A 2-stage Approach for the Nurse Rostering Problem

SAY LENG GOH¹, SAN NAH SZE², NASSER R. SABAR³, SALWANI ABDULLAH⁴, AND GRAHAM KENDALL⁵

¹Optimisation Research Group, Faculty of Computing & Informatics, Universiti Malaysia Sabah, Kampus Antarabangsa Labuan, Jln Sungai Pagar, 87000 Labuan, Malaysia (e-mail: slgoh@ums.edu.my)
²Faculty of Computer Science & Information Technology, Universiti Malaysia Sarawak, Jalan Datuk Mohammad Musa, 94300 Kota Samarahan, Sarawak, Malaysia (e-mail: snsze@unimas.my)
³Department of Computer Science and Information Technology, La Trobe University, Melbourne, Australia (e-mail: n.sabar@latrobe.edu.au)
⁴Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 UKM Bangi, Selangor, Malaysia. (e-mail: salwani@ukm.edu.my)
⁵The University of Nottingham Malaysia Campus, Jalan Broga, 43500 Semenyih, Selangor, Malaysia (e-mail: Graham.Kendall@nottingham.edu.my)

Corresponding author: Say Leng Goh (e-mail: slgoh@ums.edu.my)

ABSTRACT In this paper, we are addressing the NP-hard nurse rostering problem utilizing a 2-stage approach. In stage one, Monte Carlo Tree Search (MCTS) and Hill Climbing (HC) are hybridized in finding a feasible solution (satisfying all the hard constraints). We propose a new constant \( C \) value (which balances search diversification and intensification of MCTS) and tree policy/node selection function in the selection procedure of MCTS. In stage two, the feasible solution is further improved using Iterated Local Search (ILS) with Variable Neighbourhood Descent as the local search component. We introduce several unique neighbourhood structures for the ILS. In addition, we propose a novel perturbation strategy to allow the search to escape from local optimum. The proposed methodology is tested on the Shift Scheduling dataset (24 benchmark instances). New best results are reported for seven and two instances for the 10 and 60 minutes run respectively. An in-depth discussion on the attributes of the proposed methodology that lead to its good performance is provided.

INDEX TERMS nurse rostering, hill climbing, monte carlo tree search, iterated local search, variable neighbourhood descent.

I. INTRODUCTION

Combinatorial Optimization Problems (COP) involve finding the values for a set of variables from a discrete search space which maximizes or minimizes an objective function. Examples of these type of problems include vehicle routing [38], traveling salesman, bin packing, minimal spanning tree and timetabling. There are many types of timetabling problems e.g. educational timetabling [12], [41], [42], transportation timetabling [29] and personnel scheduling [46]. Nurse rostering is a specific type of personnel scheduling problem and plays an important role in healthcare management. It involves the assignment of shifts to nurses on a planning horizon (e.g. one month), satisfying a set of hard and soft constraints. The aim is not only to improve the operational efficiency of hospital wards by having an effective utilization of the limited resources, but also to focus on the well-being and job satisfaction of nurses.

Due to the number and nature of constraints, nurse rostering is complex and challenging for both researchers and administrators (personnel managers and head nurses) in hospitals. In fact, the nurse rostering problem is NP-hard [24]. Until recently, most nurse rosters were still constructed manually which can be tedious and time consuming. Having an effective automated nurse roster is crucial. Among issues that may be addressed by having a good roster in hospitals include:

- Under or over staffing. The automated nurse rostering system will ensure that the right number (within a predefined range) of nurses will be assigned to a ward for each shift in a day. This will not only improve the operational efficiency but also reduce the operational cost of the ward.
- Skills mismatch. Nurses with the right qualifications and skills will be assigned to shifts so that healthcare services can be delivered smoothly which will enhance the well-being and life span of the patients.
- Job dissatisfaction of nurses. Shifts will be assigned to nurses according to their requirements spec-
ified in contracts such as minimum/maximum work time, minimum/maximum consecutive shifts, preference/avoidance of shift pattern, night shift assignment, weekend assignment, days on/off, co-workers preference/avoidance etc. A nurse-centred roster will improve the life (health, family and social) quality and morale of nurses which in turn will improve the nursing service quality as well as the experience of patients.

- Shortage of nurses. Nurse shortages in Malaysia is reported in [9]. The authors concluded that a supportive work environment is important, in addition to growing the workforce in addressing the issue. As the well-being of nurses is improved by having an automated rostering system, the nurse turnover rate and thus nursing shortage are expected to be alleviated. Subsequently, this will decrease staff management costs in terms of recruitment, retention and training.

The coronavirus disease 2019 (COVID-19) was declared a pandemic by the World Health Organization (WHO) in March 2020. Since then, it has greatly impacted healthcare front liners in terms of psychological well-being, their morale and work performance due to long working hours under stressful conditions [20]. The urgency of having of an optimal roster is further underlined by the pandemic.

The contributions of this paper are as follows. We apply MCTS in nurse rostering problems for the first time. As far as we are aware, there is no such application found in MCTS in nurse rostering problems for the first time. As the well-being of nurses is improved by having an automated rostering system, the nurse turnover rate and thus nursing shortage are expected to be alleviated. Subsequently, this will decrease staff management costs in terms of recruitment, retention and training.

The remainder of this paper is organized as follows. We apply MCTS in nurse rostering problems for the first time. As far as we are aware, there is no such application found in the scientific literature. MCTS (as a relatively new search methodology) has become the focus of Artificial Intelligence (AI) due to its success in the games domain, particularly Go (where programs based on MCTS are competitive with the best human players) [6]. We propose a new C value (which balances the search diversification and intensification) and tree policy/node selection function in the selection procedure of MCTS. We introduce several unique neighbourhood structures and a novel perturbation strategy to allow ILS to escape from local optimum. The proposed methodology finds several new best results in a comparison to the current state of the art methodologies.

The objective function (Equation 3) is to satisfy the shift requests of nurses (Equations 4 and 5) and minimize under and over staffing (Equation 6). These requirements are soft constraints which means they are optional and their degree of satisfaction will determine the quality of the roster. Thus, we can assign the importance of shift on and off requests to a nurse. Meanwhile, the importance of under and over coverage respectively.

The objective function (Equation 3) is to satisfy the shift requests of nurses (Equations 4 and 5) and minimize under and over staffing (Equation 6). These requirements are soft constraints which means they are optional and their degree of satisfaction will determine the quality of the roster. Thus, we can assign the importance of shift on and off requests to a nurse. Meanwhile, the importance of under and over coverage respectively.

**II. PROBLEM DESCRIPTION**

In this section, we describe the problem and the constraints involved. We present a formal representation of the problem using an integer programming model. The details of the studied problem can be found in [15].

### A. FORMAL REPRESENTATION OF THE PROBLEM

**Parameters:**

- $N$: set of nurses.
- $h$: number of days in the planning horizon.
- $D$: set of days = \{1...h\}.
- $W$: set of weekends = \{1...h/7\}.
- $S$: set of shifts.
- $T_s$: set of shifts that cannot be assigned immediately after shift $s$.
- $D_n$: set of days that nurse $n$ cannot assigned a shift.
- $l_s$: length of shift $s$ in minutes.
- $m_{ns}^{\text{max}}$: max number of shift $s$ that can be assigned to nurse $n$.
- $b_n^{\text{min}}$: min number of minutes that nurse $n$ must be assigned.
- $b_n^{\text{max}}$: max number of minutes that nurse $n$ can be assigned.
- $c_n^{\text{min}}$: min number of consecutive shifts that nurse $n$ must work.
- $c_n^{\text{max}}$: max number of consecutive shifts that nurse $n$ can work.
- $a_n^{\text{min}}$: min number of consecutive days off that nurse $n$ can be assigned.
- $a_n^{\text{max}}$: max number of weekends that nurse $n$ can work.
- $q_{nds}$: penalty if shift $s$ is not assigned to nurse $n$ on day $d$.
- $p_{nds}$: penalty if shift $s$ is assigned to nurse $n$ on day $d$.
- $u_{ds}$: preferred total number of nurses assigned a shift $s$ on day $d$.
- $w_{ds}^{\text{min}}$: weight for cover below the preferred one for shift $s$ on day $d$.
- $w_{ds}^{\text{max}}$: weight for cover above the preferred one for shift $s$ on day $d$.

**Decision Variables:**

- $x_{nds} = \begin{cases} 1 & \text{if nurse } n \text{ is assigned to shift } s \text{ on day } d \\ 0 & \text{otherwise} \end{cases}$

- $k_{nw} = \begin{cases} 1 & \text{if nurse } n \text{ works on weekend } w \\ 0 & \text{otherwise} \end{cases}$

- $y_{ds}$: total below the preferred cover for shift $s$ on day $d$.
- $z_{ds}$: total above the preferred cover for shift $s$ on day $d$.

### Constraints:

The objective function (Equation 3) is to satisfy the shift requests of nurses (Equations 4 and 5) and minimize under and over staffing (Equation 6). These requirements are soft constraints which means they are optional and their degree of satisfaction will determine the quality of the roster. Thus, we can assign the importance of shift on and off requests to a nurse. Meanwhile, the importance of under and over coverage respectively.

$$\text{Minimize } \sum_{i=1}^{3} SC_i$$  \hspace{1cm} (3)
SC1: Shift on penalty.

\[ \sum_{n \in N} \sum_{d \in D} q_{nds} \cdot (1 - x_{nds}) \quad s \in S \] (4)

SC2: Shift off penalty.

\[ \sum_{n \in N} \sum_{d \in D} p_{nds} \cdot x_{nds} \quad s \in S \] (5)

SC3: Date specific cover penalty.

\[ \sum_{d \in D} w_{ds}^{\text{min}} \cdot y_{ds} + \sum_{d \in D} w_{ds}^{\text{max}} \cdot z_{ds} \quad s \in S \] (6)

Minimising the soft constraints (Equation 1) is subject to the fulfilment of the hard constraints presented below. Hard constraints are compulsory. A solution that satisfies all the hard constraints is called a feasible solution.

HC1: A nurse cannot be assigned more than one shift per day.

\[ \sum_{s \in S} x_{nds} \leq 1 \quad n \in N, d \in D \] (7)

HC2: A minimum rest time is required after each shift thus certain shifts cannot follow others. For example, an early shift cannot follow a late shift.

\[ x_{nds} + x_{n(d+1)t} \leq 1 \quad n \in N, s \in S, d \in \{1...|D|\}, t \in T_s \] (8)

HC3: The total number of a shift assigned to a nurse must not exceed the maximum allowed. For example, some nurses do not work night shift or work a maximum number of night shifts.

\[ \sum_{d \in D} x_{nds} \leq m_{ns}^{\text{max}} \quad n \in N, s \in S \] (9)

HC4: The workload (in minutes) of a nurse must be between a minimum and a maximum.

\[ b_{n}^{\text{min}} \leq \sum_{d \in D} l_{s} \cdot x_{nds} \leq b_{n}^{\text{max}} \quad n \in N, s \in S \] (10)

HC5: The number of consecutive shifts assigned to a nurse must not exceed the maximum allowed.

\[ \sum_{j = d}^{d+c_{n}^{\text{max}}} x_{njs} \leq c_{n}^{\text{max}} \quad n \in N, s \in S, d \in \{1...|D| - c_{n}^{\text{max}}\} \] (11)

HC6: The minimum number of consecutive shifts. If the minimum is four, then the sequences {off-on-off}, {off-on-on-off} and {off-on-on-on-off} are not allowed.

\[ x_{nds} + (s - \sum_{j = d}^{d+c_{n}^{\text{max}}} x_{njs}) + x_{n(d+c+1)s} > 0 \quad n \in N, s \in S, e \in \{1...c_{n}^{\text{min}} - 1\}, d \in \{1...|D| - (e + 1)\} \] (12)

HC7: The minimum number of consecutive days off. If the minimum is three, then the sequences {on-off-on} and {on-off-on-on-off} are not allowed.

\[ (1 - x_{nds}) + \sum_{j = d+1}^{d+c} x_{njs} + (1 - x_{n(d+c+1)s}) > 0 \quad n \in N, s \in S, e \in \{1...c_{n}^{\text{min}} - 1\}, d \in \{1...|D| - (e + 1)\} \] (13)

HC8: The maximum number of weekends. A weekend is considered worked if a worker has a shift on either Saturday or Sunday.

\[ k_{nw} \leq x_{n(7w-1)s} + x_{n(7w)s} \leq 2k_{nw} \quad n \in N, s \in S, w \in W \] (14)

\[ \sum_{w \in W} k_{nw} \leq a_{n}^{\text{max}} \quad n \in N \] (15)

III. RELATED WORK

An overview of the nurse rostering problem can be found in [8], [11]. As policies, legalities and regulations differ between countries and hospitals, nurse rostering problems differ in terms of requirements (constraints) [16], [23].

The first international nurse rostering competition (INRC-I) was organized in 2010 with the goal to generate and compare new approaches to the problem in an objective manner [22]. The winner of INRC-I, Valouxis et al. [45] used a systematic two phase approach. In the first phase, the daily workload for each nurse was decided using an integer programming formulation with local search processes, while in the second phase, the specific shifts were assigned using an integer programming formulation. Other finalists of the competition were; Burke and Curtois [7] applied an ejection approach for small instances and a branch and price algorithm for medium and long instances; Nonobe [33] employed a tabu search, with a mechanism to dynamically control the tabu tenure and constraint weights; Bilgin et al. [3] applied a hybrid approach where a greedy shuffle is deployed after running a hyper-heuristic for 80% of the computation time. Recent approaches for the nurse rostering problem include adaptive variable neighbourhood search [43], hybrid harmony search [3], randomized variable neighbourhood search [49], impartiality based iterated local search [32], population-based local search [2] and a hybrid of variable neighbourhood search and dynamic programming [1].

The second international nurse rostering competition (INRC-II) was organized in 2014 [9]. INRC-II is a simplified version of INRC-I. A multi-stage problem formulation was added. History information (border data such last worked shift of each nurse) has to be taken into account by solvers. Among the finalists of the competition were network flow based mixed integer linear programming [17], rotation based branch-and-price [28] and a sequence-based selection hyper-heuristic [26]. Among the competitive approaches post competition were variable neighbourhood search [19], simulated...
annaling [10] and a hyper-heuristic based upon a hidden markov model [25]. Rahimian et al. [35] proposed a hybrid integer programming and variable neighbourhood search algorithm for the Shift Scheduling dataset (24 instances) introduced in [15]. A greedy heuristic was used to generate an initial solution. Then a variable neighbourhood search algorithm and integer programming (IP) based on a ruin and recreate framework was run alternately until the stopping criteria was met. In the framework, a scoring scheme was used to identify high penalty parts of the solution which were destroyed. The solution was then recreated by an IP solver. IP was applied again on the best found solution using the remaining time limit. Strong results were reported. Other approaches applied on these instances were branch-and-price, ejection chain heuristic and Gurobi IP solver as reported in [15]. New approaches applied on these instances were integer programming [39], linear programming based on a column generation heuristic [40] and a hybrid of mixed integer programming and simulated annealing [44]. The other popular NRP benchmark datasets (ORTEC and NSPLib) and the recent solution methodologies to address them are shown in Table 1.

MCTS is a relatively new technique and is being deployed in various application domains such as games [14], feature selection [30] and parameter tuning [17]. MCTS has rarely been used for combinatorial optimization problems. The application of MCTS algorithms on job shop scheduling problems can be found in [13], [37]. Matsumoto et al. applied Single Player MCTS (SP-MCTS) on reentrant scheduling problems [31]. The application of MCTS on variants of the traveling salesman problems can be found in [34]. Goh et al. applied MCTS to address course timetabling problems [18].

IV. PROPOSED METHODOLOGY

The overview of our proposed algorithm is presented in Algorithm 1. We employ a 2-stage approach. In stage 1, we attempt to find a feasible solution where all the hard constraints are satisfied. If a feasible solution is found, stage 2 is activated where we focus on decreasing the soft constraint violations without compromising any of the hard constraints.

Algorithm 1: Nurse Rostering

1. bestSol ← empty solution
2. if FOUNDFEASIBLESOLUTION(bestSol) then
3. ILS(bestSol)
4. else
5. print "No feasible solution found."
6. end

A. STAGE 1: FINDING A FEASIBLE SOLUTION

As shown in Algorithm 2, we try to find a feasible solution for each nurse n utilising Monte Carlo Tree Search (MCTS) and Hill Climbing (HC) (if required). A global feasible solution is found if a feasible solution is found for all the nurses. Note that the f function here is referring to hard constraint violations.

Algorithm 2: FOUNDFEASIBLESOLUTION(bestSol)

1. foreach n ∈ N do
2. MCTS(n, bestSol)
3. if f(bestSol,n) ≠ 0 then
4. HC(n, bestSol)
5. end
6. end
7. if f(bestSol)=0 then
8. return true
9. else
10. return false
11. end

1) Monte Carlo Tree Search (MCTS)

MCTS is depicted in Figure 1. Each node in the tree is a state and each link is an action leading to a state. Each node records an average value and a visit count. There are four main steps in MCTS which are selection, expansion, simulation and back-propagation. In the selection step, the tree is traversed from the root until a leaf node is reached. In the expansion step, a child node is added to the tree. In the simulation step, a simulation is run from the child node to produce an outcome. In the propagation step, the traversed nodes including the child node are updated with values from the outcome. These steps are repeated within available resources.

In our MCTS implementation, in each exploration/iteration, we attempt to assign a shift s to nurse n on each day d (in the planning horizon) in a constructive manner. At the end of the exploration, nodes in MCTS are updated according to a reward value (based on the solution generated). These nodes will guide shift assignment for nurse n in the next exploration. The exploration stops whenever a feasible solution for nurse n is found or a certain number of iterations is reached. We maintain a record of the best solution bestSol for further processing by HC (in case we could not find a feasible solution for nurse n). The class diagram for the node class is shown in Figure 2. The approximate complexity for
MCTS is $O(e|D||S|)$, where $e$ is the number of exploration, $|D|$ is the number of days and $|S|$ is the number of shifts.

The details of the MCTS procedure are presented in Algorithm 3. A root node, $rootNode$ is created (line 1). The diversification co-efficient, $C$ is initialized with a value of 5.0 (based on computational experience). In each exploration, a current solution for nurse $n$, $curSol_n$ is initialized to an empty solution (line 5). Day $d$ is set as 1 (line 6). The $rootNode$ is added to the list of visited nodes, $visitedNode$ (line 7). A tree is traversed in the TREE procedure which comprises of the selection and expansion steps (line 8). Shifts are assigned to nurse $n$ in both the TREE and SIMULATION procedures. SIMULATION returns a reward value which is used to update the visited nodes in the BACKPROPAGATION procedure. $C$ is updated using a decay rate of 0.9999 (line 11).

The details of the TREE procedure are shown in Algorithm 4. $rootNode$ is set as $currentNode$. A tree is traversed through selected nodes until a leaf node is reached (lines 2-7). During tree traversal, the visited nodes are kept in the $visitedNode$ list (line 4). Shifts (of the selected nodes) are assigned to nurse $n$, on day $d$ (line 5) and $d$ is incremented by 1 (line 6). At the leaf node, the tree is expanded by the EXPANSION procedure (line 8). One of the children of $currentNode$ is selected as $childNode$ (line 9). $childNode$ is added to $visitedNode$ (line 10). A shift (of $childNode$) is assigned to nurse $n$, on day $d$ (line 11). $d$ is incremented by 1 (line 12).

The SELECTION procedure is given in Algorithm 5. In this procedure, a node with the highest value among the child nodes of $currentNode$ is selected and returned. Instead of using the common Upper Confidence Bound (UCB) in Equation 16, we use Equation 17 as the node evaluation.
function or tree policy (line 4). In Equation [16], \( v_i \) is the value and \( n_i \) is the visit count of child node \( i \) while \( n_p \) is the visit count of the current (parent) node. It can be observed that unvisited children are given the largest possible value so that all of them are considered at least once. \( C \) is a constant when set higher will promote search diversification as it prioritises less frequently visited nodes. In Equation [17], \( v_i \) is the value of child node \( i \). \( C \) is a co-efficient that determines the priority given to randomness (diversification) when evaluating the child nodes. The random component introduced in the node evaluation function/tree policy decreases the branching factor and effectively increases the tree depth. A deeper tree allows MCTS to make better decisions (in assigning shifts to a nurse) and is therefore more efficient.

### Algorithm 6: EXPANSION\((currentNode, n, d, curSol_n)\)

1. \( foreach \ shift \in validShifts(n) \ do \)
2. \( \quad if \ FEASIBLE(shift, currentNode.shift, n, d, curSol_n) \ then \)
3. \( \quad currentNode.children \leftarrow currentNode.children \cup Node(shift) \)
4. \( \quad end \)

### Algorithm 5: SELECTION\((currentNode, C)\)

1. \( max \leftarrow -1 \)
2. \( selectedNode \leftarrow null \)
3. \( foreach \ node \in currentNode.children \ do \)
4. \( \quad value \leftarrow node.value + C \times \text{RANDOM}(0, 1) \)
5. \( \quad if \ value > max \ then \)
6. \( \quad \quad selectedNode \leftarrow node \)
7. \( \quad \quad max \leftarrow value \)
8. \( end \)
9. \( return \ selectedNode \)

\[
v_i + C \sqrt{\frac{\ln n_p}{n_i}} \quad (16)
\]

\[
v_i + C \times \text{random}(0, 1) \quad (17)
\]

The details of the EXPANSION procedure are shown in Algorithm 6. All the valid and feasible shifts for nurse \( n \), on day \( d \) are added as the child nodes of the current node, \( currentNode \). The feasibility of a shift for a day can be fully tested for the hard constraints; minimum rest time (HC2), maximum total shift (HC3), maximum sequence (HC5), minimum sequence (HC6 and HC7) and pattern (HC8). For example, if a shift is feasible for a day, the violation of these constraints is zero. The workload hard constraint (HC4) is violated when the total time unit is more than a maximum value or less than a minimum value. Therefore, it can only be partially tested (for maximum value). A shift is considered feasible for a day even if the current total time unit is less than a minimum value.

In the SIMULATION procedure (Algorithm 7), a valid and feasible shift is assigned to nurse \( n \) on each remaining day \( d \). All the valid shifts for nurse \( n \) are assigned to \( shifts \) list (line 2). A shift is probabilistically selected from \( shifts \) list (line 5). If the shift is feasible for a particular day, it will be assigned to nurse \( n \) on day \( d \) (line 16). Otherwise, the shift is removed from \( shifts \) list. Another shift is selected and tested for that day. If a valid and feasible shift is nonexistent for a day, a reward value defined in Equation [18] is returned (line 7). After assigning a valid and feasible shift to nurse \( n \) on every remaining day \( d \), \( bestSol_n \) is updated if \( f(curSol_n) < f(bestSol_n) \) and a reward value as shown in Equation [19] is returned (line 23). At this juncture, only the workload constraint (HC4) is possibly violated. Note that we modify the original calculation of the workload constraint violations. For example, if time unit < minimum, the constraint violation is \([\text{minimum} - \text{time unit}] \) (instead of 1). Likewise, if time unit > maximum, the constraint violation is \([\text{time unit} - \text{maximum}] \) (instead of 1).

\[
-(1.0 - \frac{d}{|D|}) \quad (18)
\]

\[
\frac{1.0}{f(curSol_n)} + 1 \quad (19)
\]

The BACKPROPAGATION procedure is shown in Algorithm 8. We increment the visit count and the value (as a cumulative mean of reward) of each node in the \( visitedNode \) list.

2) Hill Climbing (HC)

If MCTS could not find a feasible solution for nurse \( n \), a HC procedure (Algorithm 9) is invoked. For each day \( d \), we try to replace the shifts for a random block of days. We call this neighbourhood structure \( X \)-Extraction (Section IV-B1) where \( X \) is set to a random value between 1 to 4. We focus on minimizing the workload constraint (HC4) violations without compromising other hard constraints (zero violations). As previously noted, we have modified the violation calculation of this constraint. As the name of this procedure suggests, we only accept improving moves. If the candidate solution is worse than the current solution, the original shifts are kept. The process is repeated for 60 seconds. The procedure is exited when a feasible solution for nurse \( n \) is found (line 11).
B. STAGE 2: ITERATED LOCAL SEARCH (ILS)

In stage 2, we focus on minimizing soft constraint violations. If a feasible solution is found, ILS (Algorithm 10) is executed. The procedures VND and PERTURBATION are run alternately until a time limit $t$ is exceeded.

1) Variable Neighbourhood Descent (VND)

The VND procedure is shown in Algorithm 11. $k$ neighbourhood structures are defined (line 1). $k$ is initialised to 1 (line 2). At each iteration, a record of the current cost is kept (line 4) as record. Note that the $f$ function is referring to soft constraint violations. Candidate solutions are generated using the neighbourhood structure $k$. The candidate solution is accepted as the current solution if the candidate cost is less than or equal to the current cost (lines 7-9). The best solution is updated if the current cost is better than the best cost (lines 11-13). If the current cost is equivalent to record, $k$ is incremented by 1 (line 15). Otherwise, $k$ is reset to 1 (line 17). The process is repeated until $k = k_{\text{max}}$. Note that we are using first descent instead of the conventional steepest descent variant of VND, to save computational time in order to allow more transitions (accepted moves) to occur. The approximate complexity for the VND procedure is $O(ik_{\text{max}})$ where $i$ is the number of iterations which varies according to the search landscape. The neighbourhood structures applied are:

1. $\text{NS1 (X-Horizontal Exchange)}$: Two blocks of $X$ shifts (same nurse, different days) are swapped, see Figure 3.
2. $\text{NS2 (X-Vertical Exchange)}$: Two blocks of $X$ shifts (same days, different nurses) are swapped, see Figure 4.
3. $\text{NS3 (X-Extraction)}$: A block of $X$ shifts are extracted and replaced with new shifts, see Figure 5.

In VND, we set the $X$ variable to a random value in the range of 1 to 4.

2) Perturbation

The PERTURBATION procedure is presented in Algorithm 12. $\text{step}$ and $k$ are initialised to 1 (lines 2 and 3). A record of the current cost is kept as record (line 4). We attempt to gradually perturb the current solution until the cost changes
Algorithm 11: VND(curSol, bestSol)

1. neighbourhood structures, $N_k$, $k = 1, 2...k_{max}$
2. $k ← 1$
3. repeat
4. $record ← f(curSol)$
5. repeat
6. generate a candidate solution, $canSol ∈ N_k(curSol)$
7. if $f(canSol) ≤ f(curSol)$ then
8. $curSol ← canSol$
9. end
10. until this end condition
11. if $f(curSol) < f(bestSol)$ then
12. $bestSol ← curSol$
13. end
14. if $f(curSol) = record$ then
15. $k ++$
16. else
17. $k ← 1$
18. end
19. until $k = k_{max}$

FIGURE 3. An example of 2-Horizontal Exchange. E (Early), L (Late) and N (Night) are shifts.

$$\Delta f ≥ 0.05 \text{ (5%)}. \text{ In each iteration, the current solution is perturbed by utilising a neighbourhood structure, } N_k. \text{ The value of } k \text{ is incremented by 1 and the current solution is perturbed again if } \Delta f < 0.05. \text{ When } k = k_{max}, \text{ step is incremented by 1 and } k \text{ is reset to 1. The current solution is repeatedly perturbed if necessary.}$$

The PERTURB procedure is shown in Algorithm 13

A candidate solution is generated using neighbourhood structure $N_k$. The candidate solution is accepted as the current solution if the candidate cost is less than or equal to $record + record \times step \times 0.01$. Obviously, the acceptance criterion is relaxed. The best solution is updated if the current cost is better than the best cost. This process is repeated for $|N| \times |D| \times |S|$ iterations. We utilise the same set of neighbourhood structures in VND. However, the $X$ variable is set to a random value in the range of 5 to 7. The approximate complexity for this procedure is $O(|N||D||S|)$.

V. EXPERIMENTAL RESULTS

The experiments in this paper are conducted on a machine (Intel Xeon 3.3 GHz with 16 GB RAM) running Windows Server 2019. We tested the proposed methodology on the Shift Scheduling dataset introduced by Curtois and Rong Qu [15]. The dataset consists of 24 instances which were designed to reflect real world requirements. They vary from small (8 nurses, 14 days, 1 shift) to large (150 nurses, 364 days, 32 shifts) as shown in Table 2. These instances were designed to represent real world requirements (ranging from easy to challenging) and yet simple to use. The increment of $|N|, |D|$ and $|S|$ from instance 1 to 24 (Table 2) indicates the increasing size of the problem instances. Shift is referring shift types e.g., early, late, night shifts etc. We run the algorithm for a total of 30 times for each instance. Its important to note that all the solutions generated are validated using the software RosterViewer provided by the dataset owner.

A. FINDING A FEASIBLE SOLUTION

1) Comparing C Values

Table 3 shows the Hard Constraint Violations (HCV) and time to feasibility in seconds when using different C values in the SELECTION procedure of MCTS. Note that we are using the original Equation 16 as the tree policy. We propose to set $C$ to 5.0 and gradually decrement it according to $C_{i+1} = C_i \times \alpha$ (decay rate) in each exploration. This geometric cooling schedule is adopted from the Simulated Annealing (SA) algorithm. We omitted the easy instances 1 to 12 as their HCV and time to feasibility are all zero. For the hardest instance 21, setting $C$ too low (1.0) or too high (10.0) is
Algorithm 12: PERTURBATION(curSol, bestSol)
1 neighbourhod structures, \( N_k, k = 1, 2 \ldots k_{\text{max}} \)
2 \( \text{step} \leftarrow 1 \)
3 \( k \leftarrow 1 \)
4 \( \text{record} \leftarrow f(\text{curSol}) \)
5 do
6 \( \text{PERTURB}(\text{curSol}, \text{bestSol}, \text{step}, N_k, \text{record}) \)
7 \( \text{if} \ k = k_{\text{max}} \text{ then} \)
8 \( \text{step} + + \)
9 \( k \leftarrow 1 \)
10 \( \text{end} \)
11 \( \text{else} \)
12 \( \text{k} + + \)
13 \( \text{if} \ \text{record} > f(\text{curSol}) \text{ then} \)
14 \( \Delta f \leftarrow \frac{\text{record} - f(\text{curSol})}{\text{record}} \)
15 \( \text{else} \)
16 \( \Delta f \leftarrow \frac{f(\text{curSol}) - \text{record}}{\text{record}} \)
17 \( \text{end} \)
18 while \( \Delta f < 0.05 \)

Algorithm 13: PERTURB(curSol, bestSol, step, N_k, record)
1 repeat
2 \( \text{generate a candidate solution,} \)
3 \( \text{canSol} \in N_k(\text{curSol}) \)
4 \( \text{if} \ f(\text{canSol}) \leq \text{record} + \text{record} \times \text{step} \times 0.05 \text{ then} \)
5 \( \text{curSol} \leftarrow \text{canSol} \)
6 \( \text{if} \ f(\text{curSol}) < f(\text{bestSol}) \text{ then} \)
7 \( \text{bestSol} \leftarrow \text{curSol} \)
8 \( \text{end} \)
9 until this end condition

TABLE 2. Statistics of the benchmark instances. \(|N|\) is the number of nurses, \(|D|\) is the number of days and \(|S|\) is the number of shifts.

| Instance | \(|N|\) | \(|D|\) | \(|S|\) |
|----------|--------|--------|--------|
| 1        | 8      | 14     | 1      |
| 2        | 14     | 14     | 2      |
| 3        | 20     | 14     | 3      |
| 4        | 10     | 28     | 2      |
| 5        | 16     | 28     | 2      |
| 6        | 18     | 28     | 3      |
| 7        | 20     | 28     | 3      |
| 8        | 30     | 28     | 4      |
| 9        | 36     | 28     | 4      |
| 10       | 40     | 28     | 5      |
| 11       | 50     | 28     | 6      |
| 12       | 60     | 28     | 10     |
| 13       | 120    | 28     | 18     |
| 14       | 32     | 42     | 4      |
| 15       | 45     | 42     | 6      |
| 16       | 20     | 56     | 3      |
| 17       | 32     | 56     | 4      |
| 18       | 22     | 84     | 3      |
| 19       | 40     | 84     | 5      |
| 20       | 50     | 182    | 6      |
| 21       | 100    | 182    | 8      |
| 22       | 50     | 364    | 10     |
| 23       | 100    | 364    | 16     |
| 24       | 150    | 364    | 32     |

TABLE 3. Mean HCV/Mean time to feasibility(s). \(n=30\).

| Values of \(C\) | Proposed \(t\)-test (p value) |
|-----------------|-----------------------------|
| 1.0             | 5.0                         |
| 13              | 0.000.01                    |
| 14              | 0.000.01                    |
| 15              | 0.000.00                    |
| 16              | 0.000.00                    |
| 17              | 0.000.00                    |
| 18              | 0.000.00                    |
| 19              | 0.000.00                    |
| 20              | 0.000.00                    |
| 21              | 0.005.33                    |
| 22              | 0.004.50                    |
| 23              | 0.004.23                    |
| 24              | 0.004.15                    |

Goh et al.: Preparation of Papers for IEEE TRANSACTIONS and JOURNALS
standard deviation for both groups are zero.

| Inst. | Heuristic Based Simulation | MCTS | t-test (p value) |
|-------|---------------------------|------|-----------------|
| 100   | 0.0000/0.00               | 0.0000/0.00 | -               |
| 15    | 0.0000/0.00               | 0.0000/0.00 | -               |
| 16    | 0.0000/0.00               | 0.0000/0.00 | -               |
| 17    | 0.0000/0.00               | 0.0000/0.00 | -               |
| 18    | 0.0000/0.00               | 0.0000/0.00 | -               |
| 19    | 0.0000/0.00               | 0.0000/0.00 | -               |
| 20    | 0.0000/0.00               | 0.0000/0.00 | -               |
| 21    | 0.0000/0.00               | 0.0000/0.00 | -               |
| 22    | 2404.20/-                 | 0.0000/12.04 | 0.000          |
| 23    | 0.0000/0.00               | 0.0000/0.00 | -               |
| 24    | 0.0000/0.00               | 0.0000/0.00 | -               |
| Avg   | 0.0000/0.00               | 0.0000/0.00 | -               |

**TABLE 5.** Mean HCV/ Mean time to feasibility(s). n=30.

4) Feasibility

Table 4 shows the feasibility and time to feasibility of the solutions. The combination of MCTS and HC managed to find a feasible solution in every run for each instance. A feasible solution is found in < 1 second for instances 1 to 20. It takes ≤ 12 seconds for instances 22 to 24. Instance 21 seems to be the most challenging as it takes between 7 and 15 seconds to find a feasible solution.

**TABLE 6.** Feasibility (%) and time to feasibility. n=30.

| Inst. | Feasibility (%) | Minimum | Average | Maximum |
|-------|-----------------|---------|---------|---------|
| 1     | 100             | 0       | 0.00    | 0       |
| 2     | 100             | 0       | 0.00    | 0       |
| 3     | 100             | 0       | 0.00    | 0       |
| 4     | 100             | 0       | 0.00    | 0       |
| 5     | 100             | 0       | 0.00    | 0       |
| 6     | 100             | 0       | 0.00    | 0       |
| 7     | 100             | 0       | 0.00    | 0       |
| 8     | 100             | 0       | 0.00    | 0       |
| 9     | 100             | 0       | 0.00    | 0       |
| 10    | 100             | 0       | 0.00    | 0       |
| 11    | 100             | 0       | 0.00    | 0       |
| 12    | 100             | 0       | 0.00    | 0       |
| 13    | 100             | 0       | 0.00    | 0       |
| 14    | 100             | 0       | 0.00    | 0       |
| 15    | 100             | 0       | 0.00    | 0       |
| 16    | 100             | 0       | 0.00    | 0       |
| 17    | 100             | 0       | 0.00    | 0       |
| 18    | 100             | 0       | 0.00    | 0       |
| 19    | 100             | 0       | 0.00    | 0       |
| 20    | 100             | 0       | 0.06    | 0       |
| 21    | 100             | 7       | 12.04   | 15      |
| 22    | 100             | 1       | 5.52    | 12      |
| 23    | 100             | 2       | 4.54    | 9       |
| 24    | 100             | 2       | 4.39    | 8       |

**B. IMPROVING THE QUALITY OF SOLUTION**

1) Comparing the Influence of Neighbourhood Structures

To observe the influence of the proposed neighbour structures, we run each with a combination of neighbour structure(s) iteratively in a sequential manner using hill climbing acceptance criterion, with a runtime of 1 minute. The mean soft constraint violations are shown in Table 7. From observation, combinations of neighbourhood structures give better results than any individual neighbourhood structure. The combination of <NS1+NS2+N3> is good for most instances. The search may not have encountered local optimums yet for these instances where this combination of heuristics is not that effective.

2) Convergence Behaviour of the Proposed ILS

We run the proposed ILS on instance 8 using a random seed of 1. A hill climbing (HC) is run using the same seed for comparison purpose. Figure 6 shows the convergence behaviour of both the proposed ILS and HC. It appears that the HC stagnates relatively early at around 2200 (soft constraint violations). HC is assumed to be trapped in a local optimum. Meanwhile, the proposed ILS went through a series of fluctuations in terms of current cost (due to perturbations) and finally achieves a lower soft constraint violations (approx. 1500). It shows the perturbations in the proposed ILS are effective in helping the search to escape from local optimums.

3) Comparing with the State-of-the-art Methodologies

Table 9 shows the quality of the solutions generated by the proposed algorithm with a runtime of 10 minutes, in comparison with other state of art methods (the details are given in Table 6). The dash "-" symbol indicates that the algorithm could not find a feasible solution within the time limit. The optimal solutions are underlined. The bold values indicate the best known result for an instance. Our proposed methodology obtained the best results for 10 instances (1, 2, 4, 18, 19, 20, 21, 22, 23 and 24). For reference, the solver A (Branch & Price) was run without time limit. The solvers B (Gurobi) and C (Ejection Chain) were run on a (Intel Core i3 2 Duo 3.16 GHz and 8 GB RAM) computer. The solver D (Branch & Price) was run without time limit. The optimal solutions are underlined. The bold values indicate the best known result for an instance. Our proposed methodology obtained the best results for 10 instances (1, 2, 4, 18, 19, 20, 21, 22, 23 and 24). The solver E (Hybrid IP+VNS) was performed on a (Intel Core i3 3.40 GHz and 4GB RAM) computer. Meanwhile, the solver F (Gurobi) and G (Gurobi) seem to perform well on small instances. Both the hybrids.
The influence of the proposed neighbourhood structures. Shown are mean soft constraint violations. Other state of art methods. The proposed algorithm reduces the runtime of 60 minutes, we run the proposed algorithm in finding a feasible solution faster and it therefore has more workload constraint (HC4). We can only prevent a shift nurse on a particular day. All the hard constraints (HC3, HC5, HC6, HC7 and HC8) can be fully tested except vacancy is set relatively lower than selecting any shift. This alleviates work underloading, the probability of selecting a time unit is less than a minimum after its assignment. To TABLE 7. The influence of the proposed neighbourhood structures. Shown are mean soft constraint violations. n=10. TABLE 8. Solver details

As the other state of the art methods were compared using the runtime of 60 minutes, we run the proposed algorithm for the same time limit. Table 10 compares the solutions generated by the proposed algorithm with those produced by other state of art methods. The proposed algorithm reduces the mean results for all the instances. It attains the best results for instances 1, 2, 3, 4, 21 and 24. Note that we did not include solvers F and G here as their runtime exceeded the time limit of 60 minutes which hinges an objective comparison.

VI. DISCUSSION

In MCTS, feasibility is tested before assigning a shift to a nurse on a particular day. All the hard constraints (HC3, HC5, HC6, HC7 and HC8) can be fully tested except workload constraint (HC4). We can only prevent a shift assignment that will cause the time unit (of a nurse) to exceed the maximum. A shift is considered feasible even though the time unit is less than a minimum after its assignment. To alleviate work underloading, the probability of selecting a set is relatively lower than selecting any shift. This setting also applies to HC, VND and perturbation procedures where the neighbourhood structure X-Extraction is utilized. MCTS is utilized in finding a feasible solution. Due to the
randomness property of the simulation component, MCTS does not possess the precision required especially in a tight environment. HC helps to fine tune the solution in terms of workload constraint violations.

We propose to set the $C$ (Equation 17 in the SELECTION procedure) to a fixed value and gradually decrement it using a geometric cooling schedule. Setting $C$ to a relatively high value in the beginning allows search diversification as nodes are randomly selected. As the search progress, $C$ is decremented as well as the priority given to the random component. Effectively, priority is increasingly given to the value $v_i$ component when selecting nodes. This allows the search to intensify based on the information collected earlier towards the end of the search. Therefore, our proposed $C$ value is effective.

A tree policy (Equation 17) for node evaluation in the SELECTION procedure is proposed. The commonly used UCB tree policy (Equation 16) gives unvisited nodes largest possible values so that they are considered at least once as a way to promote search diversification. However, we think that it is unnecessary to consider every unvisited node and replace the visit component in the equation with a random component instead. Therefore, the proposed tree policy manages to find feasible solutions in shorter times as shown in Table 4.

Note that we modify the calculation of workload constraint (HC4) violations. The original calculation returns a penalty of 1 for a nurse whose total time unit is either more than a maximum or less than a minimum. The modified calculation returns a penalty of [minimum-time unit] if time unit < minimum and [time unit-maximum] if time unit > maximum. For example, at one time point, the time unit for a nurse is 5 and maximum is 1. The violation is 1 (based on original calculation) and 4 (based on modified calculation). In the original calculation, it is seen as equal cost move. In the modified calculation, it is seen as a worsening move. This difference affects the acceptance and rejection of moves. This modification is important in providing the accuracy required in the calculation of the reward value (Equation 19 in the SIMULATION procedure so that MCTS can work effectively. Furthermore, this modification is crucial for our HC (Algorithm 9) to work properly.

Several new best results are achieved for the runtimes of 10 and 60 minutes as presented in Tables 9 and 10. We hold a competitive advantage as our proposed MCTS and HC combo can find a feasible solution quickly. This leaves us with plenty of remaining time to focus on improving the solution quality.

We opt to utilise VND as the local search for ILS because of its diversification capability. It can perform effective search through the employment of multiple neighbourhood structures. In addition, VND serves as a natural indicator of local optimum when it stops running (after considering all the neighbourhood structures).

From observation, a stagnating current cost indicates that the search is stuck in a local optimum. Therefore, when the execution of VND stops, we perturb the solution until

| Inst. | A          | B          | C          | D          | E          | MCTS + ILS          |
|-------|------------|------------|------------|------------|------------|--------------------|
| 1     | 607        | 607        | 607        | 607        | 607        | 607 (607.00)       |
| 2     | 828        | 828        | 828        | 828        | 828        | 828 (828.00)       |
| 3     | 1001       | 1002       | 1001       | 1001       | 1001       | 1001 (1002.60)     |
| 4     | 1716       | 1716       | 1716       | 1716       | 1716       | 1716 (1719.00)     |
| 5     | 1160       | 1143       | 1138       | 1143       | 1143       | 1150 (1244.40)     |
| 6     | 1952       | 1950       | 2258       | 1950       | 1950       | 2048 (2053.40)     |
| 7     | 1058       | 1056       | 1269       | 1056       | 1056       | 1077 (1080.60)     |
| 8     | 1308       | 1323       | 2260       | 1344       | 1322       | 1374 (1437.20)     |
| 9     | 439        | 439        | 463        | 439        | 439        | 491 (494.60)       |
| 10    | 4631       | 4631       | 4797       | 4631       | 4631       | 4663 (4682.00)     |
| 11    | 3443       | 3443       | 3661       | 3443       | 3443       | 3437 (3463.20)     |
| 12    | 4046       | 4040       | 5211       | 4040       | 4040       | 4173 (4179.40)     |
| 13    | -          | 3109       | 3037       | 1905       | 2900       | 3224 (3260.40)     |
| 14    | -          | 1280       | 1847       | 1279       | 1280       | 1324 (1331.40)     |
| 15    | -          | 4964       | 5935       | 3928       | 4190       | 4366 (4443.00)     |
| 16    | 3323       | 3233       | 4048       | 3255       | 3255       | 3435 (3511.00)     |
| 17    | -          | 5851       | 7835       | 5750       | 5848       | 5913 (5942.40)     |
| 18    | -          | 4760       | 6404       | 4662       | 4650       | 4904 (4930.40)     |
| 19    | -          | 5420       | 5531       | 3224       | 3218       | 3425 (3560.80)     |
| 20    | -          | -          | 9750       | 4913       | -          | 5063 (5131.80)     |
| 21    | -          | -          | 36688      | 23191      | -          | 21731 (21775.00)   |
| 22    | -          | -          | 516686     | 32126      | -          | 34855 (35303.80)   |
| 23    | -          | -          | 54384      | 3794       | -          | 18947 (19133.20)   |
| 24    | -          | -          | 156858     | 2281440    | -          | 56001 (56469.80)   |

**TABLE 10.** Comparison of MCTS + ILS with the state of the art methods with runtime of 60 minutes. Shown are best (mean) results in terms of soft constraint violations. n=5.
the current cost changes at least 5%. We gradually (step by step) relax the acceptance criterion in case the current cost is still stagnant. The difference between the neighbourhood structures applied in VND and perturbation is the size range of block X. We hope the variation will make the search space more connected and allow the search to diversify and escape from local optimum in case it is trapped.

VII. CONCLUSION

We presented the implementation details of utilising a hybrid of MCTS and HC algorithms in finding feasible solutions. Specifically, the proposed C value (SELECTION procedure of MCTS) was compared with other fixed values. MCTS and HC perform best by gradually decrementing an initial C value based on a geometric cooling schedule. We also compared the proposed tree policy with the UCB tree policy commonly utilized in MCTS. The proposed tree policy allows feasible solutions to be found in shorter times. In addition, we demonstrated the importance of the learning component in MCTS. MCTS and HC manage to achieve 100% feasibility quickly. Furthermore, an ILS (VNS as local search) was proposed in improving solution quality. The proposed neighbourhood structures were shown to be suitable for the problem instances. A convergence curve of the proposed ILS was presented which shows its effectiveness in escaping from local optima. Finally, a comparison was made between the proposed methodology and the state-of-the-art methodologies, in terms of soft constraint violations. New best results are found for a number of instances for 10 and 60 minutes run.

VIII. FUTURE WORK

We notice that given the same amount of time (300 minutes), shorter runs (10 minutes × 30 runs) are better than longer runs (60 minutes × 5 runs) in obtaining the best results for instances 9, 12 and 13. This implies that the search becomes trapped in the longer runs for these instances. In future, we seek to improve the diversification of the proposed algorithm (ILS) to benefit from longer runs. We may consider devising additional neighbourhood structures, embedding tabu mechanisms and even hybridising the proposed methodology with population-based algorithms, which are often used for their diversification capability. The algorithms we would consider include genetic algorithms and particle swarm optimization.

ACKNOWLEDGMENT

This research work is funded by Universiti Malaysia Sabah (Reference: SGA0040-2019).

REFERENCES

[1] Mohammed Abdelghany, Amr B. Eltawil, Zakaria Yahia, and Kauhide Nakata. A hybrid variable neighbourhood search and dynamic programming approach for the nurse rostering problem. Journal of Industrial and Management Optimization, 17:2051, 2021.
[2] Annmar Abuhamdah, Wadi Boulila, Ghaith M. Jaradat, Anas M. Quteishat, Mutaseem K. Alsmaidi, and Ibrahim A. Almarashdeh. A novel population-based local search for nurse rostering problem. International Journal of Electrical and Computer Engineering, 11:471, 2021.
[3] Mohammed A. Awadallah, Mohammed Azmi Al-Betar, Ahmad Tajudin Khader, Asaju La’aro Bolaji, and Mahmud Alkoffash. Hybridization of harmony search with hill climbing for highly constrained nurse rostering problem. Neural Computing and Applications, 28:463–482, 2017.
[4] T. Barnett, P. Namasiyavam, and D. A. A. Narudin. A critical review of the nursing shortage in malaysia. International nursing review, 57:32–39, 2010.
[5] Burak Bilgin, Peter Demeester, Wim Vancroonenburg, Greet Vanden Berghe, and Tony Wauters. A hyper-heuristic combined with a greedy shuffle approach to the nurse rostering competition. In Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling (PATAT’10), 2010.
[6] Cameron B Browne, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in Games, 4(1):1–43, 2012.
[7] Edmund K Burke and Tim Curtois. An ejection chain method and a branch and price algorithm applied to the instances of the first international nurse rostering competition, 2010. In Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling PATAT, volume 10, page 13, 2010.
[8] Edmund K Burke, Patrick De Causmaecker, Greet Vanden Berghe, and Hendrik Van Landeghem. The state of the art of nurse rostering. Journal of scheduling, 7(6):441–499, 2004.
[9] Sara Ceschia, Nguyen Dang, Patrick De Causmaecker, Stefaan Haspeslagh, and Andrea Schaar. The second international nurse rostering competition. Annals of Operations Research, 274:171–186, 2019.
[10] Sara Ceschia, Rosita Guido, and Andrea Schaar. Solving the static inre-ti nurse rostering problem by simulated annealing based on large neighborhoods. Annals of Operations Research, 288:95–113, 2020.
[11] B. Cheang, H. Li, A. Lim, and B. Rodrigues. Nurse rostering problems—a bibliographic survey. European journal of operational research, 151:447–460, 2003.
[12] Mei Ching Chen, San Nah Sze, Say Leng Goh, Nasser R. Sabar, and Graham Kendall. A survey of university course timetabling problem: Perspectives, trends and opportunities. IEEE Access, 9:106515–106529, 2021.
[13] J. Chou, C. Liang, H. Wu, I. Wu, and T. Wu. A new mcts-based algorithm for multi-objective flexible job shop scheduling problem. In Proc. Conf. Technologies and Applications of Artificial Intelligence (TAAI), pages 136–141, 2015.
[14] Mattia Crippa, Pier Luca Lanzì, and Fabio Marocchi. An analysis of single-player monte carlo tree search performance in sokoban. Expert Systems with Applications, 192:116224, 2022.
[15] Tim Curtois and Rong Qu. Computational results on new staff scheduling benchmark instances. Technical report, ASAP Research Group, School of Computer Science, University of Nottingham, 2014.
[16] Kathryn A. Dowsland. Nurse scheduling with tabu search and strategic oscillation. European journal of operational research, 106:393–407, 1998.
[17] Matthew J. Felicetti and Dianhui Wang. Deep stochastic configuration networks with optimised model and hyper-parameters. Information Sciences, 600:431–441, 2022.
[18] Say Leng Goh, Graham Kendall, and Nasser R. Sabar. Monte carlo tree search in finding feasible solutions for course timetabling problem. International Journal on Advanced Science, Engineering and Information Technology, 9:1936, 2019.
[19] Rafael A. M. Gomes, Tiulo A. M. Toffolo, and Haroldo Gambini Santos. Variable neighborhood search accelerated column generation for the nurse rostering problem. Electronic Notes in Discrete Mathematics, 58:31–38, 2017.
[20] F. Guessoum, S. Haddadi, and E. Gattal. Simple, yet fast and effective two-phase method for nurse rostering. American Journal of Mathematical and Management Sciences, 39:1–19, 2020.
[21] Salim Haddadi. Three-phase method for nurse rostering. International Journal of Management Science and Engineering Management, 14:193–205, 2019.
[22] Stefaan Haspeslagh, Patrick De Causmaecker, Andrea Schaerf, and Martin Stoveik. The first international nurse rostering competition 2010. Annals of Operations Research, 218:221–236, 2014.
[23] Han Huang, Weijia Lin, Zhiyong Lin, Zhiheng Hao, and Andrew Lim. An evolutionary algorithm based on constraint set partitioning for nurse rostering problems. Neural Computing and Applications, 25:703–715, 2014.
PROFESSOR DR. SALWANI ABDULLAH She obtained her BSc in Computer Science from The University of Technology Malaysia and her master’s degree specializing in Computer Science from The National University of Malaysia. She did her Ph.D. in Computer Science at The University of Nottingham, United Kingdom. Now, she is a Professor in Computational Optimization at the Faculty of Information Science and Technology, The National University of Malaysia. Her research interest falls under Artificial Intelligence and Operation Research, particularly computational optimization algorithms (heuristic and meta-heuristic, evolutionary algorithms, local search) that involve different real-world applications for single and multi-objective continuous and combinatorial optimization problems such as timetabling, scheduling, routing, nurse rostering, dynamic optimization, data mining problems (feature selection, clustering, classification, time-series prediction), intrusion detection and search based software testing.

PROFESSOR DR. GRAHAM KENDALL is a Senior Vice President of the PETRA Group and the Chief Executive of the Good Capitalism Forum (GCF). Prior to this, he worked at the University of Nottingham for 21 years and, prior to this, he worked in the IT sector, holding several senior positions.

Prior to joining the PETRA Group, Graham worked in Malaysia for 10 years, being based at the University of Nottingham’s campus in Semenyih where he was the CEO & Provost, and also a Pro-vice Chancellor of the global University.

He is (or has been, prior to leaving academia) a fellow of both the British Computer Society and the Operational Research Society. He holds honorary and distinguished professorial positions at universities in Hong Kong and India. He is an Emeritus Professor at the University of Nottingham.

He is a world recognised academic, with his research focussing on solving real world problems through the use of Operations Research, utilising Artificial Intelligence methods which are based on Charles Darwin’s principles of natural evolution.

He has written over 250 peer reviewed papers and has served on several editorial boards including being the Editor-in-Chief of the IEEE Transactions of Computational Intelligence and AI in Games and an Associate Editor of the Journal of the Operational Research Society.

***