The first international nurse rostering competition 2010 (INRC2010)

Neural Comput & Applic
DOI 10.1007/s00521-015-2076-8

ORIGINAL ARTICLE

Hybridization of harmony search with hill climbing for highly constrained nurse rostering problem

Mohammed A. Awadallah[1] · **Mohammed Azmi Al-Betar**[2] · **Ahamad Tajudin Khader**[3] · **Asaju La'aro Bolaji**[4] · **Mahmud Alkoffash**[2]

Received: 26 January 2015 / Accepted: 11 October 2015

Abstract This paper proposes a hybrid harmony search algorithm (HHSA) for solving the highly constrained nurse rostering problem (NRP). The 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 of constraints. The harmony search is a metaheuristic approach, where the metaheuristics are the most successful methods for tackling this problem. In HHSA, the harmony search algorithm is hybridized with the hill climbing optimizer to empower its exploitation capability. Furthermore, the memory consideration operator of the HHSA is modified by replacing the random selection scheme with the global-best concept of particle swarm optimization to accelerate its convergence rate. The standard dataset published in the first international nurse rostering competition 2010 (INRC2010) was utilized to evaluate the proposed HHSA. Several convergence scenarios have been employed to study the effects of the two HHSA modifications. Finally, a comparative evaluation against twelve other methods that worked on the INRC2010 dataset is carried out. The experimental results show that the proposed method achieved five new best results, and 33 best published results out of 69 instances as achieved by other comparative methods.

Keywords Metaheuristic · Harmony search · Nurse rostering · Hill climbing · Particle swarm optimization

✉ Mohammed A. Awadallah
ma.awadallah@alaqsa.edu.ps

Mohammed Azmi Al-Betar
mohbetar@bau.edu.jo

Ahamad Tajudin Khader
tajudin@cs.usm.my

Asaju La'aro Bolaji
lbasaju@unilorin.edu.ng

Mahmud Alkoffash
alkoffash@yahoo.com

1 Department of Computer Science, Al-Aqsa University, P. O. Box 4051, Gaza, Palestine

2 Department of Information Technology, Al-Huson University College, Al-Balqa Applied University, P.O. Box 50, Al-Huson, Irbid, Jordan

3 School of Computer Sciences, Universiti Sains Malaysia, 11800 Pulau Pinang, Malaysia

4 Department of Computer Science, University of Ilorin, Ilorin, Nigeria

1 Introduction

The nurse rostering problem (NRP) is a combinatorial optimization problem which belongs to the NP-hard category in almost all its variations [65]. Solving a real-world NRP manually often requires a large amount of time and cost. Due to its complex nature and motivation to produce automated nurse rostering system for the hospital administration, considerable researches in this area have been done over the years. Burke et al. [25], Cheang et al. [31] and Den Bergh Jorne [32] have conducted surveys on various methods that are applied for solving the NRP. Many of these methods were specifically proposed to solve real-world NRPs. It can be noted from the literature that the exact solutions have not been achieved by these methods due to the combinatorial nature of this problem. Clearly, the research in this domain is still open and presenting an efficient method to tackle this problem can be considered a significant achievement.

Published online: 24 October 2015

Numerous real-world datasets investigated by the researchers to test their methods are adopted from different hospitals. Some of which are sampled from Vienna hospital, Australia [44]; UK Hospital [33, 54]; Riyadh Al-Kharj hospital, Saudi Arabia [19]; Canada hospitals [59]; Chinese hospitals [48]; Auckland hospitals, New Zealand [58]; Malaysia hospitals [45]; US hospitals [21]; German hospitals [47]; "ORTEC" dataset from Dutch Hospital, Netherlands [27–29, 67]; Queen's Medical Centre University Hospital Trust (QMC) Nottingham, UK [22, 51]; York Hospital in York, PA [50]; Lisbon hospital, Portugal [60]; and "NSPLib" dataset that comes from Belgian hospital [56]. For more information about these methods or the datasets, see the ASAP research group website.[1] However, due to different mathematical formulations and constraints of the real-world datasets, it is almost impossible for the researchers in the domain to compare the results of their methods. Therefore, the INRC2010 has come to the fore.

In order to standardize the NRP dataset, the CODeS research group at Katholieke Universiteit Leuven in Belgium, SINTEF Group in Norway and the University of Udine in Italy organized the first international nurse rostering competition (INRC2010) [46]. The dataset of this competition consists of 69 instances which reflect many cases of the real-world problem that are varied in size and complexity. This dataset is chosen for this research due to its adoption by the nurse rostering researchers to evaluate and compare their methods. The methods used to investigate the INRC2010 dataset are Rizzato [68], Nonobe [62], Lüz [55], Valouxis [72], Bilgin [23], Burke [30], Santos [69], Anwar [10], Asaju [11] and Tassopoulos [71]. The overview of these methods is provided in Sect. 2.

The NRP considered in this research consists of a number of nurses assigned to a set of shifts over a predefined rostering period, which is subject to two types of constraints: *hard* and *soft*. This problem includes two hard constraints (H_1, H_2 as shown in Table 1) which must be satisfied. Furthermore, it consists of 15 soft constraints ($S_1, \ldots, S_{15}$ as shown in Table 1) which should be respected as much as possible. It is important to note that the solution (i.e. roster) is *feasible* if it satisfies the two hard constraints and its *quality* is determined by calculating the penalty of violating the soft constraints. In general, the roster quality is measured by using an objective function. The basic aim is to find a feasible roster with a good enough quality.

Formally, the NRP consists of a set of m nurses, $\mathcal{N} = \{n_1, n_2, \ldots, n_m\}$; each has a specific skill from the set of skill categories $\mathcal{K} = \{k_1, k_2, \ldots, k_q\}$, where q is the total number of skill categories. Each nurse has a specific work contract from the set of contracts $\mathcal{C} = \{c_1, c_2, \ldots, c_w\}$, where w is the total number of contracts. The rostering period is a set of b days, $\mathcal{D} = \{d_1, d_2, \ldots, d_b\}$, and each day has different shifts from a set of r shifts, $\mathcal{S} = \{s_1, s_2, \ldots, s_r\}$. The total number of timeslots is $p = b \times r$, where $\mathcal{T} = \{t_1, t_2, \ldots, t_p\}$ is the set of timeslots. It should be noted that the mathematical formulation of the hard and soft constraints can be seen in [16].

Table 1 Hard and soft constraints as defined by the INRC2010

Symbol	The constraint
H_1	All demanded shifts must be assigned to a nurse
H_2	A nurse can only work one shift per day, i.e. no two shifts can be assigned to the same nurse on a day
S_1	Maximum number of assignments for each nurse during the scheduling period
S_2	Minimum number of assignments for each nurse during the scheduling period
S_3	Maximum number of consecutive working days
S_4	Minimum number of consecutive working days
S_5	Maximum number of consecutive free days
S_6	Minimum number of consecutive free days
S_7	Assign complete weekends
S_8	Assign identical complete weekends
S_9	Two free days after a night shift
S_{10}	Requested day-off
S_{11}	Requested day-on
S_{12}	Requested shift-off
S_{13}	Requested shift-on
S_{14}	Alternative skill
S_{15}	Unwanted patterns

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

$$\min \quad f(\boldsymbol{x}) = \sum_{s=1}^{15} c_s \cdot g_s(\boldsymbol{x}). \tag{1}$$

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 roster $\boldsymbol{x}$.

Metaheuristics are most successful methods that have been used to solve NRP and found to have ability of generating satisfactory solutions within a reasonable time [25]. The metaheuristic methods are classified into two groups: local search-based and population-based methods. The local search-based method starts with one solution and exploits it until a local optimal solution is reached. The main shortcoming of this method is its tendency to getting stuck in problem's local optima. In contrast, the population-based method starts with a set of solutions and explores different search space regions simultaneously

[1] http://www.cs.nott.ac.uk/tec/NRP/.

until the stop condition is reached. The main limitation of this method is that the search does not focus on exploitation of the already-visited regions. In order to strike a right balance between the global exploration and the local exploitation, the researchers tend to hybridize a population-based method as a global explorer agent with a local search-based method as an exploiter agent to complement their advantages and cover their shortcomings. In the recent times, many hybrid methods have been proposed to tackle NRP in order to cope with ruggedness of its search space. Burke [24] proposed a hybrid genetic algorithm to tackle NRP, where the steepest descent improvement heuristic is utilized as a local search procedure. The performance of their hybrid method is better than tabu search and hybrid tabu search. In another development, the hybridization of genetic algorithm with simulated annealing hyper-heuristic is proposed to solve NRP by Bai [20]. Experimentally, the performance of their method outperforms the genetic algorithm. Similarly, the hill climbing optimizer is hybridized within genetic algorithm to solve this problem by Özcan [66]. Other studies that worked on hybrid methods for NRP can be found in [25, 31, 32].

Harmony Search Algorithm (HSA) is a music-inspired metaheuristic population-based algorithm proposed by Geem [39]. It has been successfully applied to a wide variety of optimization problems such as gene selection problems [70], water pump switching [35], structural design [53], vehicle routing [40], water network design [36], tour routing [41], travelling salesman problem [39], course timetabling [1, 4], examination timetabling [3], office-space-allocation [15], Wastewater Treatment Optimization for Fish Migration [42], feature selection [49] and many others [9, 37, 57].

HSA starts with a population of solutions, where the solution is a part of the region in the problem search space. It is worth mentioning that the problem search space includes all different solutions to the problem. Iteratively, the HSA generates one new solution by using its three operators: *memory consideration*, *random consideration* and *pitch adjustment*. The *memory consideration* operator is used to generate a new solution by recombining the features of the current solutions in the population. The *random consideration* operator is employed to diversify the new solution by visiting new regions in the problem search space. Furthermore, the *pitch adjustment* operator is similar to local search-based method, where the neighbourhood structures are incorporated to enhance the new solution locally. If the quality of the new solution is better than the quality of the worst solution in the population, then the HSA replaces the worst solution with the new one and this process is repeated until the stop condition is reached.

HSA has several advantages over other metaheuristics: (i) it needs fewer mathematical requirements in the initial search, (ii) it iteratively generates a candidate solution by considering all existing solutions in the population at each iteration, (iii) it is simple to tailor for different types of optimization problems, and (iv) it has a novel derivative criteria which reduces the number of iterations required to converge towards local minima [38, 52].

Recently, the HSA is being modified and hybridized to improve its performance, due to the combinatorial nature of the search space for the highly constrained optimization problems. These include tuning its parameters [43, 73, 75], modifying its operators such as the memory consideration [2, 7, 77], and the pitch adjustment [4, 64, 73], or hybridized with other metaheuristic components [4, 5, 34, 61, 74, 76]. Quit recently, the structured population techniques are also embedded with the framework of HSA to improve its diversity [6, 8].

In this paper, a hybrid harmony search algorithm (HHSA) is proposed for tackling the NRP using INRC2010 dataset. In the recent time, the HSA has been successfully applied for NRP using the same dataset [12–14, 16–18]. Although, the results produced by the harmony search methods for NRP are very competitive, its performance can be further improved by empowering its local exploitation capability and the convergence rate. Therefore, this paper hybridized the HSA with the hill climbing optimizer (HCO) in order to improve its exploitation capability and thus deal with ruggedness of the NRP solution search space. It should be noted that other local search-based methods (e.g. simulated annealing, great deluge etc.) are available; however, the HCO provides pure exploitation without exploration as required by the HHSA to deal with the nature of the NRP. Furthermore, the memory consideration operator of the HHSA is modified by replacing the random selection scheme with the global-best concept of the particle swarm optimization (PSO) to accelerate its convergence rate. The experimental results using INRC2010 dataset show that the proposed HHSA achieved five new best results, and 33 best published results out of 69 instances as achieved by other comparative methods. The new HHSA could be an alternative approach for the researchers in the domain of NRP.

The remainder of this paper is organized as follows. In Sect. 2, the review of relevant methods that have been employed for INRC2010 dataset shall be provided. The descriptions of the HSA and HHSA for the NRP are given in Sects. 3 and 4, respectively. A small real-world example of applying the HHSA for the NRP is illustrated in Sect. 5. Experimental results of the HHSA are presented and compared with the results of other comparative methods in Sect. 6. Lastly, Sect. 7 outlines the conclusion and future research works.

2 Literature review

In this section, we review all relevant methods that have been applied to solve the NRP using the INRC2010 dataset.

Valouxis [72] used integer programming with some neighbourhood structures 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 used three additional neighbourhood structures in the first phase: (i) rescheduling 1 day, (ii) rescheduling 2 days and (iii) reshuffling the shifts among nurses. In another study, Burke [30] adapted 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 for almost all instances of INRC2010 dataset.

Nonobe [62] modelled NRP as constraint optimization problem (COP) 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 [63]. Similarly, the adaptation of tabu search is investigated by Lüz [55] to solve the NRP. Their solution method consist of two phases: (i) random heuristic is used to construct a feasible roster by allocating the different nurses to the different shifts randomly. (ii) the tabu search algorithm is used to improve the feasible roster locally by using two neighbourhood structures (i.e. move and swap).

In another study, a hybridization of hyper-heuristic with a greedy shuffle move is applied by Bilgin [23] for NRP. The simulated annealing hyper-heuristic is initially adapted to generate a feasible roster and tried to satisfy the soft constraints as much as possible. Then, the greedy shuffle move is used for further improvements. Similarity, Rizzato [68] used a heuristic method for solving the INRC2010 dataset. The heuristic method constructed a feasible roster while simultaneously trying to satisfy five pre-defined soft constraints. Furthermore, three local search procedures are used later for further enhancements.

Santos [69] presented an 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 neighbourhood descent (VND) is used for further improvements. In another study, the harmony search hyper-heuristic is proposed by Anwar [10] to tackle NRP with promising results.

Recently, the presentation of the variable neighbourhood search (VNS) to tackle the INRC2010 dataset is given by Tassopoulos [71] where their method consist of two stages: the different nurses are allocated to whole working days in the first stage, while in the second stage, the nurses are assigned to specific shift types. Note that in these two stages, set of swap procedures are utilized to improve the quality of the solutions. Finally, the artificial bee colony algorithm is investigated by Asaju [11] to solve NRP with good results for the initial investigation.

Table 2 provides the mean rank of the competitors in the INRC2010. This table includes the lowest mean ranks of five competitors for each track. Apparently, Valouxis [72] is the winner, where their method ranked first in all three INRC2010 tracks; Nonobe [62] occupied the second, third and fourth positions in sprint, medium and long tracks, respectively; Lüz [55] ranked third and fourth in the sprint and medium tracks of INRC2010, respectively; Burke [30] employed two solution methods, where the VDS algorithm is ranked fourth for sprint track, and the branch and price algorithm ranked second for medium and long tracks; Bilgin [23] achieved third rank in long track, and fifth rank in sprint and medium tracks and finally, Rizzato [68] ranked fifth in the long track.

3 Harmony search algorithm for NRP

This section presents adaptation of HSA for tackling NRP. Initially, the roster is represented as a vector of allocations $\boldsymbol{x} = (x_1, x_2, \ldots, x_E)$, and each allocation x_i consists of three values (nurse number, shift number, day number). The length of vector $\boldsymbol{x}$ is E and it is calculated as shown in Eq. (2), where the $dmnd_{j,k}$ is the nurses required at shift s_k on day d_j. For example, let $\boldsymbol{x}$ = {(2, 3, 1), (1, 2, 7), …, (10, 1, 14), (4, 3, 4)} be a *feasible* roster. The roster interpreted as follows: the allocation $x_1 = (2, 3, 1)$ refers to nurse n_2 assigned to shift s_3 on day d_1. The second allocation $x_2 = (1, 2, 7)$ refers to nurse n_1 assigned to shift s_2 on day d_7, and so on. Table 3 shows an example of the roster $\boldsymbol{x}$.

$$E = \sum_{j=1}^{b} \sum_{k=1}^{r} dmnd_{j,k}. \tag{2}$$

The HSA includes five main steps that are described below and Algorithm 1 shows the pseudo-code of HSA for NRP.

Table 2 Competitor ranking for the INRC2010

Competitor	Sprint track	Medium track	Long track
Valouxis [72]	2.08	1.77	1.93
Nonobe [62]	2.45	2.30	3.73
Lüz [55]	3.10	3.67	–
Burke [30]	3.30	2.27	2.27
Bilgin [23]	4.07	5.00	2.60
Rizzato [68]	–	–	4.47

Algorithm 1 . Harmony Search Algorithm for NRP

STEP1 Initialize the parameters of NRP and HSA

Set the NRP parameters drawn from the INRC2010 dataset.
Set the HSA parameters (HMCR, PAR, NI, HMS).
Define the roster representation and utilize the objective function.

STEP2 Initialize the harmony memory

Construct rosters of the harmony memory by using heuristic ordering method and store them in ascending order, $\mathbf{HM} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^{\mathrm{HMS}}\}$.
Identify the worst roster in $\mathbf{HM}$, $\boldsymbol{x}^{\mathrm{worst}} = \boldsymbol{x}^{\mathrm{HMS}}$.

STEP3 Improvise a new roster

$\boldsymbol{x}' = \phi$ { New roster}
for $j = 1$ to E **do**
  R_j is defined using Algorithm 2
  if $(U(0,1) \leq \mathrm{HMCR})$ **then**
    if $R_j = \phi$ **then**
      $x'_j \in \boldsymbol{X}_j$ { Random consideration}
    else
      $x'_j \in R_j$ { Memory consideration}
      $rnd = U(0,1)$ { Pitch adjustment}
      if $(rnd \leq \mathrm{PAR})$ **then**
        if $(rnd \leq \mathrm{PAR}/2)$ **then**
          $Move-One-Shift(x'_j)$
        else
          $Swap-One-Shift(x'_j)$
        end if
      end if
    end if
  else
    $x'_j \in \boldsymbol{X}_j$ { Random consideration}
  end if
end for
if $\boldsymbol{x}'$ is not feasible **then**
  $Repair(\boldsymbol{x}')$ { Repair process}
end if

STEP4 Update the harmony memory

if $(f(\boldsymbol{x}') < f(\boldsymbol{x}^{\mathrm{worst}}))$ **then**
  Replaces $\boldsymbol{x}^{\mathrm{worst}}$ by $\boldsymbol{x}'$ in the $\mathbf{HM}$.
  Reordering the rosters in $\mathbf{HM}$ in ascending order.
end if

STEP5 Check the stop criterion

while (the maximum number of improvisations NI is not reached) **do**
  Repeat **STEP3** and **STEP4**
end while

Table 3 Roster $\boldsymbol{x}$ representation

Allocation	x_1	x_2	...	x_{E-1}	x_E
Nurse	n_2	n_1	...	n_{10}	n_4
Shift	s_3	s_2	...	s_1	s_3
Day	d_1	d_7	...	d_{14}	d_4

3.1 STEP1. Initialize the parameters of NRP and HSA

The parameters of NRP are normally drawn from the dataset to be processed. These parameters include the set of nurses, the set of skill categories, the set of shifts, the set of work contracts, the rostering period, matrix of weekly nurses demand, matrices of nurses preferences (day-off, day-on, shift-off, shift-on) and the set of unwanted patterns. The work contracts which includes: maximum and minimum number of shifts allocated to nurse in the rostering period, maximum and minimum number of consecutive working days, maximum and minimum number of consecutive free days, maximum working weekend in 4 weeks and the number of days-off after a series of night shifts.

The HSA parameters which include harmony memory size (HMS), harmony memory consideration rate (HMCR), pitch adjustment rate (PAR) and number of improvisation (NI) that are required to solve NRP are also initialized.

3.2 STEP2. Initialize the harmony memory

The harmony memory (HM) is a space in the computer memory that is utilized to keep a set of feasible rosters as determined by the HMS [see Eq. (3)]. The feasible rosters are constructed using the heuristic ordering strategy and stored in the HM in ascending order based on the objective

Table 4 Ordering of shifts

Shift	Weekly nurses demand							Sum of demand	Shift ordering
	M	T	W	T	F	S	S		
D	5	5	4	5	5	3	3	30	2
L	7	7	6	7	7	5	5	44	4
E	10	10	8	10	10	7	7	62	5
N	6	6	4	6	6	4	4	36	3
DH	2	2	2	2	2	1	1	12	1

function value [see example in Eq. (4)]. The heuristic ordering strategy is employed to sort the different shifts in ascending order based on the required nurses per week (i.e. weekly nurses demand) for each shift (see Table 4). Note that the lowest weekly nurses demand is the most difficult to be assigned. Then, the required nurses are assigned starting from the most difficult and ending with the easiest in accordance with the ordered shifts as generated by heuristic ordering. Furthermore, the worst roster $\boldsymbol{x}^{worst}$ (i.e. the roster with the highest penalty value) in HM is flagged, where $\boldsymbol{x}^{worst} = \boldsymbol{x}^{\mathrm{HMS}}$.

$$\mathbf{HM} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_E^1 & f(x^1) \\ x_1^2 & x_2^2 & \cdots & x_E^2 & f(x^2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_1^{\mathrm{HMS}} & x_2^{\mathrm{HMS}} & \cdots & x_E^{\mathrm{HMS}} & f(x^{\mathrm{HMS}}) \end{bmatrix}. \qquad (3)$$

3.3 STEP3. Improvise a new roster

The new roster $\boldsymbol{x}' = (x_1', x_2', \ldots, x_E')$ is generated based on three operators: (i) memory consideration, (ii) random consideration and (iii) pitch adjustment. The feasibility of the new roster $\boldsymbol{x}'$ is maintained during the improvisation process. If the improvisation process fails to generate a feasible roster, then the repair procedure is invoked to maintain the feasibility of the new roster. The three operators work as follows.

3.3.1 Memory consideration

In this operator, the allocation x_j' in the new roster $\boldsymbol{x}'$ is assigned by a feasible value from a corresponding allocations $x_j' \in \mathbf{R}_j$, where $\mathbf{R}_j \subseteq \{x_j^i | i = 1, 2, \ldots, HMS\}$ with probability (i.e. w.p.) of HMCR, where HMCR $\in$ [0, 1]. Basically, the elements of $\mathbf{R}_j$ are extracted from the rosters stored in the HM by using Algorithm 2. However, if the $\mathbf{R}_j = \phi$, this means there is no feasible value for x_j' to make a feasible new roster. In such case, the *Random Consideration* is triggered to assign the value for x_j'.

For example, the HM in Eq. (4) includes different allocations of three rosters $\boldsymbol{x}^1$ to $\boldsymbol{x}^3$. Each allocation consists of triple values (nurse number, shift number and day number). The first allocation x_1' in the new roster $\boldsymbol{x}'$ is assigned value from $\mathbf{R}_1$, where $\mathbf{R}_1$ = {(1, 3, 1), (2, 2, 1), (3, 1, 1)}. Since this is the first allocation, all elements in $\mathbf{R}_1$ are feasible for x_1'. Let's assume that x_1' takes the first value (1, 3, 1). The process is repeated for the second allocation $x_2' \in \mathbf{R}_2$, where $\mathbf{R}_2$ = {(2, 1, 1), (3, 3, 1)}. The value (1,2,1) is excluded from $\mathbf{R}_2$ because it violates to the second hard constraint H_2 (i.e. the nurse n_1 is assigned two shifts s_3 and s_2 on the same day d_1, where the nurse can be assigned a maximum of one shift on the same day). The same process is used to assign values for other allocations.

$$\mathbf{HM} = \begin{bmatrix} (1,3,1) & (2,1,1) & (3,1,1) & \ldots & (1,2,7) & 43 \\ (2,2,1) & (3,3,1) & (1,3,1) & \ldots & (1,2,4) & 60 \\ (3,1,1) & (1,2,1) & (2,2,1) & \ldots & (3,3,5) & 80 \end{bmatrix}. \qquad (4)$$

Algorithm 2 Procedure of filling $\mathbf{R}_j$

```
for (i = 1 to HMS) do
    flag = false
    R_j = φ
    count = 0
    for (k = 1 to j–1) do
        if Nurse(x'_i) = Nurse(x'_k) and Day(x'_i) = Day(x'_k) then
            flag = true
            GOTO 1
        end if
    end for
    if flag = false then
        for (k = 1 to j–1) do
            if (Day(x'_i) = Day(x'_k) and Shift(x'_i) = Shift(x'_k)) then
                count = count + 1
            end if
        end for
        s = Shift(x'_i)
        d = Day(x'_i)
        if count < dmnd_{d,s} then
            R_j = R_j + x'_i
        end if
    end if
end for
```

3.3.2 Random consideration

This operator randomly selects a value for allocation x_j' from its feasible range $\mathbf{X}_j$ with probability of 1-HMCR where the rules of heuristic ordering are considered. In practice, the elements of $\mathbf{X}_j$ are extracted from the search space of NRP using Algorithm 3. The *memory consideration* and *random consideration* operators select the value of x'_j as follows:

$$x'_j \leftarrow \begin{cases} \mathbf{R}_j & \text{w. p.} \quad \text{HMCR}, \\ \mathbf{X}_j & \text{w. p.} \quad (1 - \text{HMCR}). \end{cases} \tag{5}$$

Algorithm 3 Procedure of filling $\mathbf{X}_j$

1: **for** $(d = 1 \text{ to } b)$ **do**
2:   **for** $(s{=}1 \text{ to } r)$ **do**
3:     **for** $(n{=}1 \text{ to } m)$ **do**
4:       **if** $\text{Skill}(n) \geq \text{skill}(s)$ **then**
5:         **for** $(k = 1 \text{ to } j{-}1)$ **do**
6:           **if** $(n = \text{Nurse}(x'_k) \textit{ and } d = \text{Day}(x'_k))$ **then**
7:             GOTO 3
8:           **else**
9:             $\mathbf{X}_j = \mathbf{X}_j + (n,s,d)$
10:           **end if**
11:         **end for**
12:       **end if**
13:     **end for**
14:   **end for**
15: **end for**

3.3.3 Pitch adjustment

This operator adjusts the allocation x'_j selected by the memory consideration to its neighbouring value with probability of PAR, where PAR $\in$ [0, 1], as follows:

$$\text{pitch adjustment for } x'_j? \leftarrow \begin{cases} \text{Yes} & \text{w. p.} \quad \text{PAR}, \\ \text{No} & \text{w. p.} \quad (1 - \text{PAR}). \end{cases} \tag{6}$$

If the pitch adjustment decision for the allocation x'_j is 'Yes', two pitch adjustment procedures are used to adjust its value, each of which is controlled by a specific PAR range as in Eq. (7).

$$x'_j \leftarrow \begin{cases} \textbf{\textit{Move-One-Shift}}(x'_j) & 0 \leq rnd \leq PAR1, \\ \textbf{\textit{Swap-One-Shift}}(x'_j) & PAR1 < rnd \leq PAR2, \\ \textbf{\textit{Do nothing}} & PAR2 < rnd \leq 1. \end{cases} \tag{7}$$

where *rnd* generates a random number between 0 and 1, PAR1 = (PAR/2) and PAR2 = PAR. Note that the two pitch adjustment procedures ***Move-One-Shift***(x'_j) and ***Swap-One-Shift***(x'_j) have the same chance to be used. Algorithm 4 provides the pseudo-code of the pitch adjustment procedures.

The two proposed pitch adjustment procedures are designed to run as follows:

1. ***Move-One-Shift***(x'_j). The nurse of the selected allocation x'_j is replaced by another nurse selected randomly considering the feasibility with probability of [0, PAR1]. Figure 1 shows an example of this pitch adjustment procedure, where the early shift E is released from nurse n_1 and reassigned to the nurse n_4.
2. ***Swap-One-Shift***(x'_j). The shift of selected allocation x'_j is swapped with another shift from another allocation x'_k on the same day considering the feasibility with probability of (PAR1, PAR2]. Figure 2 shows an example of this pitch adjustment procedure, where the night shift N of nurse n_1 is exchanged with the early shift E of the nurse n_3.

In this work, any local changes that do not improve the new roster, or result in an unfeasible roster, are discarded. It is worth noting that when the improvisation process is completed by using its operators (i.e. memory consideration, random consideration, and pitch adjustment), the new roster is tested for completion (i.e. all allocations are assigned with values). If not complete, the repair process is triggered to fulfil unassigned allocations with feasible values.

3.3.4 Repair process

When the improvisation process (i.e. STEP 3) of the new roster is completed, the feasibility of a new roster must be checked. During the improvisation process of generating a new roster, in cases when *memory consideration* or *random consideration* fails to assign value to some allocations. This will lead to an incomplete new roster. Then, the repair procedure is triggered to give values to the unassigned allocations. For further clarification, the repair process consists of three steps as follows:

- Identify all allocations that are not scheduled in the new roster.
- Identify the day(s) where the nurses demands are not completely scheduled in the new roster.

Algorithm 4 The pseudo-code of the pitch adjustment procedures

$x'_j \in \boldsymbol{x}'$
$OldPenalty = f\ (\boldsymbol{x}')$
$\boldsymbol{x}'' =$ pitch adjustment procedure for(x'_j)
$NewPenalty = f\ (\boldsymbol{x}'')$
if $NewPenalty < OldPenalty$ **then**
  $\boldsymbol{x}' = \boldsymbol{x}''$
end if

Fig. 1 An example of ***Move-One-Shift***(x'_j) procedure. **a** *Before*. **b** *After*

(a) *Before*

	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

(b) *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. 2 An example of ***Swap-One-Shift***(x'_j) procedure. **a** *Before*. **b** *After*

(a) *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

(b) *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

– For each day identified, copy the allocations of the same day from the previous or next week.

3.4 STEP4. Update the harmony memory

If the new roster $\boldsymbol{x}'$ is better than the worst roster $\boldsymbol{x}^{worst}$ in HM, then the new roster replaces the worst roster in the HM.

3.5 STEP5. Check the stop criterion

Repeat **STEP3** and **STEP4** until the NI (maximum number of improvisation) is reached.

4 Hybrid HSA for NRP

In this section, two modifications are proposed to the adaptation of the HSA for NRP. Firstly, the random selection scheme of the memory consideration is replaced by the global-best selection scheme of PSO to improve its convergence rate. Secondly, the HCO is incorporated after the improvisation process as a new operator to empower its exploitation capability. Algorithm 5 shows the pseudo-code of the hybrid HSA for NRP, and the two new amendments are described in the following subsections.

4.1 Replace the random selection scheme by the global-best concept in memory consideration

As aforementioned, the memory consideration of the HSA select the value of the allocation x'_j in the new roster $\boldsymbol{x}'$ from any corresponding value of the same allocation stored in the HM rosters. This selection process is performed randomly without observing the "*survival of the fittest*" principle of natural selection. Recently, Al-Betar [2] proposed different selection schemes based on the *survival of the fittest* principle. Their results in global optimization problems prove that choosing the right selection scheme in the memory consideration has a high impact on the performance of the HSA for this kind of the optimization problems. In another study, Awadallah [17] replaced the random selection scheme of the memory consideration by a set of selection schemes inspired from other evolutionary algorithms. The authors tested the modified method on INRC2010 dataset. The experimental results show that the global-best selection scheme is more useful to improve the performance of the HSA when it is used for solving NRP.

In global-best selection scheme of the memory consideration, the roster with the best quality (i.e.

$\boldsymbol{x}^{best} = \boldsymbol{x}^1$) inside the HM is intensively used in the improvisation process. The value of the allocation x'_j in the new roster $\boldsymbol{x}'$ is selected from the same allocation in the best roster x_j^{best} inside the HM instead of selecting randomly from any roster in the HM with probability HMCR. It should be noted that if the new roster $\boldsymbol{x}'$ is not feasible, then the best second roster $\boldsymbol{x}^2$ is used to give its value x_j^2 for the allocation x'_j, and so on until the last roster $\boldsymbol{x}^{HMS}$ is reached. Clearly, the new roster $\boldsymbol{x}'$ inherits the characteristics of the best roster $\boldsymbol{x}^{best}$ inside the HM in most cases, and this leads to the other rosters in the HM to follow the best roster and thus improves the convergence rate.

For example, suppose the HM in Eq. (4) includes three rosters $\boldsymbol{x}^1$ to $\boldsymbol{x}^3$. The first allocation x'_1 that met the HMCR is assigned with a value from $\mathbf{R}_1$ by memory consideration, where $\mathbf{R}_1 = \{(1,3,1),(2,2,1),(3,1,1)\}$. The (1, 3, 1) is the value of the first allocation x_1^1 in the best roster $\boldsymbol{x}^1$ in the HM, where the value (2, 2, 1) from the best second roster $\boldsymbol{x}^2$ and (3, 1, 1) from the best third roster $\boldsymbol{x}^3$. Since this is the first allocation, the allocation x'_1 takes the value (1, 3, 1) based of the global-best concept. The process is repeated for the second allocation $x'_2 \in \mathbf{R}_2$, where $\mathbf{R}_2 = \{(2,1,1),(3,3,1)\}$. The allocation x'_2 is assigned with (2, 1, 1) if feasible. If not feasible, it will be assigned with (3, 3, 1). The same process is used to assign values for the reminder allocations.

Algorithm 5 Hybrid HSA for NRP

STEP1 Initialize the parameters of NRP and HSA

Set the NRP parameters drawn from the INRC2010 dataset.
Set the HSA parameters (HMCR, PAR, NI, HMS).
Define the roster representation and utilize the objective function.

STEP2 Initialize the harmony memory

Construct rosters of the harmony memory by using heuristic ordering method and store them in ascending order, $\mathbf{HM} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^{\mathrm{HMS}}\}$.
Identify the worst roster in $\mathbf{HM}$, $\boldsymbol{x}^{\mathrm{worst}} = \boldsymbol{x}^{\mathrm{HMS}}$.

STEP3 Improvise a new roster

$\boldsymbol{x}' = \phi$ {new roster}
for $j = 1$ to E **do**
  R_j is defined using Algorithm 2
  if $(U(0,1) \le \mathrm{HMCR})$ **then**
    if $R_j = \phi$ **then**
      $x'_j \in \boldsymbol{X}_j$ {random consideration}
    else
      $x'_j \in R_j$ {memory consideration}
    end if
  else
    $x'_j \in \boldsymbol{X}_j$ {random consideration}
  end if
end for
if $\boldsymbol{x}'$ is not feasible **then**
  $Repair(\boldsymbol{x}')$ { Repair process}
end if

STEP4 Hill climbing optimizer

if $(U(0,1) \le \mathrm{HCR})$ **then**
  while (Local optima is not reached) **do**
    $s = rand(1,10)$
    $j \in (1,E)$
    $\boldsymbol{x}'' = \mathcal{N}_s(x'_j)$
    if $(f(\boldsymbol{x}'') < (f(\boldsymbol{x}'))$ **then**
      $\boldsymbol{x}' = \boldsymbol{x}''$
    end if
  end while
end if

STEP5 Update the harmony memory

if $(f(\boldsymbol{x}') < f(\boldsymbol{x}^{\mathrm{worst}}))$ **then**
  Replaces $\boldsymbol{x}^{\mathrm{worst}}$ by $\boldsymbol{x}'$ in the **HM**.
  Reordering the rosters in **HM** in ascending order.
end if

STEP6 Check the stop criterion

while (the maximum number of improvisations NI is not reached) **do**
  Repeat **STEP3** to **STEP5**
end while

4.2 Hybridize hill climbing as new step

A new step to the adopted HSA has been utilized in order to hybridize HCO as a local exploiter agent, which is called HHSA. In step four of the HHSA (see Fig. 3), each new roster improvised in step three has been tested as whether or not to be enhanced by HCO with a probability of Hill Climbing Rate (HCR). The HCO is simple local search-based method which begins with a new roster. It iterates towards local optimal solution which is in the same region of the new roster using ten neighbourhood structures which are defined below. As shown in Algorithm 5, the pitch adjustment operator is omitted due to similarity in the local search procedures that are used in the hill climbing optimizer.

Note that the usage of HCR parameter is to determine the utilization of HCO. The higher the value of HCR, the higher the usage of the HCO, and thus the higher exploitation capability is provided. Although the rate of convergence is enhanced, the computational time is increased. It is noteworthy to mention that the tabu list is used in each neighbourhood structure of the HCO to avoid any repetition during the search process. The neighbourhood structures used in the HCO are as follows:

- $\mathcal{N}_1$: ***HC_Move***(). This procedure is used to move the shift of current nurse to another nurse who is selected randomly while maintaining feasibility.
- $\mathcal{N}_2$: ***HC_Swap***(). In this procedure, the shift allocated to a specific nurse is exchanged with the shift of another nurse while maintaining feasibility. Note that both nurses are assigned to two different shifts on the same day.
- $\mathcal{N}_3$: ***HC_Token-Ring***(). If the shift of a specific nurse violates the soft constraint S_7 (i.e. partial weekend), 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), the shift of a nurse is exchanged with a shift of another nurse who has shift on the same day.
- $\mathcal{N}_4$: ***HC_Cross-Move***(). This procedure selects two allocations with the same shift given to two different nurses on two different days. Note that the two allocations of these nurses are exchanged, if and only if both nurses have alternative free work days.
- $\mathcal{N}_5$: ***HC_Day-Off-Move***(). For each allocation, if the day of the allocation is preferred to be a *Day-off* for a particular nurse, in such case, this procedure is triggered to move this shift from the current nurse to another nurse who does not violate the *dayOff* preferences while maintaining the feasibility.
- $\mathcal{N}_6$: ***HC_Cyclic-Swap***(). This procedure is used to make cyclic exchanges among four shifts of two nurses in two different work days, and the feasibility should be reserved.
- $\mathcal{N}_7$: ***HC_Move-Pattern***(). This procedure is used to move a group of shifts, with a maximum of 3 days, from the current nurse to another nurse selected randomly while maintaining the feasibility.
- $\mathcal{N}_8$: ***HC_Swap-Pattern***(). This procedure is used to exchange a group of shifts, with a maximum of 3 days, among two nurses, and the feasibility should be maintained.
- $\mathcal{N}_9$: ***HC_Shuffle-Swap1***(). This procedure is used by Burke [26], where all shifts, which are allocated in a period from 1 day to a number of days up to the scheduling period are exchanged between the nurse with the worst schedule and another nurse randomly selected while maintaining the feasibility.
- $\mathcal{N}_{10}$: ***HC_Shuffle-Swap2***(). This procedure is similar to $\mathcal{N}_9$, but it is used to exchange shift patterns between two nurses who are selected randomly.

In this hybridization, any neighbourhood structure that does not lead to improve the new roster, or result in an unfeasible roster, is discarded.

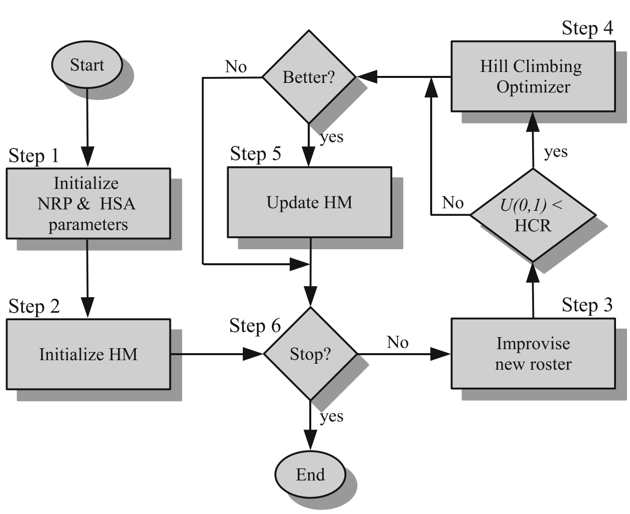

Fig. 3 The flowchart of the HHSA algorithm

5 Illustrative example of tackling NRP using HHSA

Figure 4 illustrates an example of a real-world roster for a ward in a hospital. This roster includes different schedules of five nurses for one week 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. Here, we assume that there are two types of shifts: **D** for a day shift and **N** for a night shift.

	M	T	W	T	F	S	S
	1	2	3	4	5	6	7
n_1	D	D	N	D			
n_2		D	D	N			N
n_3	N				D	D	
n_4			D	D	D		
n_5	D	N			N	N	D

Fig. 4 Illustrative example of a feasible roster

Table 5 Weekly nurses demand

Shift	Weekdays						
	M	T	W	T	F	S	S
D	2	2	2	2	2	1	1
N	1	1	1	1	1	1	1

The roster available in Fig. 4 is feasible, because the two hard constraints (see Table 1) are satisfied. Firstly, the hard constraint H_1 is satisfied, where all demanded shifts are assigned to nurses. Note that the weekly nurses demand is given in Table 5. Secondly, each nurse has one shift per day and this satisfy the second hard constraint H_2. In contrast, the roster contains some violations of soft constraints like the nurse n_1 is assigned **D** shift after **N** shift, which violates the soft constraint S_9 (i.e. two free days after night shift). Furthermore, nurse n_3 is assigned **D** shift on Saturday without assigning any shift on Sunday, this violates the soft constraint S_7 (i.e. assign complete weekend). Nurse n_5 is assigned **D** shift on Saturday and **N** shift on Sunday, which violates the soft constraint S_8 (i.e. identical weekend). However, there is the possibility to generate another feasible roster, where some or all these soft constraints are satisfied.

5.1 STEP1. Initialize the parameters of NRP and HHSA

The parameters of the HHSA are initialized as NI $= 1000$, HMS $= 5$, HMCR $= 0.99$, PAR $= 0.0$ and HCR $= 0.1$. As shown in Fig. 4, the roster has five nurses $\mathcal{N} = \{n_1, n_2, n_3, n_4, n_5\}$; all nurses have the same skill. The rostering period is 7 days; each day split in two shifts $\mathcal{S} = \{\text{D}, \text{N}\}$. This roster is mapped to a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_{19})$, where the length of $\boldsymbol{x}$ is 19 calculated by Eq. (2). Each allocation x_i consists of three values (nurse number, shift number, day number). This vector corresponds to a row in the HM, and the quality of this vector is calculated by using an objective function [see Eq. (1)].

5.2 STEP2. Initialize the harmony memory

The roster shown in Fig. 4 illustrates single feasible roster constructed by the heuristic ordering. The other rosters in HM are generated by the same method as determined by the HMS as shown in Table 6. Note that the rosters in HM are sorted in an ascending order in accordance with their objective function values (see the last column of Table 6).

5.3 STEP3. Improvise a new roster

The HHSA generates a new roster $\boldsymbol{x}' = (x'_1, x'_2, \ldots, x'_{19})$ based on two operators memory consideration and random consideration, where the pitch adjustment is omitted due to the value of PAR $= 0.0$. The different allocations of $\boldsymbol{x}'$ are assigned as follows:

The three allocations x'_2, x'_{10} and x'_{19} are assigned by the random consideration operator, while the other allocations are assigned by the memory consideration operator. The memory consideration operator selects the values for each allocation in the new roster $\boldsymbol{x}'$ from the best roster $\boldsymbol{x}^1$ in the HM. If the value of the best roster does not support the feasibility of the new roster, then this operator will select the value of the best second roster $\boldsymbol{x}^2$ in the HM and so on. In contrast, the random consideration operator assigned values for each allocation from its possible range. After the improvisation step is completed, the new roster could be $\boldsymbol{x}' = [(n_3, \text{N}, 1), (n_5, \text{D}, 1), (n_1, \text{D}, 1), (n_1, \text{D}, 4), \ldots, (n_3, \text{D}, 6)]$, and the quality of the new roster as defined by $f(\boldsymbol{x}') = 110$.

5.4 STEP4. Hill climbing optimizer

Assuming that the HCR condition is met, the new roster $\boldsymbol{x}'$ is exploited locally by the ten neighbourhood structures

Table 6 Illustrative example of harmony memory (HM)

Roster	x_1	x_2	x_3	x_4	$\cdots$	x_{19}	$f(\boldsymbol{x})$
$\boldsymbol{x}^1$	(n_3, N, 1)	(n_2, N, 4)	(n_5, D, 1)	(n_1, D, 4)	$\cdots$	(n_3, D, 6)	127
$\boldsymbol{x}^2$	(n_5, N, 1)	(n_4, N, 4)	(n_1, D, 1)	(n_1, D, 3)	$\cdots$	(n_5, D, 7)	158
$\boldsymbol{x}^3$	(n_3, N, 6)	(n_2, N, 2)	(n_3, D, 5)	(n_1, D, 6)	$\cdots$	(n_3, D, 1)	188
$\boldsymbol{x}^4$	(n_2, N, 7)	(n_1, D, 6)	(n_4, D, 3)	(n_5, N, 4)	$\cdots$	(n_2, D, 2)	232
$\boldsymbol{x}^5$	(n_1, N, 4)	(n_3, N, 3)	(n_2, N, 6)	(n_1, D, 3)	$\cdots$	(n_3, D, 1)	270

Table 7 Illustrative example of updated HM

Roster	x_1	x_2	x_3	x_4	$\cdots$	x_{19}	$f(\boldsymbol{x})$
$\boldsymbol{x}^1$	(n_3, D, 1)	(n_5, N, 1)	(n_1, D, 1)	(n_1, D, 3)	$\cdots$	(n_3, D, 6)	56
$\boldsymbol{x}^2$	(n_3, N, 1)	(n_2, N, 4)	(n_5, D, 1)	(n_1, D, 4)	$\cdots$	(n_3, D, 6)	127
$\boldsymbol{x}^3$	(n_5, N, 1)	(n_4, N, 4)	(n_1, D, 1)	(n_1, D, 3)	$\cdots$	(n_5, D, 7)	158
$\boldsymbol{x}^4$	(n_3, N, 6)	(n_2, N, 2)	(n_3, D, 5)	(n_1, D, 6)	$\cdots$	(n_3, D, 1)	188
$\boldsymbol{x}^5$	(n_2, N, 7)	(n_1, D, 6)	(n_4, D, 3)	(n_5, N, 4)	$\cdots$	(n_2, D, 2)	232

incorporated into HCO towards a local optimal solution. The new roster could be $\boldsymbol{x}' = [(n_3, \mathrm{D}, 1), (n_5, \mathrm{N}, 1), (n_1, \mathrm{D}, 1), (n_1, \mathrm{D}, 3), \cdots, (n_3, \mathrm{D}, 6)]$, with $f(\boldsymbol{x}') = 56$.

It should be noted that the feasibility of the new roster is maintained during the improvisation process (i.e. STEP3) and hill climbing optimizer (i.e. STEP4).

5.5 STEP5. Update the harmony memory

The new roster $\boldsymbol{x}'$ is better than the worst roster $\boldsymbol{x}^5$ in HM as shown in Table 6. Thus the new roster replaces the worst roster in HM, and the rosters in HM are resorted and restored according to objective function value as shown in Table 7.

5.6 STEP6. Check the stop criterion

STEP3 to STEP5 of the HHSA are repeated until NI reaches 1000 iterations.

6 Experimental evaluation and discussions

The proposed method is evaluated using INRC2010 dataset released by Haspeslagh [46] which is freely available at INRC2010 website.[2] The dataset of the INRC2010 comprise 69 problem instances which are categorized into three tracks: sprint, medium and long. They vary in sizes (e.g. number of nurses, skills of nurses, number of contracts, characteristics of contracts, number of shifts, weekend days, unwanted patterns, etc.) and complexity. Each track of the category is divided into four types based on publishing time: 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.

A brief summary of the characteristics of the INRC2010 dataset is fully given in Table 8. This table includes the number of nurses, the nurse skills, the shifts, the work contracts, the number of unwanted patterns and the existence of day-off and shift-off nurse preferences. The main objective is to find a roster that satisfies the hard constraints with the least violations of soft constraints.

6.1 Experimental design

In order to evaluate the performance of the proposed HHSA, it is necessary to study its parameters (i.e. HMS, HMCR and HCR) to find the suitable values to be use when tackling the NRP. Note that the parameters (i.e. HMS and HMCR) which were tested in our previous work [16] are adopted, whereas the PAR value is not adopted because the pitch adjustment operator is removed in the proposed HHSA. Similarly, the last parameter (i.e. HCR) is studied with the global-best selection scheme using three convergence scenarios (see Table 9). Each convergence scenario is run 10 times for each problem instance of the INRC2010 dataset with iteration numbers fixed to 10,000 for sprint and medium track instances and 20,000 for long track instances for all runs. The difference in the number of iterations is based on the time limit set by the INRC2010. It is worthy to note that the INRC2010 time limit is 10 s for spring track, 10 min for medium track, and 10 h for long track.

The convergence scenario that achieved best results among the three (i.e. Sen_1, Sen_2 and Sen_3) is further studied using the memory consideration operator with the original random selection scheme in order to show the effectiveness of the global-best selection scheme in Sen_4.

6.2 Experimental results

The proposed HHSA is programmed using Microsoft Visual C++ version 6.0 under windows Vista. The experiments presented here ran on an Intel Machine with CoreTM processor 2.66 GHz, and 4 GB RAM. In this section, the effects of HCR parameter and the modification of the selection scheme of the memory consideration operator on HHSA for NRP are studied experimentally.

6.2.1 *Studying the effects of HCR parameter on HHSA for NRP*

The first three scenarios (i.e. Sen_1 to Sen_3) are carried out to study the behaviour of the proposed HHSA using the

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

Table 8 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_hidden*01, *Medium_hidden*02, *Medium_hidden*03, *Medium_hidden*04, *Medium_hidden*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*03, *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_late*05, *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*

Table 9 The three convergence scenarios of the HHSA

Scenario	HMS	HMCR	HCR
Sen_1	10	0.99	0.01
Sen_2	10	0.99	0.10
Sen_3	10	0.99	0.30

global-best selection scheme and three varying values of the HCR. The HCR is the probability of enhancing the new roster locally by the HCO until the local optimal solution is reached (i.e. steepest descent). As shown in Table 10, it can be seen that the performance of HHSA is improved as the value of the HCR increases. Experimentally, Sen_1 achieved the best results in six out of 69 instances, while the scenario Sen_2 obtained the best results in 27 out of 69 instances. Sen_3 achieved the best results for all instances of INRC2010 dataset. In general, the HCR with bigger value leads to better results with bigger computational time. It should be noted that when the value of HCR is bigger than 0.30, then the HHSA achieved almost the same results with Sen_3, with undesirable computational time.

Table 11 summarizes experimental results achieved by the HHSA using Sen_3. Note that this scenario achieved the best results for all problem instances of INRC2010 dataset in comparison with other scenarios. The numbers in this table represent the penalty values of 10 runs, which are computed by the objective function [see Eq. (1)]. For each problem instance, the best results (f_{best}), the average (f_{avg}) and standard deviation (σ) of 10 runs are recorded. Furthermore, the computational time of the best results (t_{best}) are also recorded. It should be noted that the proposed

Table 10 Summary of the result; the number of best solutions achieved by HHSA using various HCR values on INRC2010 dataset

Scenario	Sprint instances (33)	Medium instances (18)	Long instances (18)	Total (69)
Sen_1 (HCR = 0.01)	6	0	0	6
Sen_2 (HCR = 0.10)	20	2	5	27
Sen_3 (HCR = 0.30)	33	18	18	69

Table 11 Experimental results achieved by HHSA on INRC2010 dataset include: (f_{best}) the best results, (f_{avg}) the average results and (σ) standard deviation of 10 runs. Furthermore, (t_{best}) the time of the best results, where (*) the time limit in the INRC2010

	f_{best}	f_{avg}	σ	t_{best}(s)
*Sprint_early*01	56	56.5	0.7	4*
*Sprint_early*02	58	58.5	0.5	3*
*Sprint_early*03	51	51.8	0.8	7*
*Sprint_early*04	59	60.1	0.9	4*
*Sprint_early*05	58	58.1	0.3	6*
*Sprint_early*06	54	54.2	0.4	3*
*Sprint_early*07	56	56.7	0.7	3*
*Sprint_early*08	56	56.5	0.5	4*
*Sprint_early*09	55	55.4	0.5	4*
*Sprint_early*10	52	52.6	0.5	6*
*Sprint_hidden*01	32	34.6	1.9	67
*Sprint_hidden*02	32	33.5	1.1	84
*Sprint_hidden*03	62	63.7	2.1	6*
*Sprint_hidden*04	66	67.7	1.3	23
*Sprint_hidden*05	59	59.8	0.9	8*
*Sprint_hidden*06	130	134.7	4.8	47
*Sprint_hidden*07	153	156.5	4.2	5*
*Sprint_hidden*08	204	206.3	2.5	71
*Sprint_hidden*09	338	341.3	3.3	34
*Sprint_hidden*10	306	306.8	2.5	7*
*Sprint_late*01	37	39.9	1.6	23
*Sprint_late*02	42	44.1	1.2	19
*Sprint_late*03	48	50.2	0.9	8*
*Sprint_late*04	73	76.5	2.1	29
*Sprint_late*05	44	45.6	1.2	6*
*Sprint_late*06	42	42.6	0.8	9*
*Sprint_late*07	43	44.3	1.2	8*
*Sprint_late*08	17	17	0.0	5*
*Sprint_late*09	17	17	0.0	4*
*Sprint_late*10	43	45.3	1.3	10*
*Sprint_hint*01	75	76.2	1.2	43
*Sprint_hint*02	43	45.3	1.3	27
*Sprint_hint*03	50	53.4	2.1	36
*Medium_early*01	243	244.7	1.2	999
*Medium_early*02	242	244.5	1.4	567*
*Medium_early*03	238	240.6	1.2	902
*Medium_early*04	240	241.8	1.0	249*
*Medium_early*05	305	307.5	1.5	353*
*Medium_hidden*01	143	153.3	4.4	2173
*Medium_hidden*02	248	262.6	10.5	4301
*Medium_hidden*03	49	53.1	3.2	703
*Medium_hidden*04	87	91.3	2.3	2072
*Medium_hidden*05	169	179.5	5.6	4427
*Medium_late*01	169	178.3	5.8	3514
*Medium_late*02	26	30.5	2.5	2413
*Medium_late*03	34	38.1	2.1	583*

Table 11 continued

	f_{best}	f_{avg}	σ	t_{best}(s)
*Medium_late*04	42	45.3	1.8	1121
*Medium_late*05	131	140.6	6.1	4126
*Medium_hint*01	42	45.8	2.5	1512
*Medium_hint*02	83	92.5	4.1	2434
*Medium_hint*03	130	142.8	7.5	2760
*Long_early*01	197	200.2	2.1	11114*
*Long_early*02	226	228.7	1.3	12021*
*Long_early*03	240	240	0.0	3041*
*Long_early*04	303	305.1	0.9	5112*
*Long_early*05	284	285.7	1.2	8392*
*Long_hidden*01	380	391.1	7.7	21297*
*Long_hidden*02	110	112.8	3.6	18844*
*Long_hidden*03	44	46.8	1.9	14947*
*Long_hidden*04	27	30.4	3.3	24663*
*Long_hidden*05	53	57.3	3.7	26690*
*Long_late*01	253	259	4.9	24811*
*Long_late*02	256	265.1	5.2	16966*
*Long_late*03	256	263.7	6.8	24035*
*Long_late*04	263	271.3	5.2	15406*
*Long_late*05	98	109	6.7	20434*
*Long_hint*01	40	42.9	1.8	15060*
*Long_hint*02	28	30.5	1.8	14251*
*Long_hint*03	81	91.2	6.7	31694*

HHSA obtained the best results in 43 out of 69 instances within the time limit of the INRC2010, where the t_{best} is marked using (*). In contrast, the proposed method needs to relax the time limit of the INRC2010 to achieve the best results for the remaining instances. This is because the hybrid method needs more computational time in comparison with other approaches [24]. Based on the achieved results by the HHSA, the proposed method has high capability of exploiting the solution search space in different ways, thus guided the search towards better results.

Figure 5 plots the best results in HM at each iteration for the proposed HHSA using different HCR values (i.e. Sen_1 (HCR = 0.01), Sen_2 (HCR = 0.10) and Sen_3 (HCR = 0.30)) when exploring the search space for *Long_hidden*05 instance. The selected run is chosen randomly from the ten runs experimented for each scenario. The coloured lines in this plot show the correlation between 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_hidden*05 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, Sen_3 has the best rate of convergence, due to the high value of HCR, and this leads to high speed of convergence.

6.2.2 Studying the effect of global-best on HHSA

In order to compare the differences between the performance of the HHSA with global-best selection scheme (i.e. Sen_3) and the HHSA with random selection scheme (i.e. Sen_4), two experimental scenarios were studied. The aim is to show the advantage of using global-best selection scheme over random selection scheme on the behaviour of HHSA as shown in Table 12. It is noteworthy that the parameter settings in both scenarios are the same (i.e. HMS = 10, HMCR = 0.99 and HCR = 0.3). As shown in Table 12, the HHSA with the global-best selection scheme achieved best results in all problem instances of the INRC2010 dataset, whereas the HHSA with random selection scheme obtained best results in 15 sprint instances as achieved by the HHSA with the global-best selection scheme. The results achieved by the HHSA with global-best scheme is due to the fact that the new roster inherits the characteristics of the best roster in HM which led to an increase in the rate of convergence.

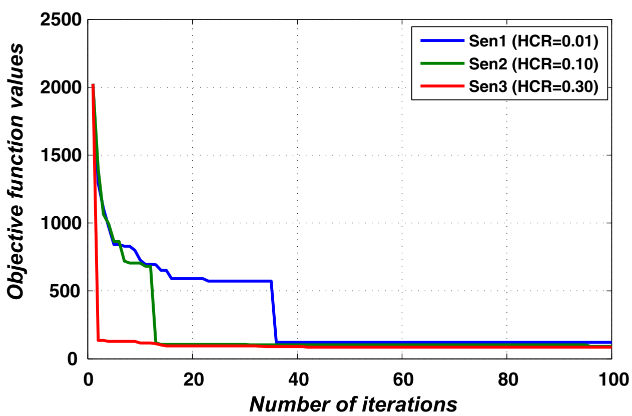

Fig. 5 Effect of various HCR values on the behaviour of HHSA using *Long_hidden*05 instance

6.3 Comparative evaluation and analysis

The best experimental results achieved by the proposed HHSA selected from Table 11 are compared with those obtained by all published methods that worked on the INRC2010 dataset. These include twelve comparative methods as summarized in Table 13.

Table 14 shows the best results obtained by the proposed HHSA in comparison with those obtained by the competitive methods. Note that the numbers in this table refers to

Table 12 Summary of the result; the number of best solutions achieved by HHSA using global-best and random selection schemes on INRC2010 dataset

Scenario	Sprint instances (33)	Medium instances (18)	Long instances (18)	Total (69)
Sen_3 (global-best)	33	18	18	69
Sen_4 (random)	15	0	0	15

Table 13 Key to the comparative methods

Key	Method
HHSA	*The proposed hybrid harmony search algorithm*
M_1	Artificial bee colony algorithm [11]
M_2	Branch and price algorithm [30]
M_3	Constraint optimization problem solver [62]
M_4	Global-best harmony search with new pitch adjustment design [16]
M_5	Harmony search hyper-heuristic algorithm [10]
M_6	Harmony search with greedy shuffle move [14]
M_7	Hyper-heuristic combined with a greedy shuffle approach [23]
M_8	Integer programming with set of neighbourhood structures [72]
M_9	Integer programming [69]
M_{10}	Tabu search with restart mechanism [55]
M_{11}	Two-phase adaptive variable neighbourhood approach [71]
M_{12}	Variable depth search algorithm [30]

Table 14 Summary of the best results achieved by the proposed HHSA and other comparative methods on INRC2010 dataset

Instance	HHSA	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}
*Sprint_early*01	**56**	62	**56**	**56**	58	58	**56**	57	**56**	**56**	**56**	**56**	**56**
*Sprint_early*02	**58**	64	**58**	**58**	60	60	62	59	**58**	**58**	**58**	**58**	**58**
*Sprint_early*03	**51**	58	**51**	**51**	53	53	57	**51**	**51**	**51**	**51**	**51**	**51**
*Sprint_early*04	**59**	66	**59**	**59**	62	62	65	60	**59**	**59**	**59**	**59**	**59**
*Sprint_early*05	**58**	63	**58**	**58**	59	**58**	61	**58**	**58**	**58**	**58**	**58**	**58**
*Sprint_early*06	**54**	58	**54**	**54**	56	55	56	**54**	**54**	**54**	**54**	**54**	**54**
*Sprint_early*07	**56**	61	**56**	**56**	58	58	59	**56**	**56**	**56**	**56**	**56**	**56**
*Sprint_early*08	**56**	58	**56**	**56**	57	**56**	**56**	**56**	**56**	**56**	**56**	**56**	**56**
*Sprint_early*09	**55**	58	**55**	**55**	57	57	59	**55**	**55**	**55**	**55**	**55**	**55**
*Sprint_early*10	**52**	57	**52**	**52**	53	54	54	**52**	**52**	**52**	**52**	**52**	**52**
*Sprint_hidden*01	**32**	44	–	–	41	–	42	–	33	**32**	**32**	**32**	–
*Sprint_hidden*02	**32**	38	–	–	35	–	40	–	**32**	**32**	**32**	**32**	–
*Sprint_hidden*03	**62**	71	–	–	70	–	74	–	**62**	**62**	**62**	**62**	–
*Sprint_hidden*04	**66**	76	–	–	79	–	74	–	67	**66**	**66**	**66**	–
*Sprint_hidden*05	**59**	65	–	–	62	–	65	–	**59**	**59**	**59**	**59**	–
*Sprint_hidden*06	**130**	161	–	–	202	–	165	–	134	**130**	**130**	**130**	–
*Sprint_hidden*07	**153**	178	–	–	196	–	183	–	**153**	**153**	**153**	**153**	–
*Sprint_hidden*08	**204**	245	–	–	266	–	236	–	209	**204**	**204**	**204**	–
*Sprint_hidden*09	**338**	371	–	–	373	–	367	–	**338**	**338**	**338**	**338**	–
*Sprint_hidden*10	**306**	327	–	–	346	–	317	–	**306**	**306**	**306**	**306**	–
*Sprint_late*01	**37**	49	**37**	**37**	45	–	47	40	**37**	**37**	**37**	**37**	**37**
*Sprint_late*02	**42**	52	**42**	**42**	49	–	51	44	**42**	**42**	**42**	**42**	**42**
*Sprint_late*03	**48**	56	**48**	**48**	55	–	55	50	**48**	**48**	**48**	**48**	**48**
*Sprint_late*04	**73**	89	**73**	76	104	–	90	81	75	**73**	**73**	**73**	75
*Sprint_late*05	**44**	53	**44**	45	51	–	52	45	**44**	**44**	**44**	**44**	**44**
*Sprint_late*06	**42**	47	**42**	**42**	43	–	47	**42**	**42**	**42**	**42**	**42**	**42**
*Sprint_late*07	43	52	**42**	43	60	–	50	46	**42**	**42**	**42**	**42**	**42**
*Sprint_late*08	**17**	**17**	**17**	**17**	**17**	–	**17**	**17**	**17**	**17**	**17**	**17**	**17**
*Sprint_late*09	**17**	**17**	**17**	**17**	**17**	–	**17**	**17**	**17**	**17**	**17**	**17**	**17**
*Sprint_late*10	**43**	56	**43**	44	54	–	52	46	**43**	**43**	**43**	**43**	**43**
*Sprint_hint*01	**75**	85	–	–	101	–	88	78	–	–	–	–	–
*Sprint_hint*02	**43**	57	–	–	59	–	54	47	–	–	–	–	–
*Sprint_hint*03	**50**	74	–	–	77	–	67	57	–	–	–	–	–
*Medium_early*01	243	260	**240**	241	270	249	274	242	**240**	**240**	**240**	**240**	244
*Medium_early*02	242	261	**240**	**240**	275	251	275	241	**240**	**240**	**240**	**240**	241
*Medium_early*03	238	259	**236**	**236**	265	247	281	238	**236**	**236**	**236**	**236**	238
*Medium_early*04	240	257	**237**	238	263	248	272	238	**237**	**237**	**237**	**237**	240
*Medium_early*05	305	329	**303**	304	334	315	324	304	**303**	**303**	**303**	**303**	308
*Medium_hidden*01	143	188	–	–	253	–	404	–	130	**111**	117	122	–
*Medium_hidden*02	248	284	–	–	361	–	406	–	221	221	**220**	221	–
*Medium_hidden*03	49	64	–	–	93	–	176	–	36	**34**	35	**34**	–
*Medium_hidden*04	87	100	–	–	135	–	162	–	81	**78**	79	79	–
*Medium_hidden*05	169	201	–	–	275	–	517	–	122	**119**	**119**	124	–
*Medium_late*01	169	206	**157**	176	254	–	231	163	158	**157**	164	161	187
*Medium_late*02	26	52	**18**	19	72	–	46	21	**18**	**18**	20	19	22
*Medium_late*03	34	70	**29**	30	75	–	56	32	**29**	**29**	30	30	46
*Medium_late*04	42	65	**35**	37	79	–	68	38	**35**	**35**	36	**35**	49
*Medium_late*05	131	178	**107**	125	238	–	269	122	**107**	**107**	117	112	161

Table 14 continued

Instance	HHSA	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}
*Medium_hint*01	42	69	–	–	89	–	61	**40**	–	–	–	–	–
*Medium_hint*02	**83**	141	–	–	194	–	130	91	–	–	–	–	–
*Medium_hint*03	**130**	187	–	–	242	–	184	144	–	–	–	–	–
*Long_early*01	**197**	242	**197**	**197**	256	214	332	**197**	**197**	**197**	**197**	**197**	198
*Long_early*02	226	277	**219**	**219**	299	245	392	220	**219**	**219**	222	**219**	223
*Long_early*03	**240**	269	**240**	**240**	286	248	342	**240**	**240**	**240**	**240**	**240**	242
*Long_early*04	**303**	337	**303**	**303**	356	317	404	**303**	**303**	**303**	**303**	**303**	305
*Long_early*05	**284**	327	**284**	**284**	337	298	376	**284**	**284**	**284**	**284**	**284**	286
*Long_hidden*01	380	445	–	–	747	–	4459	–	363	**346**	**346**	349	–
*Long_hidden*02	110	130	–	–	225	–	1064	–	90	**89**	**89**	**89**	–
*Long_hidden*03	44	59	–	–	121	–	156	–	**38**	**38**	**38**	**38**	–
*Long_hidden*04	27	47	–	–	134	–	106	–	**22**	**22**	**22**	**22**	–
*Long_hidden*05	53	76	–	–	146	–	132	–	**41**	**41**	45	**41**	–
*Long_late*01	253	288	**235**	**235**	601	–	580	241	**235**	**235**	237	239	286
*Long_late*02	256	293	**229**	**229**	596	–	569	245	**229**	**229**	**229**	234	290
*Long_late*03	256	306	**220**	**220**	585	–	559	233	**220**	**220**	222	227	290
*Long_late*04	263	303	**221**	**221**	621	–	596	246	**221**	222	227	232	280
*Long_late*05	98	141	**83**	**83**	393	–	321	87	**83**	**83**	**83**	**83**	110
*Long_hint*01	40	52	–	–	134	–	118	**33**	–	–	–	–	–
*Long_hint*02	28	39	–	–	102	–	114	**17**	–	–	–	–	–
*Long_hint*03	81	116	–	–	375	–	270	**55**	–	–	–	–	–

the penalty value (lowest is best) as calculated using Eq. (1). The '–' indicator shows where the method did not run for that particular problem instance. The number highlighted in bold font refers to the best results.

The results show that in *sprint track* instances, the proposed HHSA achieved three new best results for the hint instances (i.e. *Sprint_hint*01 = **75**, *Sprint_hint*02 = **43**, and *Sprint_hint*03 = **50**). Furthermore, in the same track the HHSA is able to achieve 29 best results as achieved by the other comparative methods, while the results of the HHSA are comparable with those obtained by the other methods in the remaining sprint instances. It should be noted that the performance of the HHSA is better than the performance of some of the comparative methods (i.e. M_1, and M_4 to M_7), where these methods are based on population-based approaches. In contrast, the performance of the proposed methods is comparable with the remaining methods, which are local search-based approaches.

For the *medium track*, the proposed HHSA achieved two new best results (i.e. *Medium_hint*02 = **83** and *Medium_hint*03 = **130**) out of 18 instances, while the results of the HHSA are comparable with those obtained by the other comparative methods in the remaining medium instances. It is noteworthy that the performance of the HHSA is better than the performance of some of the comparative methods (i.e. M_1, M_4, M_5, M_6, and M_{12}). In contrast, the remaining comparative methods (i.e. M_2, M_3 and M_7 to M_{11}) have better performance than the proposed HHSA except for the hint instances.

Finally, the proposed HHSA obtained the best published results in four out of 18 instances of the *long track*, while the results of the HHSA are comparable with those obtained by the other comparative methods in the remaining instances. Note that the performance of the HHSA is better than the performance of the M_1, M_4, M_5, M_6 and M_{11} methods, while the performance of the remaining comparative methods are better than the performance of the HHSA.

Clearly, the HHSA successfully achieved five new best results, and obtained the best published results in 33 instances as achieved by other comparative methods. This evidently shows that the proposed HHSA has capability to explore the solution search space of the NRP in different ways in generating the desired solution.

The rank system utilized in the INRC2010 [46] is employed here to rank the comparative methods. Table 15 provides the mean rank of the comparative methods on each type of the problem instance of the INRC2010. The detailed description of the rank system is provided in [46]. As shown in Table 15, the '–' indicator means where the method did not run these problem instances. The number highlighted in bold font refers to the first rank.

Table 15 The ranking of the proposed HHSA and other comparative methods

Problem type	HHSA	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}
Sprint_early	**5.05**	12.90	**5.05**	**5.05**	10.85	9.70	10.70	6.45	**5.05**	**5.05**	**5.05**	**5.05**	**5.05**
Sprint_hidden	**2.80**	6.95	–	–	7.20	–	6.85	–	3.80	**2.80**	**2.80**	**2.80**	–
Sprint_late	5.00	10.55	**4.60**	6.35	9.85	13.00	9.90	8.05	4.95	**4.60**	**4.60**	**4.60**	4.95
Sprint_hint	**1.00**	3.67	–	–	5.00	–	3.33	2.00	–	–	–	–	–
Medium_early	8.30	11.20	**3.20**	5.20	12.30	10.00	12.50	7.10	**3.20**	**3.20**	**3.20**	**3.20**	8.40
Medium_hidden	5.00	6.00	–	–	7.00	–	8.00	–	3.60	**1.60**	2.00	2.80	–
Medium_late	8.00	10.80	**2.00**	6.10	11.60		10.60	6.40	2.30	**2.00**	5.40	4.00	8.80
Medium_hint	**1.33**	4.33	–	–	4.00	–	3.33	2.00	–	–	–	–	–
Long_early	5.40	11.20	**4.20**	**4.20**	12.40	10.00	12.40	4.80	**4.20**	**4.20**	5.00	**4.20**	8.80
Long_hidden	5.00	6.00	–	–	7.40	–	7.60	–	3.00	**2.10**	2.50	2.40	–
Long_late	8.00	10.80	**2.00**	6.10	11.60		10.60	6.40	2.30	**2.00**	5.40	4.00	8.80
Long_hint	2.00	3.00	–	–	4.67	–	4.33	**1.00**	–	–	–	–	–

For the *sprint track*, the proposed HHSA is ranked first as others of the comparative methods for *Sprint_early* and *Sprint_hidden* instances. Furthermore, the HHSA obtained the first rank for the *Sprint_hint* instances, whereas it ranked third in the *Sprint_late* instances. For the *medium track*, the proposed HHSA obtained the first rank for the *Medium_hint* instances, while it ranked fourth, fifth and seventh for the *Medium_early*, *Medium_hidden*, and *Medium_late* respectively. Finally, the proposed HHSA ranked second, fourth, fifth and seventh for the *Long_hint*, *Long_early*, *Long_hidden* and *Long_late* respectively. It should be noted that the M_9 method obtained the first rank in nine out of twelve types, while this method did not compete in the remaining three types.

7 Conclusion and future work

In this present study, the harmony search algorithm has been hybridized with the hill climbing optimizer to improve its exploitation capability. Furthermore, the memory consideration operator of the harmony search has been modified by replacing the random selection scheme with the global-best concept of PSO to accelerate its convergence rate. We tested the proposed method using standard dataset published in INRC2010. The experimental results reveal that the hybrid approach is able to obtain good results in terms of solution quality and time requirements. The future work is to test the effectiveness of different combination of neighbourhood structures in the hill climbing optimizer for NRP, to adapt the proposed HHSA for the second international nurse rostering competition 2015, where the dataset is available at INRC-II website.[3] Furthermore, to investigate the idea of a multi-population harmony search, where the previous good solutions are kept in external archive. The solutions in the archive are used to diversify the solutions in the main population in order to overcome the weakness of convergence.

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

References

1. Al-Betar M, Khader A (2012) A harmony search algorithm for university course timetabling. Ann Oper Res 194(1):3–31
2. Al-Betar M, Doush I, Khader A, Awadallah M (2011) Novel selection schemes for harmony search. Appl Math Comput 218(10):6095–6117
3. Al-Betar M, Khader A, Doush I (2014) Memetic techniques for examination timetabling. Ann Oper Res 218(1):23–50
4. Al-Betar MA, Khader AT, Zaman M (2012) University course timetabling using a hybrid harmony search metaheuristic algorithm. IEEE Trans Syst Man Cybern Part C Appl Rev 42(5):664–681
5. Al-Betar MA, Ahmad ON, Khader AT, Awadallah MA (2013a) Incorporating great deluge with harmony search for global optimization problems. In: Bansal JC, Singh PK, Deep K, Pant M, Nagar A (eds) Proceedings of seventh international conference on bio-inspired computing: theories and applications (BIC-TA 2012), advances in intelligent systems and computing, vol 201. Springer, India, pp 275–286. doi:10.1007/978-81-322-1038-2_24
6. Al-Betar MA, Khader AT, Awadallah MA, Alawan MH, Zaqaibeh B (2013) Cellular harmony search for optimization problems. J Appl Math 2013:1–20. doi:10.1155/2013/139464
7. Al-Betar MA, Khader AT, Geem ZW, Doush IA, Awadallah MA (2013c) An analysis of selection methods in memory consideration for harmony search. Appl Math Comput 219(22):10,753–10,767
8. Al-Betar MA, Awadallah MA, Khader AT, Abdalkareem ZA (2015) Island-based harmony search for optimization problems. Expert Syst Appl 42(4):2026–2035
9. Alia O, Mandava R (2011) The variants of the harmony search algorithm: an overview. Artif Intell Rev 36(1):49–68
10. Anwar K, Awadallah MA, Khader AT, Al-Betar MA (2014) Hyper-heuristic approach for solving nurse rostering problem. In:

2014 IEEE symposium on computational intelligence in ensemble learning (CIEL), pp 1–6
11. Asaju LB, Awadallah MA, Al-Betar MA, Khader AT (2015) Solving nurse rostering problem using artificial bee colony algorithm. In: ICIT 2015 the 7th international conference on information technology, 32–38. doi:10.15849/icit.2015.0005
12. Awadallah M, Khader A, Al-Betar M, Bolaji A (2011a) Nurse rostering using modified harmony search algorithm. In: Panigrahi B, Suganthan P, Das S, Satapathy S (eds) Swarm, evolutionary, and memetic computing, vol 7077., Lecture notes in computer science. Springer, Berlin Heidelberg, pp 27–37
13. Awadallah M, Khader A, Al-Betar M, Bolaji A (2011b) Nurse scheduling using harmony search. In: Sixth international conference on bio-inspired computing: theories and applications (BIC-TA), 2011, pp 58–63
14. Awadallah M, Khader A, Al-Betar M, Bolaji A (2012a) Harmony search with greedy shuffle for nurse rostering. Int J Nat Comput Res (IJNCR) 3(2):22–42
15. Awadallah M, Khader A, Al-Betar M, Woon P (2012b) Office-space-allocation problem using harmony search algorithm. In: Huang T, Zeng Z, Li C, Leung C (eds) Neural information processing, vol 7664., lecture notes in computer science. Springer, Berlin, pp 365–374
16. Awadallah M, Khader A, Al-Betar M, Bolaji A (2013a) Global best harmony search with a new pitch adjustment designed for nurse rostering. J King Saud Univ Comput Inf Sci 25(2):145–162
17. Awadallah M, Khader A, Al-Betar M, Bolaji A (2014) Harmony search with novel selection methods in memory consideration for nurse rostering problem. Asia Pac J Oper Res 31(3):1–39. doi:10.1142/S0217595914500146
18. Awadallah MA, Khader AT, Al-Betar MA, Bolaji AL (2013) Hybrid harmony search for nurse rostering problems. 2013 IEEE Symposium on computational intelligence in scheduling (SCIS) pp 60–67
19. Azaiez M, Al Sharif S (2005) A 0–1 goal programming model for nurse scheduling. Comput Oper Res 32(3):491–508
20. Bai R, Burke E, Kendall G, Li J, McCollum B (2010) A hybrid evolutionary approach to the nurse rostering problem. IEEE Trans Evolut Comput 14(4):580–590
21. Bard J, Purnomo H (2005) Preference scheduling for nurses using column generation. Eur J Oper Res 164(2):510–534
22. Beddoe G, Petrovic S (2006) Enhancing case-based reasoning for personnel rostering with selected tabu search concepts. J Oper Res Soc 58(12):1586–1598
23. Bilgin B, Demeester P, Misir M, Vancroonenburg W, Vanden Berghe G (2012) One hyper-heuristic approach to two timetabling problems in health care. J Heuristics 18(3):401–434
24. Burke E, Cowling P, De Causmaecker P, Berghe G (2001a) A memetic approach to the nurse rostering problem. Appl Intell 15(3):199–214
25. Burke E, De Causmaecker P, Berghe G, Van Landeghem H (2004a) The state of the art of nurse rostering. J Sched 7(6):441–499
26. Burke E, De Causmaecker P, Petrovic S, Berghe GV (2004b) Variable neighbourhood search for nurse rostering problems. In: Resende MGC, de Sousa JP, Viana A (eds) Metaheuristics: computer decision-making, chapter 7. Kluwer Academic Publishers, Norwell, pp 153–172
27. Burke E, Curtois T, Post G, Qu R, Veltman B (2008) A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. Eur J Oper Res 188(2):330–341
28. Burke E, Li J, Qu R (2010) A hybrid model of integer programming and variable neighbourhood search for highly-constrained nurse rostering problems. Eur J Oper Res 203(2):484–493
29. Burke E, Li J, Qu R (2012) A pareto-based search methodology for multi-objective nurse scheduling. Ann Oper Res 196(1):91–109
30. Burke EK, Curtois T (2014) New approaches to nurse rostering benchmark instances. Eur J Oper Res 237(1):71–81
31. Cheang B, Li H, Lim A, Rodrigues B (2003) Nurse rostering problems-a bibliographic survey. Eur J Oper Res 151(3):447–460
32. Den Bergh Jorne V, Bruecker Philippe D, Erik D, Boeck Liesje D (2013) Personnel scheduling: a literature review. Eur J Oper Res 226(3):367–385
33. Dowsland K (1998) Nurse scheduling with tabu search and strategic oscillation. Eur J Oper Res 106(2–3):393–407
34. Gao XZ, Jokinen T, Wang X, Ovaska S, Arkkio A (2010) A new harmony search method in optimal wind generator design. In: 2010 XIX international conference on electrical machines (ICEM). Rome, pp 1–6
35. Geem Z (2005) Harmony search in water pump switching problem. In: Wang L, Chen K, Ong Y (eds) Advances in natural computation, vol 3612., lecture notes in computer science. Springer, Berlin, pp 751–760
36. Geem Z (2006) Optimal cost design of water distribution networks using harmony search. Eng Optim 38(3):259–277
37. Geem Z (2008a) Harmony search applications in industry. In: Prasad B (ed) Soft computing applications in industry, studies in fuzziness and soft computing, vol 226. Springer, Berlin, pp 117–134
38. Geem Z (2008b) Novel derivative of harmony search algorithm for discrete design variables. Appl Math Comput 199(1):223–230
39. Geem Z, Kim J, Loganathan G (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68
40. Geem Z, Lee K, Park Y (2005a) Application of harmony search to vehicle routing. Am J Appl Sci 2(12):1552–1557
41. Geem Z, Tseng CL, Park Y (2005b) Harmony search for generalized orienteering problem: best touring in china. In: Wang L, Chen K, Ong Y (eds) Advances in natural computation, vol 3612, lecture notes in computer science. Springer, Berlin, pp 741–750
42. Geem ZW, Kim JH (2014) Wastewater treatment optimization for fish migration using harmony search. Math Prob Eng 2014
43. Geem ZW, Sim KB (2010) Parameter-setting-free harmony search algorithm. Appl Math Comput 217(8):3881–3889
44. Gutjahr W, Rauner M (2007) An ACO algorithm for a dynamic regional nurse-scheduling problem in Austria. Comput Oper Res 34(3):642–666
45. Hadwan M, Ayob M, Sabar NR, Qu R (2013) A harmony search algorithm for nurse rostering problems. Inf Sci 233:126–140
46. Haspeslagh S, DeCausmaecker P, Schaerf A, Stlevik M (2014) The first international nurse rostering competition 2010. Ann Oper Res 218(1):221–236. doi:10.1007/s10479-012-1062-0
47. Hofe H (1997) Conplan/siedaplan: Personnel assignment as a problem of hierarchical constraint satisfaction. In: Proceedings of the 3rd international conference on practical applications of constraint technologies (PACT-97), vol 97, pp 257–271
48. Huang H, Lin W, Lin Z, Hao Z, Lim A (2014) An evolutionary algorithm based on constraint set partitioning for nurse rostering problems. Neural Comput Appl 25(3–4):703–715
49. Inbarani H, Bagyamathi M, Azar A (2015) A novel hybrid feature selection method based on rough set and improved harmony search. Neural Comput Appl 1–22. doi:10.1007/s00521-015-1840-0
50. De Grano ML, Medeiros D, Eitel D (2009) Accommodating individual preferences in nurse scheduling via auctions and optimization. Health Care Manag Sci 12(3):228–242
51. Landa-silva D, Le K (2008) A simple evolutionary algorithm with self-adaptation for multi-objective nurse scheduling. In: Cotta C, Sevaux M, Srensen K (eds) Adaptive and multilevel

metaheuristics, studies in computational intelligence, vol 136. Springer, Berlin, pp 133–155
52. Lee K, Geem Z (2005) A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput Methods Appl Mech Eng 194(36–38):3902–3933
53. Lee K, Geem Z, Lee S, Bae K (2005) The harmony search heuristic algorithm for discrete structural optimization. Eng Optim 37(7):663–684
54. Li J, Aickelin U (2003) A Bayesian optimization algorithm for the nurse scheduling problem. In: Proceedings of the IEEE Congress on evolutionary computation (CEC 2003), Canberra, Australia, vol 3, pp 2149–2156
55. Lü Z, Hao J (2011) Adaptive neighborhood search for nurse rostering. Eur J Oper Res 218(3):865–876
56. Maenhout B, Vanhoucke M (2007) An electromagnetic meta-heuristic for the nurse scheduling problem. J Heuristics 13(4):359–385
57. Manjarres D, Landa-Torres I, Gil-Lopez S, Del Ser J, Bilbao M, Salcedo-Sanz S, Geem Z (2013) A survey on applications of the harmony search algorithm. Eng Appl Artif Intell 26(8):1818–1831
58. Mason A, Smith M (1998) A nested column generator for solving rostering problems with integer programming. In: Caccetta LSPLYJL, SPLYJL Teo KL, V R (eds) International conference on optimisation: techniques and applications. Citeseer, pp 827–834
59. Millar H, Kiragu M (1998) Cyclic and non-cyclic scheduling of 12 h shift nurses by network programming. Eur J Oper Res 104(3):582–592
60. Moz M, Vaz Pato M (2007) A genetic algorithm approach to a nurse rerostering problem. Comput Oper Res 34(3):667–691
61. Nekkaa M, Boughaci D (2015) Hybrid harmony search combined with stochastic local search for feature selection. Neural Process Lett pp 1–22. doi:10.1007/s11063-015-9450-5
62. Nonobe K (2010) Inrc2010: An approach using a general constraint optimization solver. Tech. rep., INRC2010 (http://www.kuleuven-kortrijk.be/nrpcompetition)
63. Nonobe K, Ibaraki T (1998) A tabu search approach to the constraint satisfaction problem as a general problem solver. Eur J Oper Res 106(2):599–623
64. Omran M, Mahdavi M (2008) Global-best harmony search. Appl Math Comput 198(2):643–656
65. Osogami T, Imai H (2000) Classification of various neighborhood operations for the nurse scheduling problem. In: Goos G, Hartmanis J, Leeuwen J, Lee D, Teng SH (eds) Algorithms and computation, vol 1969., lecture notes in computer science. Springer, Berlin pp 72–83
66. Özcan E (2005) Memetic algorithms for nurse rostering. Proceedings of the 20th international conference on computer and information sciences, ISCIS'05, vol 3733. Springer, Berlin, pp 482–492
67. Qu R, He F (2009) A hybrid constraint programming approach for nurse rostering problems. In: Allen T, Ellis R, Petridis M (eds) Applications and innovations in intelligent systems XVI. Springer, London, pp 211–224
68. Rizzato D, Constantino A, Luiz de Melo E, Landa-Silva D, W R (2010) Heuristic algorithm based on multi-assignment problems for nurse rostering problem. Tech. rep., INRC2010 (http://www.kuleuven-kortrijk.be/nrpcompetition)
69. Santos H, Toffolo T, Gomes R, Ribas S (2014) Integer programming techniques for the nurse rostering problem. Ann Oper Res pp 1–27. doi:10.1007/s10479-014-1594-6
70. Shreem SS, Abdullah S, Nazri MZA (2014) Hybridising harmony search with a Markov blanket for gene selection problems. Inf Sci 258:108–121
71. Tassopoulos IX, Solos IP, Beligiannis GN (2015) A two-phase adaptive variable neighborhood approach for nurse rostering. Comput Oper Res 60:150–169. doi:10.1016/j.cor.2015.02.009
72. Valouxis C, Gogos C, Goulas G, Alefragis P, Housos E (2012) A systematic two phase approach for the nurse rostering problem. Eur J Oper Res 219(2):425–433
73. Wang CM, Huang YF (2010) Self-adaptive harmony search algorithm for optimization. Expert Syst Appl 37(4):2826–2837
74. Wang X, Gao XZ, Ovaska SJ (2009) Fusion of clonal selection algorithm and harmony search method in optimisation of fuzzy classification systems. Int J Bio Inspir Comput 1(1):80–88
75. Yadav P, Kumar R, Panda S, Chang C (2012) An intelligent tuned harmony search algorithm for optimisation. Inf Sci 196(1):47–72
76. Zou D, Gao L, Li S, Wu J, Wang X (2010) A novel global harmony search algorithm for task assignment problem. J Syst Softw 83(10):1678–1688
77. Zou D, Gao L, Li S, Wu J (2011) An effective global harmony search algorithm for reliability problems. Expert Syst Appl 38(4):4642–4648