213	−2.8031	0.0051*	−0.2548	0.7989	−2.8031	0.0051*	−2.8031	0.0051*
Long_early01	−2.0386	0.0415*	−0.6625	0.5076	−2.8031	0.0051*	−2.2424	0.0249*	−2.8031	0.0051*	−2.8031	0.0051*
Long_early02	−0.9478	0.3433	2.5992	0.0093*	−2.8031	0.0051*	−2.8031	0.0051*	−2.8031	0.0051*	−2.8031	0.0051*
Long_early03	−1.2439	0.2135	−2.6656	0.0077*	−2.8031	0.0051*	−1.5213	0.1282	−2.8031	0.0051*	−2.8031	0.0051*
Long_early04	−1.9876	0.0469	−2.8031	0.0051*	−2.8031	0.0051*	−2.5205	0.0117*	−2.8031	0.0051*	−2.8031	0.0051*
Long_early05	−0.4146	0.6784	−2.7011	0.0069*	−2.8031	0.0051*	−1.7838	0.0745	−2.8031	0.0051*	−2.8031	0.0051*
Long_hidden01	−1.5993	0.1097	−2.7011	0.0069*	−0.8154	0.4148	−1.4809	0.1386	−2.7011	0.0069*	−2.6656	0.0077*
Long_hidden02	−1.1847	0.2361	−1.9367	0.0528	−0.4146	0.6784	−0.4077	0.6835	−2.1325	0.0330*	−2.0386	0.0415*
Long_hidden03	−1.3624	0.1731	−2.0140	0.0440*	−0.2535	0.7998	−0.5331	0.5940	−1.1722	0.2411	−1.9876	0.0469
Long_hidden04	−0.4146	0.6784	−1.7838	0.0745*	−1.0703	0.2845	−1.9548	0.0506	−1.2603	0.2076	−2.4463	0.0144*
Long_hidden05	−0.5096	0.6103	−2.0896	0.0367*	−1.2232	0.2213	−1.5289	0.1263	−2.4973	0.0125*	−2.6502	0.0080*
Long_late01	−0.2548	0.7989	−2.8031	0.0051*	−2.0386	0.0415*	−2.8031	0.0051*	−2.3953	0.0166*	−2.8031	0.0051*
Long_late02	−2.4973	0.0125*	−2.5992	0.0093*	−2.0732	0.0382*	−1.0142	0.3105	−2.8031	0.0051*	−2.8031	0.0051*
Long_late03	−2.2424	0.0249*	−1.9367	0.0528	−2.4973	0.0125*	−0.4077	0.6835	−2.8031	0.0051*	−2.8031	0.0051*
Long_late04	−2.4973	0.0125*	−2.8031	0.0051*	−1.8363	0.0663	−1.9548	0.0506	−2.8031	0.0051*	−2.8031	0.0051*
Long_late05	−1.1722	0.2411	−2.3953	0.0166*	−2.8031	0.0051*	−2.2934	0.0218*	−2.8031	0.0051*	−2.8031	0.0051*
Long_hint01	−0.4146	0.6784	−2.6502	0.0080*	−2.5992	0.0093*	−2.2404	0.0251*	−1.2741	0.2026	−2.8031	0.0051*
Long_hint02	−2.2509	0.0244*	−1.9876	0.0469	−1.1847	0.2361	−0.5096	0.6103	−2.8031	0.0051*	−2.4286	0.0152*
Long_hint03	−0.0510	0.9594	−1.7178	0.0858	−0.3568	0.7213	−1.7838	0.0745	−0.2039	0.8385	−2.8031	0.0051*

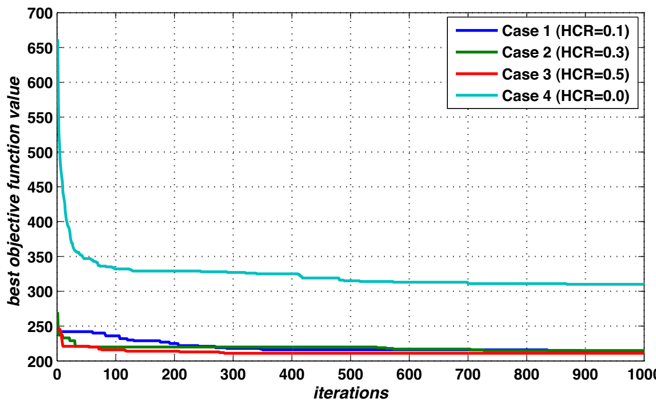


Fig. 6. Comparison between different *HCR* values using long01 instance.

the number of iterations and the objective function value (i.e., roster quality). An analysis of this Figure shows that the roster quality improves as the number of iteration increases. The slope of the curves is relatively steep in the beginning of the search, which indicates a great improvement in the quality of rosters for *Long*01 instance where there is possibly much scope for improvement. The degree of improvement becomes relatively slower as the number of generations increases. Notably, the steepest slope curve shows the scenario that achieved the best results. Clearly, $Case_3$ has the best rate of convergence, due to the high value of *HCR*, and this leads to high speed of convergence.

Wilcoxon signed-rank statistical test is used to detect the significant difference between the different experimental cases (i.e., $Case_1$ to $Case_4$) of HABC. Table 7 shows the results of the Wilcoxon signed-rank test for every pair of the experimental cases. This table includes the hypothesized value (i.e., *Dif*), and the ρ-value. The indicator '*' shows that there is a significant difference among the two experimental cases when solving a particular problem instance.

As shown in Table 7, there are significant difference among the three cases of HABC (i.e., $Case_1$ to $Case_3$) and the fourth experimental case (i.e., $Case_4$) for almost all problem instances. It should be noted that $Case_1$ to $Case_3$ the HCO is integrated within the HABC, where the HCO is not integrated within HABC in $Case_4$ and therefore all cases produce significant results against $Case_4$. As can be noted, the higher rate of *HCR* is, the significant the results will be.

It is worth mentioning that as shown in Table 7, the comparison between the results of the $Case_1$ and $Case_2$ shows there is a significant difference in 9 out of 69 instances in favor of $Case_2$. Furthermore, the comparison between the results of the $Case_1$ and $Case_3$ shows there is significant difference in 25 out of 69 instances in favor of $Case_3$. Finally, there is significant difference among $Case_2$ and $Case_3$ in 17 out of 69 instances in favor of $Case_3$.

Table 8
Key to the comparative methods.

Key	Method	Reference
ABC	Artificial bee colony	Our approach
HABC	Hybrid artificial bee colony	Our approach
M_1	Branch and price algorithm	[35]
M_2	Constraint optimization problem solver	[36]
M_3	Global-best harmony search with new pitch adjustment design	[31]
M_4	Harmony search hyper-heuristic algorithm	[41]
M_5	Harmony search with gready shuffle move	[47]
M_6	Hybrid harmony search algorithm	[40]
M_7	Hyper-heuristic combined with a greedy shuffle approach	[34]
M_8	Integer programming with set of neighbourhood structures	[33]
M_9	Integer programming	[39]
M_{10}	Tabu search with restart mechanism	[38]
M_{11}	Variable Depth Search Algorithm	[35]

5.3. Comparison with previous works

The best results achieved by the proposed HABC (i.e., see Table 5) are compared with those obtained by the ABC (i.e., see Table 6) and all previous methods using the same INRC2010 dataset. These include a total of thirteen comparative methods as summarized in Table 8.

As shown in Table 9, the results obtained by HABC seems to be competitive with those achieved by other previous methods, while the performance of the ABC is comparable with some of the previous methods. Note that the numbers in this table refer to the quality of the roster calculated by Eq. (1), where the lowest results are the best quality. The '-' indicator shows where the method did not run for these problem instances. The number highlighted in bold font marks the best results.

It is apparent from Table 9, the HABC achieved two new best results for two problem instances (i.e. *Sprint_hint*03 and *Medium_hint*03), and 35 best published results out of 69 instances

Table 9
Summary of the best results achieved by the proposed methods and other comparative methods on INRC2010 dataset.

Problem instance	ABC	HABC	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}
*Sprint_early*01	62	**56**	**56**	**56**	58	58	**56**	**56**	57	**56**	**56**	**56**	**56**
*Sprint_early*02	64	**58**	**58**	**58**	60	60	62	**58**	59	**58**	**58**	**58**	**58**
*Sprint_early*03	58	**51**	**51**	**51**	53	53	57	**51**	**51**	**51**	**51**	**51**	**51**
*Sprint_early*04	66	**59**	**59**	**59**	62	62	65	**59**	60	**59**	**59**	**59**	**59**
*Sprint_early*05	63	**58**	**58**	**58**	59	**58**	61	**58**	**58**	**58**	**58**	**58**	**58**
*Sprint_early*06	58	**54**	**54**	**54**	56	55	56	**54**	**54**	**54**	**54**	**54**	**54**
*Sprint_early*07	61	**56**	**56**	**56**	58	58	59	**56**	**56**	**56**	**56**	**56**	**56**
*Sprint_early*08	58	**56**	**56**	**56**	57	**56**	**56**	**56**	**56**	**56**	**56**	**56**	**56**
*Sprint_early*09	58	**55**	**55**	**55**	57	57	59	**55**	**55**	**55**	**55**	**55**	**55**
*Sprint_early*10	57	**52**	**52**	**52**	53	54	54	**52**	**52**	**52**	**52**	**52**	**52**
*Sprint_hidden*01	44	**32**	–	–	41	–	42	**32**	–	33	**32**	**32**	–
*Sprint_hidden*02	38	**32**	–	–	35	–	40	**32**	–	**32**	**32**	**32**	–
*Sprint_hidden*03	71	**62**	–	–	70	–	74	**62**	–	**62**	**62**	**62**	–
*Sprint_hidden*04	76	**66**	–	–	79	–	74	**66**	–	67	**66**	**66**	–
*Sprint_hidden*05	65	**59**	–	–	62	–	65	**59**	–	**59**	**59**	**59**	–
*Sprint_hidden*06	161	**130**	–	–	202	–	165	**130**	–	134	**130**	**130**	–
*Sprint_hidden*07	178	**153**	–	–	196	–	183	**153**	–	**153**	**153**	**153**	–
*Sprint_hidden*08	245	**204**	–	–	266	–	236	**204**	–	209	**204**	**204**	–
*Sprint_hidden*09	371	**338**	–	–	273	–	367	**338**	–	**338**	**338**	**338**	–

Table 9 (*Continued*)

Problem instance	ABC	HABC	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}
Sprint_hidden10	327	**306**	–	–	346	–	317	**306**	–	**306**	**306**	**306**	–
Sprint_late01	49	**37**	**37**	**37**	45	–	47	**37**	40	**37**	**37**	**37**	**37**
Sprint_late02	52	**42**	**42**	**42**	49	–	51	**42**	44	**42**	**42**	**42**	**42**
Sprint_late03	56	**48**	**48**	**48**	55	–	55	**48**	50	**48**	**48**	**48**	**48**
Sprint_late04	89	**73**	**73**	76	104	–	90	**73**	81	75	**73**	**73**	75
Sprint_late05	53	**44**	**44**	45	51	–	52	45	45	**44**	**44**	**44**	**44**
Sprint_late06	47	**42**	**42**	**42**	43	–	47	**42**	**42**	**42**	**42**	**42**	**42**
Sprint_late07	52	44	**42**	43	60	–	50	43	46	**42**	**42**	**42**	**42**
Sprint_late08	**17**	**17**	**17**	**17**	**17**	–	**17**	**17**	**17**	**17**	**17**	**17**	**17**
Sprint_late09	**17**	**17**	**17**	**17**	**17**	–	**17**	**17**	**17**	**17**	**17**	**17**	**17**
Sprint_late10	56	**43**	**43**	44	54	–	52	**43**	46	**43**	**43**	**43**	**43**
Sprint_hint01	85	**75**	–	–	101	–	88	**75**	78	–	–	–	–
Sprint_hint02	57	46	–	–	59	–	54	**43**	47	–	–	–	–
Sprint_hint03	74	**50**	–	–	77	–	67	51	57	–	–	–	–
Medium_early01	260	245	**240**	241	270	249	274	248	242	**240**	**240**	**240**	244
Medium_early02	261	245	**240**	**240**	275	251	275	248	241	**240**	**240**	**240**	241
Medium_early03	259	242	**236**	**236**	265	247	281	243	238	**236**	**236**	**236**	238
Medium_early04	257	240	**237**	238	263	248	272	245	238	**237**	**237**	**237**	240
Medium_early05	329	308	**303**	304	334	315	324	311	304	**303**	**303**	**303**	308
Medium_hidden01	188	155	–	–	253	–	404	190	–	130	**111**	117	–
Medium_hidden02	284	254	–	–	361	–	406	264	–	221	221	**220**	–
Medium_hidden03	64	54	–	–	93	–	176	59	–	36	**34**	35	–
Medium_hidden04	100	94	–	–	135	–	162	96	–	81	**78**	79	–
Medium_hidden05	201	177	–	–	275	–	517	190	–	122	**119**	**119**	–
Medium_late01	206	174	**157**	176	254	–	231	175	163	158	**157**	164	187
Medium_late02	52	31	**18**	19	72	–	46	31	21	**18**	**18**	20	22
Medium_late03	70	38	**29**	30	75	–	56	40	32	**29**	**29**	30	46
Medium_late04	65	48	**35**	37	79	–	68	44	38	**35**	**35**	36	49
Medium_late05	178	134	**107**	125	238	–	269	137	122	**107**	**107**	117	161
Medium_hint01	69	48	–	–	89	–	61	46	**40**	–	–	–	–
Medium_hint02	141	94	–	–	194	–	130	92	**91**	–	–	–	–
Medium_hint03	187	**140**	–	–	242	–	184	144	144	–	–	–	–
Long_early01	242	**197**	**197**	**197**	256	214	332	215	**197**	**197**	**197**	**197**	198
Long_early02	277	229	**219**	**219**	299	245	392	248	220	**219**	**219**	222	223
Long_early03	269	**240**	**240**	**240**	286	248	342	246	**240**	**240**	**240**	**240**	242
Long_early04	337	**303**	**303**	**303**	356	317	404	314	**303**	**303**	**303**	**303**	305
Long_early05	327	**284**	**284**	**284**	337	298	376	296	**284**	**284**	**284**	**284**	286
Long_hidden01	445	400	–	–	747	–	4459	417	–	363	**346**	**346**	–
Long_hidden02	130	117	–	–	225	–	1064	123	–	90	**89**	**89**	–
Long_hidden03	59	51	–	–	121	–	156	49	–	**38**	**38**	**38**	–
Long_hidden04	47	29	–	–	134	–	106	32	–	**22**	**22**	**22**	–
Long_hidden05	76	56	–	–	146	–	132	56	–	**41**	**41**	45	–
Long_late01	288	257	**235**	**235**	601	–	580	258	241	**235**	**235**	237	286
Long_late02	293	263	**229**	**229**	596	–	569	259	245	**229**	**229**	**229**	290
Long_late03	306	262	**220**	**220**	585	–	559	268	233	**220**	**220**	222	290
Long_late04	303	261	**221**	**221**	621	–	596	272	246	**221**	222	227	280
Long_late05	141	102	**83**	**83**	393	–	321	112	87	**83**	**83**	**83**	110
Long_hint01	52	42	–	–	134	–	118	44	**33**	–	–	–	–
Long_hint02	39	30	–	–	102	–	114	30	**17**	–	–	–	–
Long_hint03	116	83	–	–	375	–	270	96	**55**	–	–	–	–

as obtained by the other comparative methods. Furthermore, HABC had comparable performance with the other comparative methods in the remaining instances of INRC2010 dataset. In contrast, the performance of the ABC is comparable with M_3 and M_5.

6. Conclusion

This paper tackled the nurse rostering problem using a hybridization of the hill climbing optimizer within the artificial bee colony algorithm, named HABC. The proposed algorithm evaluated using the standard dataset published at the first international nurse rostering competition 2010 (INRC2010) [30]. The performance of the proposed algorithm is compared against eleven other comparative methods that worked on the same INRC2010 dataset. Apparently, the proposed algorithm achieved two new results for two problem instances, while obtained the best published results in 35 out of 69 instances as achieved by other comparative methods. Future work can be directed to: (i) adapting the proposed HABC using the second international nurse rostering competition 2015, where the dataset is available at INRC-II website[2]; (ii) adapting the proposed HABC to other scheduling problems; (iii) measuring the exploration and exploitation capabilities of the proposed approach [48]; (iv) tuning the different parameters of the proposed HABC.

References

[1] H. Millar, M. Kiragu, Cyclic and non-cyclic scheduling of 12 h shift nurses by network programming, Eur. J. Oper. Res. 104 (3) (1998) 582–592.

[2] http://mobiz.vives.be/inrc2/.

[2] B. Cheang, H. Li, A. Lim, B. Rodrigues, Nurse rostering problems – a bibliographic survey, Eur. J. Oper. Res. 151 (3) (2003) 447–460.
[3] E.K. Burke, P. De Causmaecker, G.V. Berghe, H. Van Landeghem, The state of the art of nurse rostering, J. Sched. 7 (6) (2004) 441–499.
[4] J. Van den Bergh, J. Beliën, P. De Bruecker, E. Demeulemeester, L. De Boeck, Personnel scheduling: A literature review, Eur. J. Oper. Res. 226 (3) (2013) 367–385.
[5] K.A. Dowsland, Nurse scheduling with tabu search and strategic oscillation, Eur. J. Oper. Res. 106 (2) (1998) 393–407.
[6] E. Burke, P. De Causmaecker, G.V. Berghe, A hybrid tabu search algorithm for the nurse rostering problem, in: Simulated Evolution and Learning, Springer, 1999, pp. 187–194.
[7] M.J. Brusco, L.W. Jacobs, Cost analysis of alternative formulations for personnel scheduling in continuously operating organizations, Eur. J. Oper. Res. 86 (2) (1995) 249–261.
[8] E. Burke, P. De Causmaecker, S. Petrovic, G.V. Berghe, Variable neighborhood search for nurse rostering problems, in: Metaheuristics: Computer Decision-Making, Springer, 2004, pp. 153–172.
[9] E.K. Burke, T. Curtois, G. Post, R. Qu, B. Veltman, A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem, Eur. J. Oper. Res. 188 (2) (2008) 330–341.
[10] B. Bilgin, P. De Causmaecker, B. Rossie, G.V. Berghe, Local search neighbourhoods for dealing with a novel nurse rostering model, Ann. Oper. Res. 194 (1) (2012) 33–57.
[11] W.J. Gutjahr, M.S. Rauner, An aco algorithm for a dynamic regional nurse-scheduling problem in Austria, Comput. Oper. Res. 34 (3) (2007) 642–666.
[12] B. Maenhout, M. Vanhoucke, An electromagnetic meta-heuristic for the nurse scheduling problem, J. Heuristics 13 (4) (2007) 359–385.
[13] X. Cai, K. Li, A genetic algorithm for scheduling staff of mixed skills under multi-criteria, Eur. J. Oper. Res. 125 (2) (2000) 359–369.
[14] U. Aickelin, K.A. Dowsland, An indirect genetic algorithm for a nurse-scheduling problem, Comput. Oper. Res. 31 (5) (2004) 761–778.
[15] C.-C. Tsai, S.H. Li, A two-stage modeling with genetic algorithms for the nurse scheduling problem, Expert Syst. Appl. 36 (5) (2009) 9506–9512.
[16] E.K. Burke, T. Curtois, R. Qu, G.V. Berghe, A scatter search methodology for the nurse rostering problem, J. Oper. Res. Soc. 61 (11) (2010) 1667–1679.
[17] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACM Comput. Surveys (CSUR) 35 (3) (2003) 268–308.
[18] M. Črepinšek, S.-H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: a survey, ACM Comput. Surveys (CSUR) 45 (3) (2013) 35.
[19] C. Blum, J. Puchinger, G.R. Raidl, A. Roli, Hybrid metaheuristics in combinatorial optimization: a survey, Appl. Soft Comput. 11 (6) (2011) 4135–4151.
[20] R. Bai, E.K. Burke, G. Kendall, J. Li, B. McCollum, A hybrid evolutionary approach to the nurse rostering problem, IEEE Trans. Evol. Comput. 14 (4) (2010) 580–590.
[21] E. Burke, P. Cowling, P. De Causmaecker, G.V. Berghe, A memetic approach to the nurse rostering problem, Appl. Intell. 15 (3) (2001) 199–214.
[22] E. Özcan, Memetic algorithms for nurse rostering, in: P. Yolum, T. Güngör, F. Gürgen, C. Özturan (Eds.), Computer and Information Sciences – ISCIS 2005, Vol. 3733 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2005, pp. 482–492.
[23] D. Karaboga, An Idea Based on Honey Bee Swarm for Numerical Optimization, Techn. Rep. TR06, Erciyes Univ. Press, Erciyes, 2005.
[24] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: artificial bee colony (abc) algorithm and applications, Artif. Intell. Rev. (2012) 1–37.
[25] A. Bolaji, A. Khader, M. Al-Betar, M. Awadallah, Artificial bee colony, its variants and applications: a survey, J. Theor. Appl. Inf. Technol. 47 (2) (2013) 434–459.
[26] A.L. Bolaji, A.T. Khader, M.A. Al-Betar, M.A. Awadallah, University course timetabling using hybridized artificial bee colony with hill climbing optimizer, J. Comput. Sci. 5 (5) (2014) 809–818.
[27] A.L. Bolaji, A.T. Khader, M.A. Al-Betar, M.A. Awadallah, A hybrid nature-inspired artificial bee colony algorithm for uncapacitated examination timetabling problems, J. Intell. Syst. 24 (1) (2015) 37–54.
[28] S. Pulikanti, A. Singh, An artificial bee colony algorithm for the quadratic knapsack problem, in: C. Leung, M. Lee, J. Chan (Eds.), Neural Information Processing, Vol. 5864 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2009, pp. 196–205.
[29] J. Li, Q. Pan, S. Xie, Flexible job shop scheduling problems by a hybrid artificial bee colony algorithm, in: 2011 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2011, pp. 78–83.
[30] S. Haspeslagh, P. De Causmaecker, A. Schaerf, M. Stølevik, The first international nurse rostering competition 2010, Ann. Oper. Res. 218 (1) (2014) 221–236.
[31] M.A. Awadallah, A.T. Khader, M.A. Al-Betar, A.L. Bolaji, Global best harmony search with a new pitch adjustment designed for nurse rostering, J. King Saud Univ. Comput. Inf. Sci. 25 (2) (2013) 145–162.
[32] Z. Lü, J. Hao, Adaptive tabu search for course timetabling, Eur. J. Oper. Res. 200 (1) (2010) 235–244.
[33] C. Valouxis, C. Gogos, G. Goulas, P. Alefragis, E. Housos, A systematic two phase approach for the nurse rostering problem, Eur. J. Oper. Res. 219 (2) (2012) 425–433.
[34] B. Bilgin, P. Demeester, M. Misir, W. Vancroonenburg, G. Vanden Berghe, One hyper-heuristic approach to two timetabling problems in health care, J. Heuristics 18 (3) (2012) 401–434.
[35] E.K. Burke, T. Curtois, New approaches to nurse rostering benchmark instances, Eur. J. Oper. Res. 237 (1) (2014) 71–81.
[36] K. Nonobe, Inrc2010: An approach using a general constraint optimization solver, Tech. rep., INRC2010, 2010, Available from: http://www.kuleuven-kortrijk.be/nrpcompetition
[37] K. Nonobe, T. Ibaraki, A tabu search approach to the constraint satisfaction problem as a general problem solver, Eur. J. Oper. Res. 106 (2) (1998) 599–623.
[38] Z. Lü, J. Hao, Adaptive neighborhood search for nurse rostering, Eur. J. Oper. Res. 218 (3) (2011) 865–876.
[39] H. Santos, T. Toffolo, R. Gomes, S. Ribas, Integer programming techniques for the nurse rostering problem, Ann. Oper. Res. (2014) 1–27.
[40] M.A. Awadallah, A.T. Khader, M.A. Al-Betar, A.L. Bolaji, Hybrid harmony search for nurse rostering problems, in: 2013 IEEE Symposium on, Computational Intelligence in Scheduling (SCIS), IEEE, 2013, pp. 60–67.
[41] K. Anwar, M.A. Awadallah, A.T. Khader, M.A. Al-Betar, Hyper-heuristic approach for solving nurse rostering problem, in: 2014 IEEE Symposium on, Computational Intelligence in Ensemble Learning (CIEL), IEEE, 2014, pp. 1–6.
[42] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Global Optim. 39 (3) (2007) 459–471.
[43] M. Mernik, S.-H. Liu, D. Karaboga, M. Črepinšek, On clarifying misconceptions when comparing variants of the artificial bee colony algorithm by offering a new implementation, Inf. Sci. 291 (2015) 115–127.
[44] K. Sörensen, Metaheuristics – the metaphor exposed, Int. Trans. Oper. Res. 22 (1) (2015) 3–18.
[45] M. Črepinšek, S.-H. Liu, M. Mernik, Replication and comparison of computational experiments in applied evolutionary computing: common pitfalls and guidelines to avoid them, Appl. Soft Comput. 19 (2014) 161–170.
[46] M. Črepinšek, S.-H. Liu, L. Mernik, M. Mernik, Is a comparison of results meaningful from the inexact replications of computational experiments? Soft Comput. (2015) 1–13, http://dx.doi.org/10.1007/s00500-014-1493-4 (in press).
[47] M. Awadallah, A. Khader, M. Al-Betar, A. Bolaji, Harmony search with greedy shuffle for nurse rostering, Int. J. Nat. Comput. Res. 3 (2) (2012) 22–42.
[48] S.-H. Liu, M. Mernik, D. Hrnčič, M. Črepinšek, A parameter control method of evolutionary algorithms using exploration and exploitation measures with a practical application for fitting Sovova's mass transfer model, Appl. Soft Comput. 13 (9) (2013) 3792–3805.