Received 23 May 2022; accepted 21 June 2022, date of publication 24 June 2022, date of current version 6 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3186097

RESEARCH ARTICLE

A 2-Stage Approach for the Nurse Rostering Problem

SAY LENG GOH[1], SAN NAH SZE[2], NASSER R. SABAR[3], SALWANI ABDULLAH[4], AND GRAHAM KENDALL[5]

[1]Optimisation Research Group, Faculty of Computing and Informatics, Universiti Malaysia Sabah Labuan International Campus, Labuan 87000, Malaysia
[2]Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak, Kota Samarahan, Sarawak 94300, Malaysia
[3]Department of Computer Science and Information Technology, La Trobe University, Melbourne, VIC 3086, Australia
[4]Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi, Selangor 43600, Malaysia
[5]School of Computer Science, University of Nottingham, Semenyih, Selangor 43500, Malaysia

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

This work was supported by Universiti Malaysia Sabah under Reference SGA0040-2019.

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.

The associate editor coordinating the review of this manuscript and approving it for publication was Huaqing Li.

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 specified 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 [4]. 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 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 remainder of this paper is organized as follows. The nurse rostering problem and its formal representation are described in Section II. Related work is reviewed in Section III. The proposed methodology is described in Section IV. We present the experimental results in Section V. An in-depth discussion is provided in Section VI. Finally, the conclusion and suggestions for future work are given in Section VII and VIII 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

1) PARAMETERS

N:	set of nurses.
h:	number of days in the planning horizon.
D:	set of days $= \{1 \ldots h\}$.
W:	set of weekends $= \{1 \ldots h/7\}$.
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}^{max}:	max number of shift s that can be assigned to nurse n.
b_n^{min}:	min number of minutes that nurse n must be assigned.
b_n^{max}:	max number of minutes that nurse n can be assigned.
c_n^{min}:	min number of consecutive shifts that nurse n must work.
c_n^{max}:	max number of consecutive shifts that nurse n can work.
o_n^{min}:	min number of consecutive days off that nurse n can be assigned.
a_n^{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}^{min}:	weight for cover below the preferred one for shift s on day d.
w_{ds}^{max}:	weight for cover above the preferred one for shift s on day d.

2) DECISION VARIABLES

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

$$k_{nw} = \begin{cases} 1 & \text{if nurse } n \text{ works on weekend } w \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

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.

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. q_{nds} and p_{nds} in Equations 4 and 5 are weights that show the importance of shift on and off requests to a nurse. Meanwhile,

w_{ds}^{min} and w_{ds}^{max} in Equation 6 are weights that underline the importance of under and over coverage respectively.

$$Minimize \sum_{i=1}^{3} SC_i \tag{3}$$

SC_1: Shift on penalty.

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

SC_2: Shift off penalty.

$$\sum_{n \in N} \sum_{d \in D} p_{nds} \cdot x_{nds} \quad s \in S \tag{5}$$

SC_3: Date specific cover penalty.

$$\sum_{d \in D} w_{ds}^{min} \cdot y_{ds} + \sum_{d \in D} w_{ds}^{max} \cdot z_{ds} \quad s \in S \tag{6}$$

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.

$$\sum_{s \in S} x_{nds} \leq 1 \quad n \in N, d \in D \tag{7}$$

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.

$$x_{nds} + x_{n(d+1)t} \leq 1 \quad n \in N,\ s \in S,\ d \in \{1 \ldots |D| - 1\},\ t \in T_s \tag{8}$$

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.

$$\sum_{d \in D} x_{nds} \leq m_{ns}^{max} \quad n \in N,\ s \in S \tag{9}$$

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

$$b_n^{min} \leq \sum_{d \in D} l_s \cdot x_{nds} \leq b_n^{max} \quad n \in N,\ s \in S \tag{10}$$

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

$$\sum_{j=d}^{d+c_n^{max}} x_{njs} \leq c_n^{max} \quad n \in N, s \in S, d \in \{1 \ldots |D| - c_n^{max}\} \tag{11}$$

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.

$$x_{nds} + \left(s - \sum_{j=d+1}^{d+e} x_{njs}\right) + x_{n(d+e+1)s} > 0$$
$$n \in N, s \in S, e \in \{1 \ldots c_n^{min} - 1\}, d \in \{1 \ldots |D| - (e+1)\} \tag{12}$$

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

$$\left(1 - x_{nds}\right) + \sum_{j=d+1}^{d+e} x_{njs} + \left(1 - x_{n(d+e+1)s}\right) > 0$$
$$n \in N, s \in S, e \in \{1 \ldots o_n^{min} - 1\}, d \in \{1 \ldots |D| - (e+1)\} \tag{13}$$

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

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

$$\sum_{w \in W} k_{nw} \leq a_n^{max} \quad n \in N \tag{15}$$

III. RELATED WORK

An overview of the nurse rostering problem can be found in [8], [11], [50]. 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 method 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.* [5] 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 neghbourhood search [43], hybrid harmony search [3], randomized variable neighbourhood search [49], neutrality 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 [36], 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 annealing [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 neigbourhood 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
                                      // Stage 1
2 if FOUNDFEASIBLESOLUTION(bestSol) then
                                      // Stage 2
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 $n \in N$ do
2      MCTS($n$, bestSol)
3      if $f(bestSol_n) \neq 0$ 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.

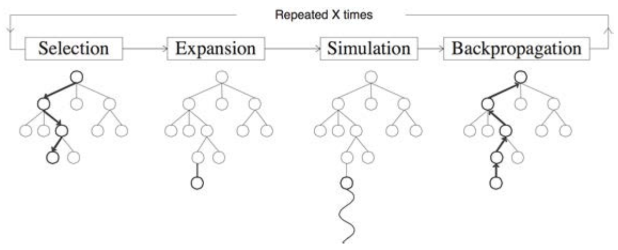


FIGURE 1. MCTS.

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

TABLE 1. Solution methodologies for the NRP benchmark datasets.

Benchmark NRP	Year	Methodology	Reference
INRC-I	2019	Neutrality-based Iterated Local Search	[32]
	2021	Population-based Local Search	[2]
		Hybrid of Variable Neighbourhood Search and Dynamic Programming	[1]
INRC-II	2020	Rotation based Branch-and-price	[28]
		Simulated Annealing	[10]
	2021	Hyper-heuristic based upon a Hidden Markov Model	[25]
Shift Scheduling	2017	Hybrid Integer Programming and Variable Neighbourhood Search	[35]
	2018	Integer Programming	[39]
	2020	First-order Linear Programming in a Column Generation-based Heuristic	[40]
		Hybrid Fix-and-optimize and Simulated Annealing	[44]
ORTEC	2015	Particle Swarm Optimization	[47]
	2017	Hybrid Integer Programming and Variable Neighbourhood Search	[35]
	2019	Simulated Annealing	[27]
NSPLib	2019	Opposition-based Parallel Harmony Search	[48]
		Variable Fixing Heuristic, Iterated Local Search and Mixed Integer Programming	[21]
	2020	Variable Fixing Heuristic and Mixed Integer Programming	[20]

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.

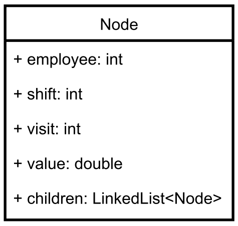


FIGURE 2. Node class definition.

Algorithm 3: MCTS(n, $bestSol$)

```
1  create a root node, $rootNode$
2  $C \leftarrow 5.0$
3  $exploration \leftarrow 0$
4  while $exploration < 200000$ do
5      $curSol_n \leftarrow$ empty solution
6      $d \leftarrow 1$
7      $visitedNode \leftarrow visitedNode \cup rootNode$
8      TREE($rootNode, visitedNode, C, n, d, curSol_n$)
9      $reward \leftarrow$ SIMULATION($n, d, curSol_n, bestSol$)
10     BACKPROPAGATION($reward, visitedNode$)
11     $C \leftarrow C * 0.9999$
12     $exploration++$
13 end
```

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 is 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)

Algorithm 4: TREE($rootNode, visitedNode, C, n, d, curSol_n$)

```
1  $currentNode \leftarrow rootNode$
2  while current$Node$ is not leaf do
3      $currentNode \leftarrow$ SELECTION($currentNode, C$)
4      $visitedNode \leftarrow visitedNode \cup currentNode$
5      $curSol_{nd} \leftarrow currentNode.shift$
6      $d++$
7  end
8  EXPANSION($currentNode, n, d, curSol_n$)
9  $childNode \leftarrow$ select one of $currentNode.children$
     randomly
10 $visitedNode \leftarrow visitedNode \cup childNode$
11 $curSol_{nd} \leftarrow childNode.shift$
12 $d++$
```

and is therefore more efficient.

$$v_i + C\sqrt{\frac{\ln n_p}{n_i}} \tag{16}$$

$$v_i + C \times RANDOM(0, 1) \tag{17}$$

Algorithm 5: SELECTION($currentNode, C$)

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

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* (line 5). If the shift is feasible for a particular day, it will

Algorithm 6: EXPANSION($currentNode, n, d, curSol_n$)

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

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 non-existent 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 [minimum - time unit] (instead of 1). Likewise, if time unit > maximum, the constraint violation is [time unit - maximum] (instead of 1).

$$-(1.0 - \frac{d}{|D|}) \tag{18}$$

$$\frac{1.0}{f(curSol_n) + 1} \tag{19}$$

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.

Algorithm 7: SIMULATION($n, d, curSol_n, bestSol$)

```
while $d \le |D|$ do
    $shifts \leftarrow validShifts(n)$
    do
        if $shifts \neq \emptyset$ then
            $shift \leftarrow$ select a shift probabilistically from $shifts$
        else
            return $-(1 - d/|D|)$
        end
        if FEASIBLE($shift, curSol_{nd-1}, n, d, curSol_n$) then
            $state \leftarrow$ true
        else
            $shifts \leftarrow shifts - shift$
            $state \leftarrow$ false
        end
    while $\neg state$
    $curSol_{nd} \leftarrow shift$
    $shifts \leftarrow \emptyset$
    $d++$
end
if $f(curSol_n) < f(bestSol_n)$ then
    $bestSol_n \leftarrow curSol_n$
end
return $1.0/(f(curSol_n) + 1)$
```

Algorithm 8: BACKPROPAGATION($Reward, visitedNode$)

```
foreach $node \in visitedNode$ do
    $node.visit++$
    $node.value \leftarrow node.value + (reward - node.value)/node.visit$
end
```

Algorithm 9: HC($n, bestSol$)

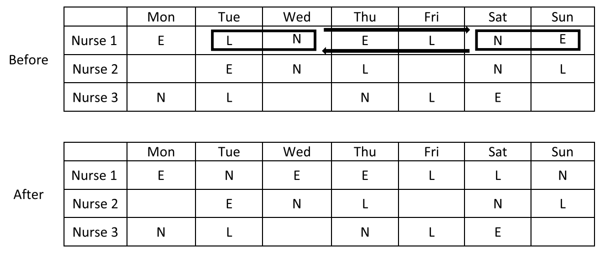
```
current solution, $curSol \leftarrow bestSol$
repeat
    for $d = 1$ to $|D|$ do
        $block \leftarrow$ RANDOM$[1, 4]$
        for $i = d$ to $block$ do
            $canSol_{nd} \leftarrow$ select a shift probabilistically from $validShifts(n)$
        end
        if $f(canSol_n) \le f(curSol_n)$ then
            $curSol_n \leftarrow canSol_n$
            if $f(curSol_n) = 0$ then
                exit outer loop
            end
        else
            replace the original shifts into $curSol_n$
        end
    end
until this end condition
$bestSol \leftarrow curSol$
```

Algorithm 10: ILS($bestSol$)

```
current solution, $curSol \leftarrow bestSol$
repeat
    VND($curSol, bestSol$)
    PERTURBATION($curSol, bestSol$)
until this end condition
```

1) VARIBLE 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 $k = k_{max}$. 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 $O(ik_{max})$ where i is the number of iterations which varies according to the search landscape. The neighbourhood structures applied are:



Before

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Nurse 1	E	L	N	E	L	N	E
Nurse 2		E	N	L		N	L
Nurse 3	N	L		N	L	E	

After

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Nurse 1	E	N	E	E	L	L	N
Nurse 2		E	N	L		N	L
Nurse 3	N	L		N	L	E	

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

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

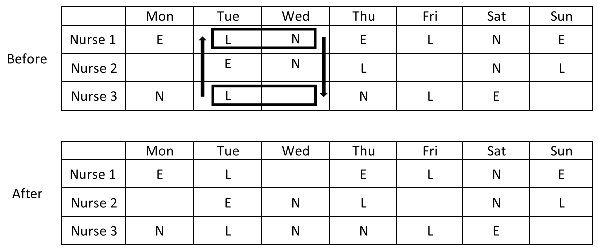
In VND, we set the X variable to a random value in the range of 1 to 4.

2) PERTURBATION

The PERTURBATION procedure is presented in Algorithm 12. *step* and k are initialised to 1 (lines 2 and 3). A record of the current cost is kept as *record* (line 4).

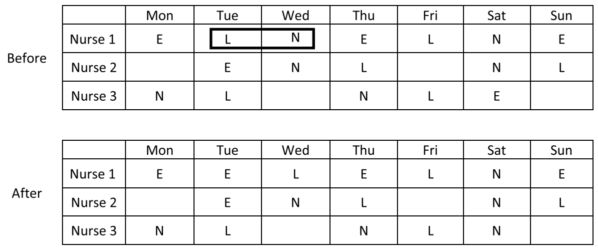
Algorithm 11: VND($curSol$, $bestSol$)

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



		Mon	Tue	Wed	Thu	Fri	Sat	Sun
Before	Nurse 1	E	L	N	E	L	N	E
	Nurse 2		E	N	L		N	L
	Nurse 3	N	L		N	L	E	
After	Nurse 1	E	L		E	L	N	E
	Nurse 2		E	N	L		N	L
	Nurse 3	N	L	N	N	L	E	

FIGURE 4. An example of 2-vertical exchange. E (Early), L (Late) and N (Night) are shifts.



		Mon	Tue	Wed	Thu	Fri	Sat	Sun
Before	Nurse 1	E	L	N	E	L	N	E
	Nurse 2		E	N	L		N	L
	Nurse 3	N	L		N	L	E	
After	Nurse 1	E	E	L	E	L	N	E
	Nurse 2		E	N	L		N	L
	Nurse 3	N	L		N	L	N	

FIGURE 5. An example of 2-extraction. E (Early), L (Late) and N (Night) are shifts.

We attempt to gradually perturb the current solution until the cost changes $\Delta f \geq 0.05$ (5%). In each iteration, the current solution is perturbed by utilising a neighbourhood structure, N_k. The value of k is incremented by 1 and the current solution is perturbed again if $\Delta f < 0.05$. When $k = k_{max}$, *step* is incremented by 1 and k 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

Algorithm 12: PERTURBATION($curSol$, $bestSol$)

```
1  neighbourhood structures, $N_k$, $k = 1, 2 \dots k_{max}$
2  $step \leftarrow 1$
3  $k \leftarrow 1$
4  $record \leftarrow f(curSol)$
5  do
6      PERTURB($curSol$, $bestSol$, $step$, $N_k$, $record$)
7      if $k = k_{max}$ then
8          $step++$
9          $k \leftarrow 1$
10     else
11         $k++$
12     end
13     if $record > f(curSol)$ then
14         $\Delta f \leftarrow \frac{record - f(curSol)}{record}$
15     else
16         $\Delta f \leftarrow \frac{f(curSol) - record}{record}$
17     end
18 while $\Delta f < 0.05$
```

solution if the candidate cost is less than or equal to $record + record * step * 0.01$. 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 $O(|N||D||S|)$.

Algorithm 13: PERTURB($curSol$, $bestSol$, $Step$, N_k, $Record$)

```
1  repeat
2      generate a candidate solution, $canSol \in N_k(curSol)$
3      if $f(canSol) \leq record + record * step * 0.05$ then
4          $curSol \leftarrow canSol$
5          if $f(curSol) < f(bestSol)$ then
6              $bestSol \leftarrow curSol$
7          end
8      end
9  until this end condition
```

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

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

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 equally bad. A low C value promotes search intensification, while a high C value promotes search diversification. It seems a balance is achieved when C is fixed to 5 but it is not sufficient to find even a feasible solution for this instance. On the other hand, our proposed C value with decay rate $\alpha = 0.9999$ allows MCTS and HC to attain 100% feasibility. We perform t-tests to compare the means (HCV) between C value of 5.0 and the proposed C value with decay rate $\alpha = 0.9999$. The p value of 0.000 (less than 0.05) reveals a significant difference between the means for the instance 21. Note that p value cannot be generated for the rest of the instances because the mean and therefore standard deviation for both groups are zero.

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

	Values of C				
Inst.	1.0	5.0	10.0	Proposed	t-test (p value)
13	0.00/0.01	0.00/0.01	0.00/0.02	0.00/0.01	-
14	0.00/0.01	0.00/0.01	0.00/0.01	0.00/0.01	-
15	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	-
16	0.00/0.00	0.00/0.01	0.00/0.02	0.00/0.01	-
17	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	-
18	0.00/0.00	0.00/0.01	0.00/0.00	0.00/0.00	-
19	0.00/0.01	0.00/0.01	0.00/0.01	0.00/0.01	-
20	0.00/0.06	0.00/0.06	0.00/0.07	0.00/0.06	-
21	5673.40/-	2608.87/-	3016.87/-	0.00/15.86	0.000
22	0.00/5.53	0.00/3.67	0.00/4.97	0.00/4.47	-
23	0.00/4.50	0.00/4.39	0.00/5.12	0.00/5.44	-
24	0.00/4.23	0.00/4.15	0.00/3.86	0.00/4.02	-

2) COMPARING TREE POLICIES/NODE EVALUATION FUNCTIONS

In this section, we compare the HCV and time to feasibility using different tree policies in the SELECTION procedure of MCTS. Note that we are using the proposed C value in the previous section due to its effectiveness. The UCB tree policy is shown in Equation 16, with the proposed tree policy is given in Equation 17. As evident in Table 4, the proposed tree policy allows the hybridisation of MCTS and HC to attain feasible solutions in shorter average times for instance 21. The p value of 0.000 (less than 0.05) reveals a significant difference between the means (time to feasibility) for this instance. In addition, a lower overall average time (1.12 vs 1.25) is achieved.

TABLE 4. Mean HCV/mean time to feasibility(s). $n = 30$.

	Tree Policies/ Node Evaluation Functions		
Inst.	UCB	Proposed	t-test (p value)
13	0.00/0.01	0.00/0.01	0.055
14	0.00/0.01	0.00/0.01	1.000
15	0.00/0.00	0.00/0.00	0.561
16	0.00/0.01	0.00/0.02	0.069
17	0.00/0.00	0.00/0.00	0.184
18	0.00/0.00	0.00/0.00	0.281
19	0.00/0.01	0.00/0.01	1.000
20	0.00/0.06	0.00/0.06	0.606
21	0.00/15.86	0.00/12.04	0.000
22	0.00/4.47	0.00/5.52	0.113
23	0.00/5.44	0.00/4.54	0.067
24	0.00/4.02	0.00/4.59	0.128
Avg	0.00/1.25	0.00/1.12	

3) COMPARING SIMULATION AND MCTS

Here, we compare the simulation component of MCTS with the fully fledged MCTS (with the proposed C value and tree policy). As shown in Table 5, simulation alone fails to find a feasible solution for instance 21. The p value of 0.000 (less than 0.05) reveals a significant difference between the means (HCV) of heuristic-based simulation and MCTS for instance 21. This highlights the importance of the learning component (TREE) of MCTS. Note that p value cannot be generated for the rest of the instances because the mean and therefore standard deviation for both groups are zero.

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

Inst.	Heuristic Based Simulation	MCTS	t-test (p value)
13	0.00/0.01	0.00/0.01	-
14	0.00/0.00	0.00/0.01	-
15	0.00/0.00	0.00/0.00	-
16	0.00/0.00	0.00/0.02	-
17	0.00/0.00	0.00/0.00	-
18	0.00/0.00	0.00/0.00	-
19	0.00/0.01	0.00/0.01	-
20	0.00/0.03	0.00/0.06	-
21	2404.20/-	0.00/12.04	0.000
22	0.00/2.14	0.00/5.52	-
23	0.00/5.51	0.00/4.54	-
24	0.00/3.91	0.00/4.59	-

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

		Time to feasibility (s)		
Inst.	Feasibility (%)	Minimum	Mean	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	0.01	0
14	100	0	0.01	0
15	100	0	0.00	0
16	100	0	0.02	0
17	100	0	0.00	0
18	100	0	0.00	0
19	100	0	0.01	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	4.59	8

4) FEASIBILITY

Table 6 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 $\leq$ 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.

B. IMPROVING THE QUALITY OF SOLUTION

1) COMPARING THE INFLUENCE OF NEIGHBOURHOOD STRUCTURES

To observe the influence of the proposed neighbourhood structures, we run each with a combination of neighbourhood 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.

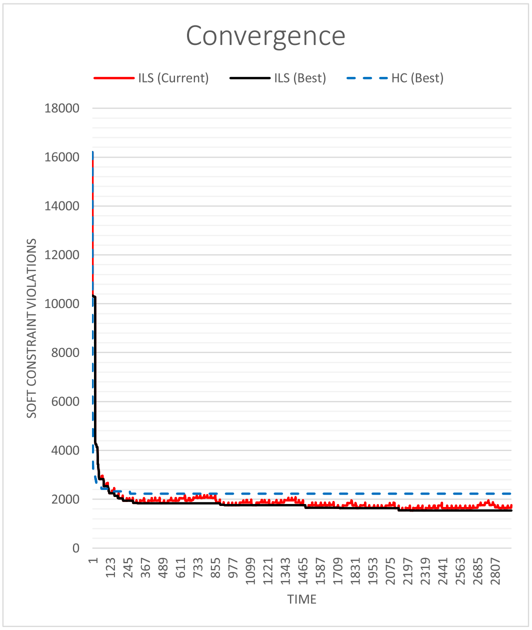


FIGURE 6. Convergence behaviour of the proposed ILS.

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 8). 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 2 Duo

TABLE 7. The influence of the proposed neighbourhood structures. Shown are mean soft constraint violations. $n = 10$.

Inst.	NS1	NS2	NS3	NS1+NS2	NS1+NS3	NS2+NS3	NS1+ NS2+ NS3
1	777.60	1038.80	874.40	**715.00**	731.90	800.10	720.20
2	2414.80	3118.30	1399.20	2940.30	**1204.30**	1305.70	1256.40
3	2016.50	1891.30	1794.30	1680.10	1418.80	1434.80	**1232.90**
4	4891.90	4582.40	2429.30	4409.70	2102.20	2322.50	**2038.50**
5	2283.90	2550.80	2593.90	2357.30	1923.40	2183.30	**1645.50**
6	8208.50	8662.00	3924.70	8030.00	2974.30	3372.90	**2715.50**
7	4337.40	4757.10	3063.60	4292.30	1698.20	2165.10	**1576.90**
8	11537.10	14738.20	5112.10	11065.30	2524.70	4641.30	**2353.20**
9	6357.90	7969.30	1788.30	5780.30	541.90	547.10	**496.10**
10	18750.40	18250.30	6882.30	18505.30	5652.30	6236.80	**5374.00**
11	29541.80	29452.90	5878.80	29369.40	**3558.30**	5345.70	3577.50
12	30323.80	30119.70	9749.20	31339.50	5539.10	7368.70	**4787.50**
13	50391.60	51837.40	11058.80	50345.70	3444.90	6887.60	**3032.70**
14	8641.60	11734.10	3647.50	7844.80	**1794.50**	3365.50	1799.80
15	11854.50	16948.60	8122.40	11793.70	5658.30	7687.20	**5627.00**
16	18706.70	19483.40	6754.30	18945.80	**3970.90**	6852.70	4127.60
17	17685.90	30008.50	9707.20	18111.50	6848.50	9015.20	**6817.20**
18	30452.40	30916.40	9903.80	31547.40	6040.80	9355.20	**6012.50**
19	28140.10	36955.30	14391.10	27911.90	5224.60	12658.90	**5165.10**
20	174636.40	175591.60	28218.30	174590.50	**7754.90**	21816.60	7894.70
21	192546.20	193780.80	62591.00	191815.00	**22861.40**	48828.30	23614.10
22	138938.20	200587.10	94132.40	141193.50	**47745.00**	87284.70	49206.60
23	455091.90	520697.40	108020.80	451236.40	**25497.60**	88850.00	27521.20
24	901692.90	1051520.20	571584.40	901820.80	**88406.20**	546251.60	93978.40

TABLE 8. Solver details.

Solver	Reference	Methodology
A	Curtois and Qu [15]	Branch & Price
B	Curtois and Qu [15]	Gurobi
C	Curtois and Qu [15]	Ejection Chain
D	Rahimian et al. [35]	Hybrid (IP+VNS)
E	Turhan and Bilgen [44]	Hybrid (MIP heuristics + SA)
F	Smet [39]	Integer Programming
G	Strandmark et al. [40]	LP based on column generation heuristic

3.16 GHz and 8 GB RAM) computer. The solver D (Hybrid IP+VNS) was performed on a (Intel Core i5 3.40 GHz and 4GB RAM) computer. Meanwhile, the solver E (Hybrid MIP heuristics+SA) was run on a (Intel Core i3 2.27 GHz and 3GB RAM). Branch & Price and Gurobi seem to perform well on small instances. Both the hybrids (solver D and E) worked effectively for small and medium sized instances but was lacking in terms of performance for large instances. Our proposed methodology produces strong results for large instances. We believe it is due to its ability in finding a feasible solution faster and it therefore has more time to focus on improving the solution quality. However, its performance is mediocre for certain small and medium instances where an optimal solution is not guaranteed unlike the exact method such as Gurobi (integer programming). Note that the solver E did not include instances 20-24 in their experiments as they thought the data (planning horizon of six months or one year) is not practical in the real world application.

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 hinders an objective comparison.

TABLE 9. Comparison of MCTS + ILS with the state of the art methods with runtime of 10 minutes. Shown are best (mean) results in terms of soft constraint violations. $n = 30$.

Inst.	A	B	C	D	E	MCTS + ILS
1	**607**	**607**	**607**	**607**	**607**	**607** (607.00)
2	**828**	**828**	923	**828**	**828**	**828** (828.43)
3	**1001**	**1001**	1003	**1001**	**1001**	1002 (1005.10)
4	**1716**	**1716**	1719	**1716**	**1716**	**1716** (1719.83)
5	1160	**1143**	1439	**1143**	**1143**	1155 (1282.37)
6	1952	**1950**	2344	**1950**	**1950**	2054 (2134.83)
7	1058	**1056**	1284	**1056**	**1056**	1080 (1128.13)
8	1308	8995	2529	1364	**1341**	1374 (1522.23)
9	**439**	**439**	474	**439**	**439**	488 (502.13)
10	**4631**	**4631**	4999	**4631**	**4631**	4664 (4686.13)
11	**3443**	**3443**	3967	**3443**	**3443**	3459 (3481.67)
12	4046	4045	5611	**4042**	4044	4161(4238.80)
13	-	500410	8707	**3109**	3200	3201 (3390.13)
14	-	1482	2542	**1281**	1295	1333 (1396.07)
15	-	78144	6049	**4144**	4420	4490 (4654.83)
16	3323	3521	4343	3306	**3253**	3453 (3613.00)
17	-	6149	7835	**5760**	6138	6012 (6127.73)
18	-	7950	6404	5049	5000	**4958** (5084.83)
19	-	29968	6522	3974	3809	**3635** (3809.40)
20	-	-	23531	5242	-	**5137** (5393.37)
21	-	-	38294	26977	-	**21833** (21952.57)
22	-	-	-	130107	-	**37227** (38494.33)
23	-	-	-	40543	-	**19926** (20448.87)
24	-	-	-	2829680	-	**62387** (64372.13)

VI. DISCUSSION

In MCTS, feasibility is tested before assigning a shift to a nurse on a particular day. All the hard constraints (HC3, 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 vacancy is set 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,

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. $n = 5$.

Inst.	A	B	C	D	E	MCTS + ILS
1	**607**	**607**	**607**	**607**	**607**	**607** (607.00)
2	**828**	**828**	837	**828**	**828**	**828** (828.00)
3	**1001**	**1002**	1003	**1001**	**1001**	**1001** (1002.60)
4	**1716**	**1716**	1718	**1716**	**1716**	**1716** (1719.00)
5	1160	**1143**	1358	**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**	3457 (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	**3225**	**3225**	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)

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 an 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 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 optimums. 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 $\times$ 30 runs) are better than longer runs (60 minutes $\times$ 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.

REFERENCES

[1] M. Abdelghany, A. B. Eltawil, Z. Yahia, and K. Nakata, "A hybrid variable neighbourhood search and dynamic programming approach for the nurse rostering problem," *J. Ind. Manage. Optim.*, vol. 17, no. 4, p. 2051, 2021.

[2] A. Abuhamdah, W. Boulila, G. M. Jaradat, A. M. Quteishat, M. K. Alsmadi, and I. A. Almarashdeh, "A novel population-based local search for nurse rostering problem," *Int. J. Electr. Comput. Eng. (IJECE)*, vol. 11, no. 1, p. 471, Feb. 2021.

[3] M. A. Awadallah, M. A. Al-Betar, A. T. Khader, A. L. Bolaji, and M. Alkoffash, "Hybridization of harmony search with Hill climbing for highly constrained nurse rostering problem," *Neural Comput. Appl.*, vol. 28, no. 3, pp. 463–482, Mar. 2017.

[4] T. Barnett, P. Namasivayam, and D. A. A. Narudin, "A critical review of the nursing shortage in Malaysia," *Int. Nursing Rev.*, vol. 57, no. 1, pp. 32–39, 2016.

[5] B. Bilgin, P. Demeester, W. Vancroonenburg, G. V. Berghe, and T. Wauters, "A hyper-heuristic combined with a greedy shuffle approach to the nurse rostering competition," in *Proc. 8th Int. Conf. Pract. Theory Automated Timetabling (PATAT)*, 2010, pp. 1–6.

[6] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of Monte Carlo tree search methods," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 1, pp. 1–43, Mar. 2012.

[7] E. K. Burke and T. Curtois, "An ejection chain method and a branch and price algorithm applied to the instances of the first international nurse rostering competition, 2010," in *Proc. 8th Int. Conf. Pract. Theory Automated Timetabling (PATAT)*, vol. 10, 2010, p. 13.

[8] E. K. Burke, P. De Causmaecker, G. V. Berghe, and H. Van Landeghem, "The state of the art of nurse rostering," *J. Scheduling*, vol. 7, no. 6, pp. 441–499, Nov. 2004.

[9] S. Ceschia, N. Dang, P. De Causmaecker, S. Haspeslagh, and A. Schaerf, "The second international nurse rostering competition," *Ann. Oper. Res.*, vol. 274, nos. 1–2, pp. 171–186, Mar. 2019.

[10] S. Ceschia, R. Guido, and A. Schaerf, "Solving the static INRC-II nurse rostering problem by simulated annealing based on large neighborhoods," *Ann. Oper. Res.*, vol. 288, no. 1, pp. 95–113, May 2020.

[11] B. Cheang, H. Li, A. Lim, and B. Rodrigues, "Nurse rostering problems—A bibliographic survey," *Eur. J. Oper. Res.*, vol. 151, pp. 447–460, Dec. 2003.

[12] M. C. Chen, S. N. Sze, S. L. Goh, N. R. Sabar, and G. Kendall, "A survey of university course timetabling problem: Perspectives, trends and opportunities," *IEEE Access*, vol. 9, pp. 106515–106529, 2021.

[13] J.-J. Chou, C.-C. Liang, H.-C. Wu, I.-C. Wu, and T.-Y. Wu, "A new MCTS-based algorithm for multi-objective flexible job shop scheduling problem," in *Proc. Conf. Technol. Appl. Artif. Intell. (TAAI)*, Nov. 2015, pp. 136–141.

[14] M. Crippa, P. L. Lanzi, and F. Marocchi, "An analysis of single-player Monte Carlo tree search performance in sokoban," *Expert Syst. Appl.*, vol. 192, Apr. 2022, Art. no. 116224.

[15] T. Curtois and R. Qu, "Computational results on new staff scheduling benchmark instances," ASAP Res. Group, School Comput. Sci., Univ. Nottingham, Nottingham, U.K., Tech. Rep., 2014.

[16] K. A. Dowsland, "Nurse scheduling with Tabu search and strategic oscillation," *Eur. J. Oper. Res.*, vol. 106, nos. 2–3, pp. 393–407, Apr. 1998.

[17] M. J. Felicetti and D. Wang, "Deep stochastic configuration networks with optimised model and hyper-parameters," *Inf. Sci.*, vol. 600, pp. 431–441, Jul. 2022.

[18] S. L. Goh, G. Kendall, and N. R. Sabar, "Monte Carlo tree search in finding feasible solutions for course timetabling problem," *Int. J. Adv. Sci., Eng. Inf. Technol.*, vol. 9, no. 6, p. 1936, Dec. 2019.

[19] R. A. M. Gomes, T. A. M. Toffolo, and H. G. Santos, "Variable neighborhood search accelerated column generation for the nurse rostering problem," *Electron. Notes Discrete Math.*, vol. 58, pp. 31–38, Apr. 2017.

[20] F. Guessoum, S. Haddadi, and E. Gattal, "Simple, yet fast and effective two-phase method for nurse rostering," *Amer. J. Math. Manage. Sci.*, vol. 39, no. 1, pp. 1–19, Jan. 2020.

[21] S. Haddadi, "Three-phase method for nurse rostering," *Int. J. Manage. Sci. Eng. Manage.*, vol. 14, no. 3, pp. 193–205, Jul. 2019.

[22] S. Haspeslagh, P. De Causmaecker, A. Schaerf, and M. Stølevik, "The first international nurse rostering competition 2010," *Ann. Oper. Res.*, vol. 218, no. 1, pp. 221–236, Jul. 2014.

[23] H. Huang, W. Lin, Z. Lin, Z. Hao, and A. Lim, "An evolutionary algorithm based on constraint set partitioning for nurse rostering problems," *Neural Comput. Appl.*, vol. 25, nos. 3–4, pp. 703–715, Sep. 2014.

[24] D. S. Johnson, "The NP-completeness column: An ongoing guide," *J. Algorithms*, vol. 6, no. 1, pp. 145–159, Mar. 1985.

[25] A. Kheiri, A. Gretsista, E. Keedwell, G. Lulli, M. G. Epitropakis, and E. K. Burke, "A hyper-heuristic approach based upon a hidden Markov model for the multi-stage nurse rostering problem," *Comput. Oper. Res.*, vol. 130, Jun. 2021, Art. no. 105221.

[26] A. Kheiri and E. Keedwell, "A sequence-based selection hyper-heuristic utilising a hidden Markov model," in *Proc. Annu. Conf. Genetic Evol. Comput.*, Jul. 2015, pp. 417–424.

[27] F. Knust and L. Xie, "Simulated annealing approach to nurse rostering benchmark and real-world instances," *Ann. Oper. Res.*, vol. 272, nos. 1–2, pp. 187–216, Jan. 2019.

[28] A. Legrain, J. Omer, and S. Rosat, "A rotation-based branch-and-price approach for the nurse scheduling problem," *Math. Program. Comput.*, vol. 12, no. 3, pp. 417–450, Sep. 2020.

[29] R. M. Lusby, J. Larsen, and S. Bull, "A survey on robustness in railway planning," *Eur. J. Oper. Res.*, vol. 266, no. 1, pp. 1–15, Apr. 2018.

[30] G. Manikandan and S. Abirami, "An efficient feature selection framework based on information theory for high dimensional data," *Appl. Soft Comput.*, vol. 111, Nov. 2021, Art. no. 107729.

[31] S. Matsumoto, N. Hirosue, K. Itonaga, N. Ueno, and H. Ishii, "Monte-Carlo tree search for a reentrant scheduling problem," in *Proc. 40th Int. Conf. Comput. Indutrial Eng.*, Jul. 2010, pp. 1–6.

[32] D. Meignan and S. Knust, "A neutrality-based iterated local search for shift scheduling optimization and interactive reoptimization," *Eur. J. Oper. Res.*, vol. 279, no. 2, pp. 320–334, Dec. 2019.

[33] K. Nonobe, "INRC2010: An approach using a general constraint optimization solver," in *Proc. 1st Int. Nurse Rostering Competition (INRC)*, 2010, pp. 1–2.

[34] D. Perez, E. J. Powley, and D. Whitehouse, "Solving the physical traveling salesman problem: Tree search and macro actions," *IIEEE Trans. Comput. Intell. AI Games*, vol. 6, no. 1, pp. 31–45, Mar. 2014.

[35] E. Rahimian, K. Akartunali, and J. Levine, "A hybrid integer programming and variable neighbourhood search algorithm to solve nurse rostering problems," *Eur. J. Oper. Res.*, vol. 258, no. 2, pp. 411–423, Apr. 2017.

[36] M. Römer and T. Mellouli, "A direct MILP approach based on state-expanded network flows and anticipation for multi-stage nurse rostering under uncertainty," in *Proc. 11th Int. Conf. Pract. Theory Automated Timetabling*, 2016, pp. 549–551.

[37] T. P. Runarsson, M. Schoenauer, and M. Sebag, "Pilot, rollout and Monte Carlo tree search methods for job shop scheduling," in *Proc. Int. Conf. Learn. Intell. Optim.* Berlin, Germany: Springer, 2012, pp. 160–174.

[38] N. R. Sabar, S. L. Goh, A. Turky, and G. Kendall, "Population-based iterated local search approach for dynamic vehicle routing problems," *IEEE Trans. Autom. Sci. Eng.*, early access, Aug. 13, 2021, doi: 10.1109/TASE.2021.3097778.

[39] P. Smet, "Constraint reformulation for nurse rostering problems," in *Proc. 12th Int. Conf. Pract. Theory Automated Timetabling*, 2018, pp. 69–80.

[40] P. Strandmark, Y. Qu, and T. Curtois, "First-order linear programming in a column generation-based heuristic approach to the nurse rostering problem," *Comput. Oper. Res.*, vol. 120, Aug. 2020, Art. no. 104945.

[41] J. S. Tan, S. L. Goh, G. Kendall, and N. R. Sabar, "A survey of the state-of-the-art of optimisation methodologies in school timetabling problems," *Expert Syst. Appl.*, vol. 165, Mar. 2021, Art. no. 113943.

[42] J. S. Tan, S. L. Goh, S. Sura, G. Kendall, and N. R. Sabar, "Hybrid particle swarm optimization with particle elimination for the high school timetabling problem," *Evol. Intell.*, vol. 14, no. 4, pp. 1915–1930, Dec. 2021.

[43] I. X. Tassopoulos, I. P. Solos, and G. N. Beligiannis, "A two-phase adaptive variable neighborhood approach for nurse rostering," *Comput. Oper. Res.*, vol. 60, pp. 150–169, Aug. 2015.
[44] A. M. Turhan and B. Bilgen, "A hybrid fix-and-optimize and simulated annealing approaches for nurse rostering problem," *Comput. Ind. Eng.*, vol. 145, Jul. 2020, Art. no. 106531.
[45] C. Valouxis, C. Gogos, G. Goulas, P. Alefragis, and E. Housos, "A systematic two phase approach for the nurse rostering problem," *Eur. J. Oper. Res.*, vol. 219, no. 2, pp. 425–433, Jun. 2012.
[46] J. Van den Bergh, J. Beliën, P. De Bruecker, E. Demeulemeester, and L. De Boeck, "Personnel scheduling: A literature review," *Eur. J. Oper. Res.*, vol. 226, no. 3, pp. 367–385, 2013.
[47] T.-H. Wu, J.-Y. Yeh, and Y.-M. Lee, "A particle swarm optimization approach with refinement procedure for nurse rostering problem," *Comput. Oper. Res.*, vol. 54, pp. 52–63, Feb. 2015.
[48] E. Cetin Yagmur and A. Sarucan, "Nurse scheduling with opposition-based parallel harmony search algorithm," *J. Intell. Syst.*, vol. 28, no. 4, pp. 633–647, Sep. 2019.
[49] Z. Zheng, X. Liu, and X. Gong, "A simple randomized variable neighbourhood search for nurse rostering," *Comput. Ind. Eng.*, vol. 110, pp. 165–174, Aug. 2017.
[50] C. M. Ngoo, S. L. Goh, S. N. Sze, N. R. Sabar, S. Abdullah, and G. Kendall, "A survey of the nurse rostering solution methodologies: The state-of-the-art and emerging trends," *IEEE Access*, vol. 10, pp. 56504–56524, 2022, doi: 10.1109/ACCESS.2022.3177280.

SAY LENG GOH received the B.I.T. degree (Hons.) from Universiti Malaysia Sabah, Malaysia, the M.Sc. degree from Imperial College London, and the Ph.D. degree from the University of Nottingham. He is an Academic with the Faculty of Computing and Informatics, Universiti Malaysia Sabah. He has published in various top-tier journals, such as *European Journal of Operational Research*, *Journal of the Operational Research Society*, and *Expert Systems With Applications*. His current research interests include artificial intelligence, operational research, discrete optimization, meta-heuristics, timetabling, and scheduling.

SAN NAH SZE received the Doctor of Philosophy (Ph.D.) degree from The University of Sydney, in 2011. She has more than ten years of experience in research and development. She is currently working as a Senior Lecturer with the Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak (UNIMAS). She had presented at numerous conferences and published her work in ISI and Scopus publications. Her research interests include educational timetabling, vehicle routing, heuristics, and meta-heuristics.

NASSER R. SABAR is a Lecturer with the Computer Science and IT Department, La Trobe University, Australia. He has published over 76 papers in international journals and peer-reviewed conferences. His current research interests include hyper-heuristic frameworks, evolutionary computation, and hybrid algorithms with a specific interest in big data optimization problems, cloud computing, dynamic optimization, and scheduling problems.

SALWANI ABDULLAH received the B.Sc. degree in computer science from the University of Technology Malaysia, the master's degree specializing in computer science from the National University of Malaysia, and the Ph.D. degree in computer science from the University of Nottingham, U.K. Currently, she is a Professor in computational optimization with the Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, National University of Malaysia. Her research interests include artificial intelligence and operation research, particularly computational optimization algorithms (heuristic and meta-heuristic, evolutionary algorithms, and 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, and time-series prediction), intrusion detection, and search-based software testing.

GRAHAM KENDALL is an Emeritus Professor at the University of Nottingham. He is the 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, he worked in Malaysia for ten years, being based at the University of Nottingham's campus in Semenyih where he was the CEO and a 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 a world recognized academic, with his research focusing on solving real world problems through the use of operations research, utilizing 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 for the IEEE Transactions on Computational Intelligence and AI in Games and an Associate Editor for the *Journal of the Operational Research Society*.