Multi-agent Deep Q-Network-based Metaheuristic Algorithm for Nurse Rostering Problem

Xinzhi Zhang[a], Yeming Yang[a], Qingling Zhu[b], Qiuzhen Lin[a,*], Weineng Chen[c], Jianqiang Li[b] and Carlos A. Coello Coello[d,e]

[a]College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China
[b]National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University, Shenzhen, China
[c]School of Computers, South China University of Technology, Guangzhou 510006, China
[d]CINVESTAV-IPN, Department of Computer Science, Mexico, D.F. 07360, Mexico
[e]Basque Center for Applied Mathematics (BCAM) Ikerbasque, Spain

ARTICLE INFO

Keywords:
Multi-agent
Deep Q-Network
Metaheuristic
Nurse Rostering Problem

ABSTRACT

The Nurse Rostering Problem (NRP) aims to create an efficient and fair work schedule that balances both the needs of employees and the requirements of hospital operations. Traditional local search-based metaheuristic algorithms, such as adaptive neighborhood search (ANS) and variable neighborhood descent (VND), mainly focus on optimizing the current solution without considering potential long-term consequences, which may easily get stuck in local optima and limit the overall performance. Thus, we propose a multi-agent deep Q-network-based metaheuristic algorithm (MDQN-MA) for NRP to harness the strengths of various metaheuristics. Each agent encapsulates a metaheuristic algorithm, where its available actions represent different perspectives of the problem environment. By combining their strengths and various perspectives, these agents can work collaboratively to navigate and search for a broader range of potential solutions effectively. Furthermore, to improve the performance of an individual agent, we model its neighborhood search as a Markov model and integrate a trained deep Q-network to consider long-term impacts for its neighborhood sequential decision-making. The experimental results clearly show that an individual agent in MDQN-MA can outperform ANS and VND, and multiple agents in MDQN-MA even perform better, achieving the best results among metaheuristic algorithms on the Second International Nurse Rostering Competition dataset.

1. Introduction

The Nurse Rostering Problem (NRP) is a classical combinatorial optimization problem (COP), which aims to optimize the work schedules of nurses [1]. Scientific and reasonable nurse scheduling needs to consider both the medical service demands of the hospital and the individual needs of the nurses. Moreover, well-planned scheduling also contributes to providing high-quality healthcare services for patients [2]. However, due to various constraints such as restricted working hours and the preferences of nurses, solving the NRP is a challenging task that has been proven to be NP-hard. Over the past few decades, extensive research has been dedicated to tackling the NRP, resulting in two primary solution categories: exact algorithms [3–6] and heuristic algorithms [7–10].

Exact algorithms employ mathematical techniques such as integer programming [11] and branch-and-price [12] to swiftly produce optimal or near-optimal solutions, which are particularly effective for small-scale NRP instances. However, their computational complexity becomes a limitation as the problem size increases, making them inefficient for a large number of nurses [13], which has spurred the development of heuristic algorithms. In contrast, heuristic algorithms utilize optimization strategies based on intuition or experience to find near-optimal solutions, which are well-suited for tackling large-scale NRP instances. Individual-based algorithms such as adaptive neighborhood search (ANS) [14–16], and variable neighborhood descent (VND) [17–20] often outperform population-based algorithms such as genetic algorithm (GA) [21] and particle swarm optimization (PSO) [22] because they can exploit the search space efficiently.

However, the traditional heuristics focus mainly on immediate solution improvements during neighborhood selection for local search, potentially leading to getting stuck in local optima due to their 'myopic' view. To address the limitations, this paper explores the potential of reinforcement learning (RL), which can consider long-term returns in tackling the NRP. The agent we proposed employs a metaheuristic algorithm that models the neighborhood selection process as a Markov decision process (MDP) and leverages the Deep Q-Network (DQN) for neighborhood selection decision-making. This method aims to generate promising solutions with sustained performance improvements. In addition, it is difficult for traditional single heuristic algorithms to comprehensively explore the solution space, while the multi-agent framework allows collaboration between different heuristic algorithms. This structure makes the system more adaptable to different characteristics of the problem, thereby improving the robustness and applicability of the algorithm. Furthermore, this paper proposes a multi-agent deep Q-network-based metaheuristic algorithm (MDQN-MA) for NRP to

---

*Corresponding author
ORCID(s): `https://orcid.org/0000-0003-2415-0401` (Q. Lin)

harness the collective strengths of multiple agents. The main contributions of this paper can be summarized as follows:

1. This paper introduces a novel MDP-based model of the neighborhood selection process in local search for solving the NRP, enabling the first attempt to apply the DQN to the neighborhood search process. The DQN can learn from past experiences and make decisions that maximize long-term rewards, enhancing the efficiency of solution exploration.

2. This paper is the first attempt to employ a multi-agent framework to tackle the NRP. Multiple agents collaborate in this framework for a more comprehensive exploration and exploitation of the solution space.

3. This paper conducts extensive experiments on the Second International Nurse Rostering Competition (INRC-II) dataset to validate the effectiveness and efficiency of MDQN-MA. The results demonstrate that MQDN-MA outperforms the performance of individual agents and the state-of-the-art metaheuristic algorithms [23–25], showing promising potential for improving upon previous methods for tackling the NRP.

The remainder of this paper is organized as follows: **Section** 2 provides an overview of the related work on the NRP and multi-agent systems for combination optimization problems. **Section** 3 introduces the mathematical model of NRP in the INRC-II, which serves as the foundation of this paper. **Section** 4 presents a detailed introduction of our proposed MDQN-MA. In **Section** 5, we perform an in-depth analysis and discussion of the experimental results obtained from comparative experiments. Finally, **Section** 6 highlights the contributions of our study and suggests future research directions.

2. Related work

In this section, we provide a concise overview of the research on NRP over the past two decades, with a specific focus on metaheuristic algorithms. We elaborate on the motivations for applying the DQN in the neighborhood selection process, which aims to overcome the limitations of traditional metaheuristic algorithms. Then we list the applications of multi-agent in the field of combinatorial optimization and elaborate on the significance of applying a multi-agent framework to NRP.

2.1. Methods for nurse rostering problem

The NRP represents a complex combinatorial optimization problem with great practical significance that involves generating high-quality schedules for nurses [3]. It has attracted widespread attention since it was proposed in the 1960s, and especially in the past decade, two international competitions [26, 27] further stimulated the interest of researchers. Various algorithms have been devised to tackle the NRP, broadly categorized into two groups: exact algorithms [3–6] and heuristic algorithms [7–10]. In recent years, hybrid algorithms [28–30] that combine the strengths of various methods, have gained prominence. Furthermore, A new trend in solving the NRP is to combine it with RL to take advantage of the power of RL to optimize the scheduling processes [31].

Exact algorithms typically employ mathematical optimization techniques such as integer programming, branch-and-price, and column generation. For example, Zenda et al. [11] applied integer linear programming to solve a real-world NRP at an Italian hospital, utilizing the CPLEX [32] solver. Burke and Curtois [33] proposed a Branch-and-Price algorithm to address the NRP, which achieved competitive results on the INRC-I dataset. These exact approaches often yield optimal solutions for small-scale instances but exhibit computational complexity challenges with larger problem sizes. In contrast, heuristic algorithms can find near-optimal solutions in a relatively short time. Consequently, heuristic algorithms have gained popularity for addressing large-scale NRPs.

Heuristic algorithms, inspired by natural processes, can be divided into two major categories: population-based algorithms and individual-based algorithms. Examples of population-based algorithms include genetic algorithms (GA) [21], particle swarm optimization (PSO) [22] and ant colony optimization (ACO) [34]. These algorithms employ mechanisms such as selection, crossover, mutation, and adaptation to guide the search process and drive the population toward promising regions of the search space. Simulated annealing [35, 36], tabu search [37], adaptive neighborhood search (ANS) [14, 25] and iterative local search [38], which are individual-based algorithms, are frequently applied to NRP. These algorithms iteratively explore solution spaces, utilizing neighborhood operators for solution improvement and modification. For instance, The Hust. Smart group [25] achieved remarkable results by combining the ANS and Tabu search in the INRC-II. Wickert et al. [39] compared VND with exact methods for INRC-II, revealing VND's superiority in handling large-scale instances.

To tackle the NRP more effectively, researchers have been exploring hybrid algorithms that aim to leverage the strengths of various algorithms. A hybrid algorithm that combines integer programming (IP) and the variable neighborhood search (VNS) was proposed in [29], outperforming the state-of-the-art algorithms in NRP. It starts with a greedy heuristic for the initial solution and then refines it using VNS and IP within a ruin-and-recreate framework.

2.2. RL-based multi-agent systems for COP

However, the popular heuristics mentioned above may fall into suboptimal results due to the lack of global information. If we can improve the "short-sightedness" problem, the effectiveness of the algorithm can be further improved. To overcome these limitations, many researchers explore the potential of RL in addressing NRP. RL aims to maximize cumulative rewards through trial and error, emphasizing long-term optimization [40]. When a problem can be formulated as a sequential decision-making task, involving well-defined states, actions, and rewards, RL represents a promising approach. RL has been applied in various combinatorial optimization problems [41], including the vehicle routing problem [42–44], satellite scheduling problem [45], and job shop scheduling [46–49].

To fully exploit the advantages of different metaheuristic algorithms, RL-based multi-agent systems (RL-MAS) have been proposed to solve the COP due to their potential to address complex problems by coordinating the actions of multiple agents [50]. This system utilizes a multi-agent framework with meta-heuristic optimization, and agents use the concept of RL to modify their actions, achieving competitive results in multiple COP. This method has been successfully applied in many fields, including multi-agent path finding [51], resource-constrained project scheduling problem [52] and vehicle routing problem [53], which validates that embedding a single metaheuristic agent into MAS is an effective strategy for solving COPs. Furthermore, many multi-agent frameworks that integrate multiple metaheuristic algorithms have been proposed in many papers [54, 55], which validate that this method is promising for tackling the NRP.

3. Problem formulation

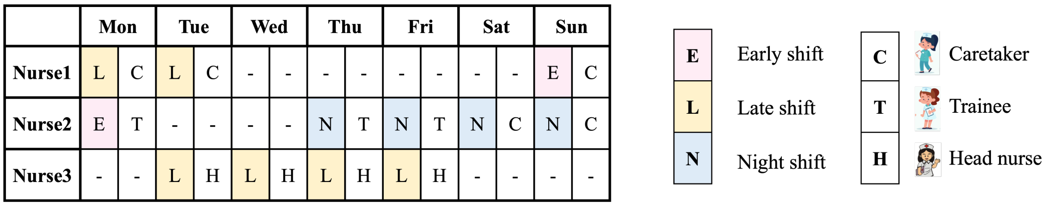
The NRP is a complex combinatorial optimization challenge, which involves allocating qualified nurses with appropriate skills for shifts over a certain period of time, considering a set of constraints. The solution to this problem is an effective nurse shift schedule, as shown in Fig. 1. The rows of the schedule represent different nurses, and the columns represent a specific period, usually a week. In Figure 1, the letter codes E, L, and N are used to represent early shift, late shift, and night shift respectively, while C, T, and H are used to identify caretakers, trainees, and head nurses, respectively. In the real world, due to the varying demands and limitations of different hospitals, NRP lacks a standardized problem definition and consistent examples. To address this issue, the International Nurse Rostering Competition (INRC) was established. The INRC-I was the first competition to provide a standardized dataset and benchmark for NRP. Since hospitals have uncertain demands for nurses, long-term shift schedules can easily need to be readjusted due to changes in workload. The INRC-II divided the entire scheduling period into multiple shorter stages, recognizing the complexities of long-term scheduling with uncertainties. When scheduling the current stage, the personnel needs of subsequent stages cannot be known, and it is impossible to alter the schedule of previous stages. In addition, balancing the workload of all nurses still needs to be considered throughout the entire time. The overall flow chart is shown in Fig. 2.

There are three types of input files: scenario, history, and week data.

- **Scenario**: Information that is global to all weeks of the entire planning horizon, such as nurse contracts and shift types.
- **History**: Information that must be passed from one week to the other to properly compute constraint violations. It includes border information and global counters.
- **Week data**: Specific data from a single week, such as daily coverage requirements and nurse preferences for specific days.

For the initial stage, labeled as $Stage0$, the history file is provided by the competition organizers. However, for subsequent stages, the history file is generated by the simulator based on the optimal solution obtained in the previous week, along with any custom files created to pass information to the next stage. The constraints are categorized into two types: hard constraints that must be satisfied, and soft constraints that can be violated but with a penalty. The solution must satisfy all the hard constraints to be considered feasible and should aim to minimize the weighted sum of soft constraint violations. To fully elaborate on this problem, we provide the mathematical formulation in the following.

- **Scenario Information**: which is global to all weeks (stages).
  - The list of nurses $N = \{n_1, \dots, n_{|N|}\}$.
    - n_1 represents the nurse with index 1.
    - $|N|$ nurses in total.
  - The list of weeks $W = \{w_1, \dots, w_{|W|}\}$.
    - w_1 represents the first week.
    - $|W|$ weeks in total.
    - Make the schedules for w_1 to w_n in turn.
  - The list of days $D = \{d_1, \dots, d_{|D|}\}$, $|D| = 7 \times |W|$.
  - The list of shifts $S = \{s_1, \dots, s_{|S|}\}$.
    - s_1 represents the shift with index 1, i.e., Early.
    - $|S|$ shifts in total.
    - G_s^{min}, G_s^{max}: The minimum and the maximum numbers of consecutive assignments for shift type s.
    - F: A list of forbidden shift type successions.
  - The list of skills $K = \{k_1, \dots, k_{|K|}\}$.
    - k_1 represents the skill with index 1, i.e., Trainee.
    - $|K|$ skills in total.
    - Each nurse $n \in N$ is associated with a set of skills $K^n \subset K$.
  - List of contracts $C = \{c_1, \cdots, c_{|C|}\}$.
    - c_1 represents the contract with index 1, i.e., Part-time.
    - $|C|$ contracts in total.



	Mon		Tue		Wed		Thu		Fri		Sat		Sun	
Nurse1	L	C	L	C	-	-	-	-	-	-	-	-	E	C
Nurse2	E	T	-	-	-	-	N	T	N	T	N	C	N	C
Nurse3	-	-	L	H	L	H	L	H	L	H	-	-	-	-

E	Early shift	C	Caretaker
L	Late shift	T	Trainee
N	Night shift	H	Head nurse

Fig. 1: The solution of NRP

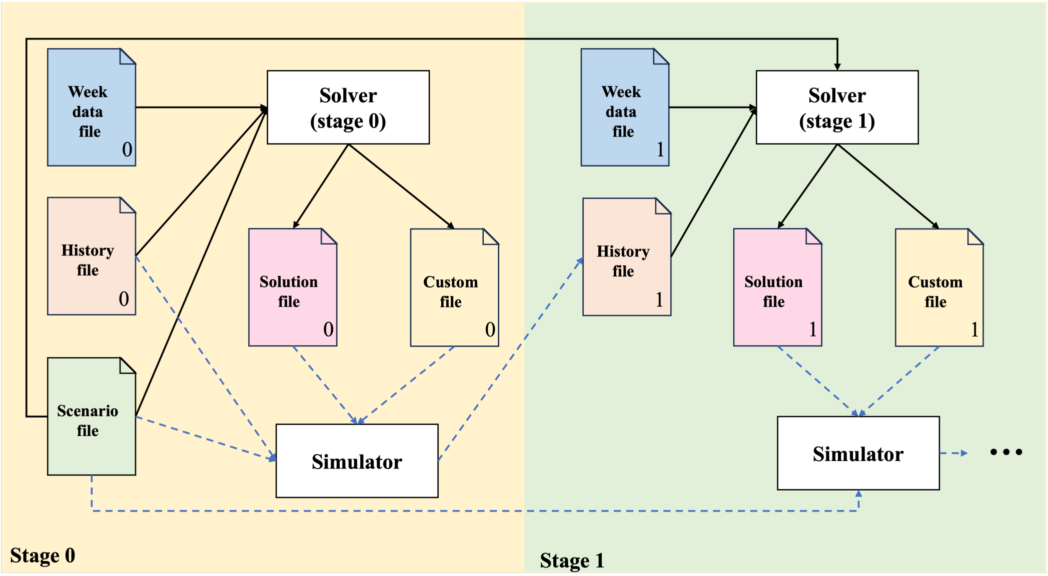


Fig. 2: The overall process of INRC-II

  - A_c^{min}, A_c^{max}: The minimum and maximum total numbers of assignments in the planning horizon for nurses with contract c.
  - W_c^{min}, W_c^{max}: The minimum and maximum numbers of consecutive working days for nurses with contract c.
  - O_c^{min}, O_c^{max}: The minimum and maximum numbers of consecutive days-off for nurses with contract c.
  - B_c^{max}: The maximum number of working weekends in the planning horizon for nurses with contract c.
  - W_c: A Boolean value representing the presence of the complete weekend constraint to the nurse, which states that the nurses with contract c should work both days of the weekend or none of them.
  - Each nurse $n \in N$ is associated with a single contract $c^n \subset C$.

- **Week data information**: which is specific to a single week, including requirements and preferences.
  - $\xi_{(d,s,k)}^{opt}$: The optimal number of nurses with skill k at shift s on day d.
  - $\xi_{(d,s,k)}^{min}$: The minimum number of nurses with skill k at shift s on day d.
  - $U_{(n,d,s)} = 1$: Nurse n hopes not to work at shift s on day d.
  - $V_{(n,d)} = 1$: Nurse n hopes to take a day off on day d.
- **History information**: which is carried over from one week to the following one.
  - s_n: Last assigned shift of nurse n.
  - l_n: The number of consecutive worked shifts of nurse n in history.
  - f_n: The number of consecutive days-off of nurse n in history.
  - l_{s_n}: The number of consecutive worked shifts of the last shift type s_n.

The problem contains four types of hard constraints (H) and eight types of soft constraints (S). Hard constraints must be satisfied, otherwise the generated solutions are infeasible. Our goal is to minimize the objective function: the weighted sum of the soft constraint violations. We use W_{S_i} to indicate the weight associated with constraint S_i, and V_{S_i} to indicate the amount of violation of constraint S_i. The weight values of soft constraints are provided in Table 1. To describe the set of hard and soft constraints with mathematical definitions, we define variables $x_{(n,s,d,k)}$ as follows:

$$x_{n,d,s,k} = \begin{cases} 1 & \text{if nurse } n \text{ is assigned shift } s \\ & \text{on day } d \text{ using skill } k; \\ 0 & \text{otherwise} \end{cases}$$

Minimize:

$$\sum_{i=\{1,2,\ldots,8\}} W_{S_i} \times V_{S_i} = \sum_{i=\{1,2,\ldots,8\}} W_{S_i} \times f_i \tag{1}$$

Subject to:

H1 (Single assignment per day): A nurse can cover at most one shift per day.

$$\sum_{s\in S}\sum_{k\in K} x_{n,d,s,k} \leq 1, \forall n \in N, d \in D \tag{2}$$

H2 (Under-staffing): For a shift s on day d, it minimally needs $\xi^{min}_{d,s,k}$ nurses with skill k.

$$\sum_{n\in N} x_{n,d,s,k} \geq \xi^{min}_{d,s,k}, \forall d \in D, s \in S, k \in K \tag{3}$$

H3 (Shift type successions): The shift type assignments of one nurse in two consecutive days must belong to the legal successions provided in the scenario.

$$\begin{aligned} &\sum_{k\in K} (x_{n,d-1,s_1,k} + x_{n,d,s_2,k}) \leq 1, \\ &\quad \forall n \in N, d \in D, (s_1, s_2) \in F \end{aligned} \tag{4}$$

H4 (Missing required skill): A shift s of a given skill k must necessarily be fulfilled by a nurse n having that skill.

$$x_{n,d,s,k} = 0, \forall n \in N, d \in D, s \in S, k \in K \backslash K^n \tag{5}$$

S1 (Insufficient staffing for optimal coverage): For a shift s on day d, it optimally needs $\xi^{opt}_{d,s,k}$ nurses with skill k.

$$f_1 = \sum_{d\in D}\sum_{s\in S}\sum_{k\in K} \max\{\xi^{opt}_{d,s,k} - \sum_{n\in N} x_{n,d,s,k}, 0\} \tag{6}$$

S2 (Consecutive assignments): the minimum and maximum numbers of consecutive working days. If there exists i that $\sum_{t=0}^{\tau}\sum_{s\in S}\sum_{k\in K} x_{n,d=i+t,s,k} = \tau$, then the number of consecutive working days is the maximum value of τ,.

$$f_2 = \sum_{n\in N} \max(0, W^{min}_{c^n} - \tau) + \sum_{n\in N} \max(0, \tau - W^{max}_{c^n}) \tag{7}$$

S3 (Consecutive assignments per shift type): Similar to S2 but considering the number of consecutive assignments per shift type.

S4 (Consecutive days off): The minimum and maximum numbers of consecutive days-off should be respected. If there exists i that $\sum_{t=0}^{\tau}\sum_{s\in S}\sum_{k\in K} x_{n,d=i+t,s,k} = 0$, then consecutive days off is the maximum value of τ.

$$f_4 = \sum_{n\in N} \max(0, O^{min}_{c^n} - \tau) + \sum_{n\in N} \max(0, \tau - O^{max}_{c^n}) \tag{8}$$

S5 (Preferences): Each assignment to an undesired shift is penalized.

$$\begin{aligned} f_5 = &\sum_{n\in N}\sum_{d\in D} (V_{n,d} \times \sum_{k\in K}\sum_{s\in S} x_{n,d,s,k} \\ &+ \sum_{n\in N}\sum_{d\in D}\sum_{s\in S} (U_{n,d,s} \times \sum_{k\in K} x_{n,d,s,k}) \end{aligned} \tag{9}$$

S6 (Complete weekend): Every nurse that has the complete weekend value set to true must work both weekend days or none.

$$\begin{aligned} f_6 = \sum_{n\in N}\sum_{d\in\, D_1} (W_{c^n} \times | \sum_{k\in K}\sum_{s\in S} x_{n,d,s,k} - \sum_{k\in K}\sum_{s\in S} x_{n,d-1,s,k} |), \\ D_1 = \{d \mid d = 7 \times w, w \in W\} \end{aligned} \tag{10}$$

S7 (Total assignments): For each nurse, the total number of assignments (working days) must be included within the limits (minimum and maximum) enforced by her/his contract.

$$\begin{aligned} f_7 = &\sum_{n\in N} \max\{\sum_{s\in S}\sum_{d\in D}\sum_{k\in K} x_{n,s,d,k} - A^{max}_{c^n}, 0\} \\ &+ \sum_{n\in N} \max\{A^{min}_{c^n} - \sum_{s\in S}\sum_{d\in D}\sum_{k\in K} x_{n,s,d,k}, 0\} \end{aligned} \tag{11}$$

S8 (Total working weekends): For each nurse, the number of working weekends must be less than or equal to the maximum.

$$\begin{aligned} f_8 = &\sum_{n\in N} \max\{ \sum_{d\in D_1} \max(\sum_{s\in S}\sum_{k\in K} x_{n,s,d,k}, \sum_{s\in S}\sum_{k\in K} x_{n,s,d-1,k}) \\ &- B^{max}_{c^n}, 0\}, D_1 = \{d \mid d = 7 \times w, w \in W\} \end{aligned} \tag{12}$$

4. Proposed algorithm

In this section, we first introduce the overall framework of our proposed algorithm MDQN-MA. In MDQN-MA, each agent executes the DQN-based neighborhood tabu search (DQN-NTS) heuristic algorithm. During each search phase, the four agents operate in parallel, competing with each other to place the best solutions obtained into the solution pool. The starting point for the next search phase is

Table 1
Weight values of the soft constraints

Soft Constraint	Weight
S_1	$W_{S_1} = 30$
S_2	$W_{S_2} = 30$
S_3	$W_{S_3} = 15$
S_4	$W_{S_4} = 30$
S_5	$W_{S_5} = 10$
S_6	$W_{S_6} = 30$
S_7	$W_{S_7} = 20$
S_8	$W_{S_8} = 30$

Algorithm 1: The framework of MDQN-MA.

```
Input: Scenario, History, WeekData,
       ActionSpaces:$\{a1, a2, a3, a4\}$,SolutionPool:
       $SolPool$
Output: Optimal schedule set $X$
1  Initialize $X$ as an empty set;
2  $h_0 \leftarrow$ History, $i \leftarrow 0$;
3  while $i < |W|$ do
4      $sol \leftarrow$ Initialization(Scenario, $h_i$, WeekData$_i$);
5      $t \leftarrow 0$;
6      while $t < itermax$ do
           // Algorithm2
7          Thread_1 $sol_1 \leftarrow$ DQN-NTS($sol, a1$);// Agent1
8          Thread_2 $sol_2 \leftarrow$ DQN-NTS($sol, a2$);// Agent2
9          Thread_3 $sol_3 \leftarrow$ DQN-NTS($sol, a3$);// Agent3
10         Thread_4 $sol_4 \leftarrow$ DQN-NTS($sol, a4$);// Agent4
11         $SolPool \leftarrow SolPool \cup \{sol_1, sol_2, sol_3, sol_4\}$;
12         $sol_{\text{best}} \leftarrow$ Select-Best-Solution($SolPool$);
13         $sol \leftarrow sol_{\text{best}}$;
14         $t \leftarrow t + 1$;
15     end
16     $X_i \leftarrow sol_{\text{best}}$;
17     $X \leftarrow X \bigcup X_i$;
18     $i \leftarrow i + 1, h_{i+1} \leftarrow$ Generate-History($sol$);
19 end
```

the best solution in the solution pool. Next, we delve into the details of the implementation of the DQN-NTS, aiming to address the issue of efficient neighborhood selection. Finally, we discuss the differences among the four agents, primarily manifested in their action spaces, as different action spaces imply different functionalities.

4.1. General framework

A schematic diagram of the proposed algorithm for week i is shown in Fig. 3. First, the initial legal solution sol of i-th weeks is generated as the starting point of the search, and then each agent occupies a thread to search in different spaces in parallel. Then the best solution searched by each agent is put into the solution pool, and the best solution is selected as the starting point for the next iterative search.

The overall framework of our proposed MDQN-MS is presented in **Algorithm** 1. The MDQN-MS is designed to tackle the NRP by making the schedule for each week in stages, following the rules outlined in the INRC-II. At each stage, it takes into account various inputs such as the scenario, history from the previous stage, week data in the current stage, and trained DQN for each agent. Finally, MDQN-MA outputs the optimal schedule set denoted as X.

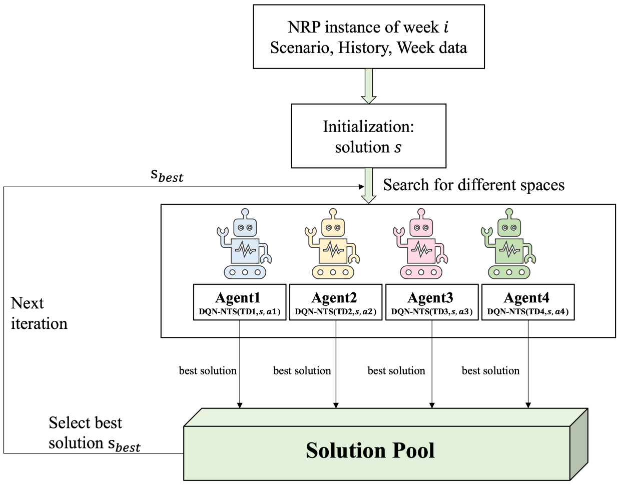


Fig. 3: Schematic diagram of the MDQN-MS during the week i-th

Algorithm 1 begins by initializing the required history information for the current stage, the week index i, and the iteration counter t (line1). The optimization algorithm is then executed until the week index i reaches $|W|$. First, a feasible initial solution sol is generated by a greedy algorithm (line 3). Next, an iterative search is performed until t reaches $itermax$ (line 4). At the beginning of each iteration, four concurrent threads are started simultaneously. An agent runs on each thread, and then each agent starts searching and obtains the best solution it finds (lines 5-8). The optimal solutions found by the agents are placed into the $SolPool$, and the best solution, denoted as sol_{best}, is selected from the $SolPool$ as sol for the starting point of the next iteration (lines 9-11). Increase the number of t by one (line 14). After the iteration ends, the global optimal solution of this week X_i is updated by sol_{best} and put into the optimal schedule set X (lines 14-15). After optimization for the i-th week, the history information for the $(i + 1)$-th week is generated by the optimal solution X_i obtained from the current stage (line 16).

4.2. DQN-based neighborhood tabu search

DQN is an RL algorithm based on deep learning that is commonly used to solve MDP. It can learn the optimal policy without prior knowledge. For each agent, we have adopted the DQN-based neighborhood tabu search algorithm. We integrate RL with neighborhood search, model the neighborhood selection process as a Markov decision process, and then apply the DQN to guide the selection of neighborhoods. DQN integrates mechanisms such as experience replay, enabling the system to learn from past choices and enhance future neighborhood selection, thereby identifying

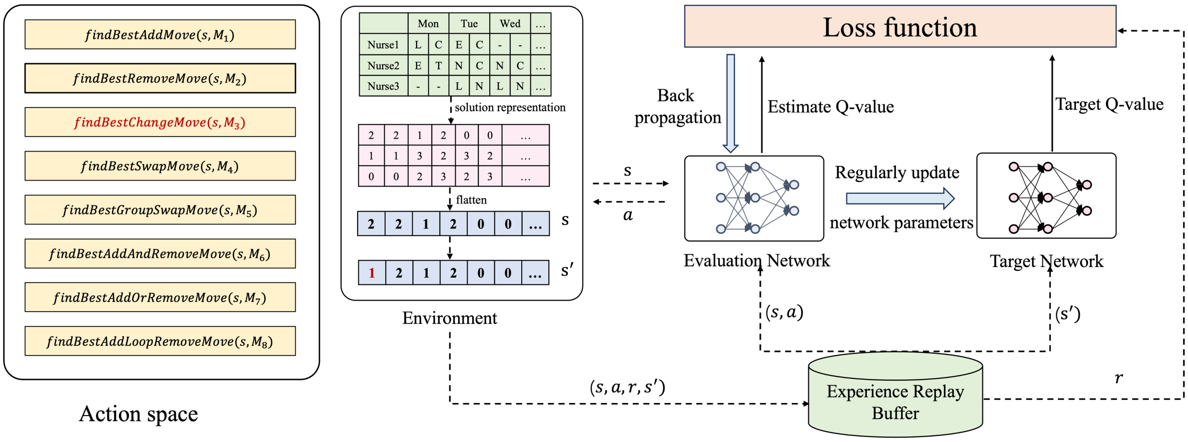


Fig. 4: DQN-based neighborhood tabu search.

effective neighborhood transition sequences from a long-term perspective. The pseudocode of this process is shown in **Algorithm** 2, and its flow is presented in Fig. 4. The difference between various agents lies in their action space.

Algorithm 2: DQN-based Neighborhood Tabu Search **(DQN-NTS)**

```
   Input: Initial solution $sol$, Action space: $A$
   Result: Improved solution $impSol$ in this stage
1  $iterCount \leftarrow 0, done \leftarrow 0$;
2  Encode $sol$ into state $s$;
3  while $iterCount < maxIter$ and $done == 0$ do
4      Select action $a$ from $A$ using $\epsilon$-greedy strategy;
5      $(impSol, \Delta imp) \leftarrow$ ApplyAction$(a, sol)$;
6      Add action $a$ To Tabu List ;
7      Reward $r \leftarrow \Delta imp$;
8      Encode $impSol$ into new state $s'$;
9      $done \leftarrow (r == 0)$;
10     /* Experience Replay */
11     Store $(s, a, r, s', done)$ in replay buffer;
12     Sample mini-batch from buffer: $(s_j, a_j, r_j, s'_j, done_j)$;
13     $y_j \leftarrow r_j + \gamma \max_{a'} \hat{Q}(s'_j, a'; \theta^-) \cdot (1 - done_j)$;
14     Update network parameters: Perform gradient
         descent on $(y_j - Q(s_j, a_j; \theta))^2$ with respect to $\theta$;
15     Every $N$ steps update target network: $\theta^- \leftarrow \theta$;
16     $iterCount \leftarrow iterCount + 1$;
17 end
```

This process utilizes DQN for neighborhood selection and integrates with tabu search. It initializes with a flag, $done$ set to 0, and a step counter, $iterCount$, set to 0, to manage loop execution and monitor progress within each episode (line 1). The current solution, sol, is encoded into a state s, serving as an input for the DQN model (line 2). The main loop of the algorithm commences, with termination conditions based on the completion of a maximum number of steps $maxIter$ or reaching a terminal state (line 3). At each iteration, an action a is chosen using an ϵ-greedy strategy by DQN from the action space A. This strategy balances exploration and exploitation by occasionally selecting random actions (line 4). The selected action is applied to the current solution through the ApplyAction function, yielding an improved solution $impSol$ and a change in cost Δimp (line 5). To prevent the immediate reversal of the action a, we include it in the tabu list (line 6). Subsequently, the reward, r, is updated to reflect the improvement (line 7), and the algorithm encodes the improved solution into a new state, s' (line 8). During this critical phase, the algorithm stores the transition tuple, $(s, a, r, s', done)$, in the replay buffer (line 11), which retains experiences for batch learning. A mini-batch of transitions is then sampled from the replay buffer to update the model (line 12). For each sampled transition, the algorithm checks if it reaches a terminal state. If so, the target value, y_j, is set to the immediate reward, r_j. Otherwise, y_j is calculated as the sum of the immediate reward and the discounted estimate of future rewards, based on the target network parameters (line 13). A gradient descent step is performed to minimize the loss between the predicted Q-values and the target values, updating the network parameters (line 14). In every N steps, the target Q-network parameters are synchronized with the Q-network to stabilize learning (line 15), and the step counter, $iterCount$, is incremented, advancing the algorithm through the episode (line 16). The loop continues until either the maximum number of steps is reached or a terminal state is identified, ensuring exploration of the solution space within the episode's constraints (line 17). By integrating experience replay and a dynamic action selection strategy, DQNNS systematically explores the solution space, aiming to iteratively improve upon the initial solution provided.

Fig. 4 provides a detailed depiction of the implementation specifications of the DQN model in neighborhood search. This model encompasses the design of state, action, and reward.

1. **State**: The design of the state aims to encompass relevant information regarding the current shift schedule. We encode the current solution sol to be treated as

the current state s. As illustrated in Figure 4, numerical values are assigned to represent the shifts and skills of each nurse. Different numbers are used to differentiate shifts and skills. For example, a value of $(0,0)$ indicates that the nurse is not working on that specific day. To facilitate generalization in DQN, each solution is structured as a $2 \times 7 \times 120$ matrix, where 2 represents the shift and skill of each nurse, 7 represents the number of days in a week, and 120 denotes the maximum number of nurses in the INRC-II dataset.

2. **Action**: The action space of each agent is defined as a combination of the best moves in designed neighborhoods. For a solution sol, which is used to obtain a new solution sol' after applying a neighborhood move m, the set of moves of the same type applied to s is denoted as $M(s)$. Within each neighborhood, the move that yields the most substantial improvement to the current solution is selected as the action a. By considering the current state s and chosen action a, the system transitions to a deterministic next state s', thereby navigating toward an optimal solution.

3. **Reward**: Designing an effective reward function is crucial in RL, as it directly impacts the learning process and the network's performance. One advantage of the NRP is that the objective function of Eq. 1 is already known. Therefore, the reward value is defined as the difference between the objective value of the original solution sol and the objective value of the solution sol' after the action is applied. This enables the DQN agent to learn and prioritize actions that lead to better solutions. It is important to note that the infeasible solutions are assigned a significant negative reward value (-100) which indicates our strong disapproval of executing actions that result in infeasible solutions.

4.3. Action spaces for agents

The difference between each agent is that they have various action spaces, i.e. a different combination of neighborhoods, which enables the search for solutions in different directions. The eight neighborhoods we designed are divided into two types: basic neighborhoods and composite neighborhoods. The first type includes four basic neighborhoods, denoted as M_1 to M_4, which perform single-step operations. Each basic neighborhood focuses on a specific aspect of the solution and modifies it accordingly. The second type consists of two composite neighborhoods, denoted as M_5 to M_8, which are composed of basic neighborhoods. These composite neighborhoods combine the modifications proposed by multiple basic neighborhoods to generate more diverse and comprehensive changes to the solution.

M_1**: Add-shift operator.** Fig. 5 shows an example of an add-shift operator, with the original schedule left and the roster after the operation right. In this figure, Nurse1 originally has a day off on Saturday, but now she is assigned to the late shift with the required skill set as a caretaker. The add-shift neighborhood, i.e. the set of add-shift operators, is defined as follows: $M_1(s) = \{m_1(n,d,h,k) | \forall n \in N, d \in D_i, h \in H, k \in K, x^H_{i,n,d} = \text{ShiftOff} \text{ and } h \neq \text{ShiftOff}\}$.

M_2**: Remove-shift operator.** Fig. 6 illustrates an example of a remove-shift operator, with the original schedule left and the roster after the operation right. In this figure, Nurse2 is originally scheduled for the night shift on Saturday, but now she is changed to have a day off on that day. The remove-shift neighborhood, i.e., the set of remove-shift operators, is defined as follows: $M_2(s) = \{m_2(n,d) | \forall n \in N, d \in D, x^H_{i,n,d} \neq \text{ShiftOff}\}$.

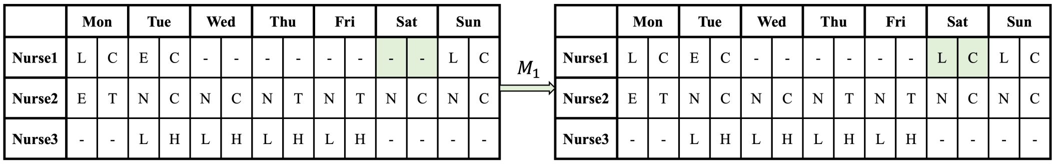
M_3**: Change-shift operator.** Fig. 7 illustrates an example of a change-shift operator, with the original schedule left and the roster after the operation right. In this figure, Nurse3 is originally scheduled for the late shift on Tuesday, but now she is changed to the early shift. The change-shift neighborhood, i.e., the set of change-shift operators, is defined as follows: $M_3(s) = \{m_3(n,d,s,k) | \forall n \in N, d \in D, x^H_{i,n,d} = \text{ShiftOff} \cap h \neq \text{ShiftOff} \cap (x^H_{i,n,d} \neq h \cup x^K_{i,n,d} \neq k)\}$.

M_4**: Swap-shift operator.** Fig. 8 illustrates an example of a swap-shift operator, with the original schedule left and the roster after the operation right. Nurse1 is originally scheduled for the late shift on Monday with the skill of Caretaker, while Nurse2 is scheduled for the early shift with the skill of Trainee. Now Nurse1 is changed to the early shift with the skill of Trainee, and Nurse2 is changed to the late shift with the skill of Caretaker on Monday. The swap-shift neighborhood, i.e., the set of swap-shift operators, is defined as follows: $M_4(s) = \{m_4(n_1,n_2,d) | \forall n \in N, d \in D, x^H_{i,n_1,d} = x^H_{i,n_2,d}, x^K_{i,n_1,d} = x^K_{i,n_2,d}\}$.

Although the single-move basic neighborhoods have lower computational complexity, they may miss out on the potential for achieving more considerable improvements through combined actions. To address this limitation, it is imperative to incorporate composite neighborhoods during the neighborhood search process. These composite neighborhoods offer a more comprehensive approach by considering multiple actions simultaneously. By expanding the search space with composite neighborhoods, we can effectively explore a broader range of solutions, which allows for the discovery of more advantageous combinations of actions to achieve significant enhancements.

M_5**: Group-swap operator.** The group-swap operator functions as follows: for two selected nurses, we swap their shifts from d_1 to d_2. The set of neighboring operators for the group-swap neighborhood is defined as follows: $M_5(s) = \{m_5(n_1,n_2,d) \mid \forall n_1,n_2 \in N, d \in [d_1,d_2]\}$.

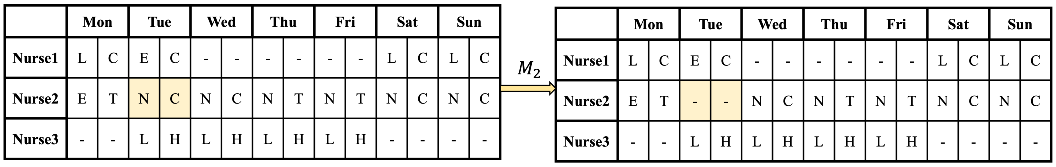
M_6**: Add-and-remove-shift operator.** The add-and-remove operator functions as follows: for a selected nurse, if the nurse does not have any scheduled work on day d, we perform an add-shift operator by assigning the nurse to a shift h on day d. Conversely, if the nurse has already been assigned on day d, we perform a remove-shift operator by assigning the nurse a day off on day d. The set of neighboring operators for the add-and-remove neighborhood is given by: $M_6(s) = \{m_1(n,d,h,k) \mid \forall n \in N, d \in D, h \in H, k \in$



	Mon		Tue		Wed		Thu		Fri		Sat		Sun	
Nurse1	L	C	E	C	-	-	-	-	-	-	-	-	L	C
Nurse2	E	T	N	C	N	C	N	T	N	T	N	C	N	C
Nurse3	-	-	L	H	L	H	L	H	L	H	-	-	-	-

	Mon		Tue		Wed		Thu		Fri		Sat		Sun	
Nurse1	L	C	E	C	-	-	-	-	-	-	L	C	L	C
Nurse2	E	T	N	C	N	C	N	T	N	T	N	C	N	C
Nurse3	-	-	L	H	L	H	L	H	L	H	-	-	-	-

Fig. 5: Example of the add-shift operator.



	Mon		Tue		Wed		Thu		Fri		Sat		Sun	
Nurse1	L	C	E	C	-	-	-	-	-	-	L	C	L	C
Nurse2	E	T	N	C	N	C	N	T	N	T	N	C	N	C
Nurse3	-	-	L	H	L	H	L	H	L	H	-	-	-	-

	Mon		Tue		Wed		Thu		Fri		Sat		Sun	
Nurse1	L	C	E	C	-	-	-	-	-	-	L	C	L	C
Nurse2	E	T	-	-	N	C	N	T	N	T	N	C	N	C
Nurse3	-	-	L	H	L	H	L	H	L	H	-	-	-	-

Fig. 6: Example of the remove-shift operator.

$K, x^H_{i,n,d} = \text{ShiftOff} \wedge h \neq \text{ShiftOff}\} \cup \{m_2(n,d) \mid \forall n \in N, d \in D, x^H_{i,n,d} \neq \text{ShiftOff}\}$.

M_7**: Add-or-remove-shift operator.** The add-or-remove operator functions as follows: it randomly selects the best move from the M_1 or M_2. The set of neighboring operators for the add-or-remove neighborhood is given by: $M_7(s) = \{M_1(s) | rand < 0.5\} \cup \{M_2(s) | rand > 0.5\}$. In this definition, $rand$ refers to a random number between 0 and 1.

M_8**: Add-loop-remove-shift operator.** The add-loop-remove operator functions as follows: M_2 is executed only after the best move in the M_1 neighborhood does not improve the solution and vice versa. The set of neighboring operators for the add-loop-remove neighborhood is given by:

$$M_8(s) = \begin{cases} M_1(s), & \text{if } flag = \text{true} \\ M_2(s), & \text{otherwise} \end{cases}.$$

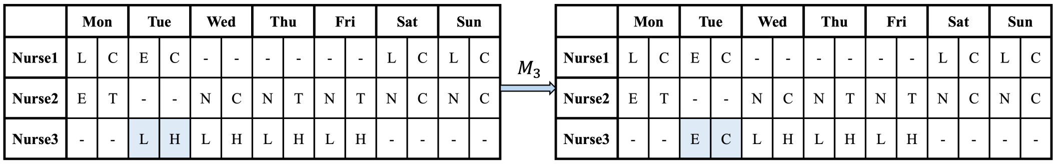
In this definition, the $flag$ changes according to the function we want to achieve.

It is crucial to balance exploration and exploitation when developing an effective search strategy. To achieve this goal, we devised four agents with distinct neighborhood operator combinations based on experimental results and theoretical basis. The action spaces of the four agents we designed are combinations of the above eight types of neighborhoods. The specific details are shown in Table 2. In the algorithm we proposed, these four agents cooperate and reach the approximate optimal solution.

1. **Agent1-Comprehensive Exploration Agent:**Agent1 integrates six neighborhood operators ($M_1 - M_6$), which is the agent with the largest number of neighborhoods among four agents. This shows that it employs multiple neighborhoods including single neighborhoods and composite neighborhoods and to conduct a diversified search strategy, which can improve the exploration of the search space.

2. **Agent2-Focused Exploitation Agent:**Agent2 employs three small single-step neighborhood operators ($M_1 - M_3$), which allow it to focus on perturbation in a narrow range of the solution space. This contributes to a more thorough exploitation of that specific region around the current best solution.

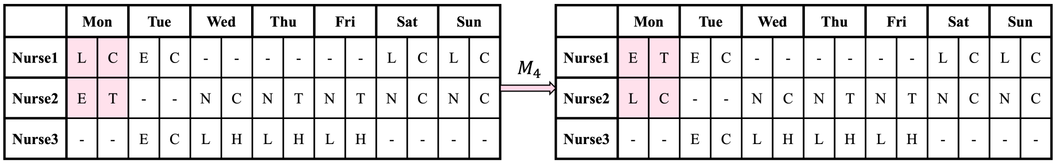
3. **Agent3-Swap Exploitation Agent:**Agent3 employs two swapping neighborhood operators (M_4, M_5) to optimize the solution by swapping the shift of nurses,



	Mon		Tue		Wed		Thu		Fri		Sat		Sun	
Nurse1	L	C	E	C	-	-	-	-	-	-	L	C	L	C
Nurse2	E	T	-	-	N	C	N	T	N	T	N	C	N	C
Nurse3	-	-	L	H	L	H	L	H	L	H	-	-	-	-

	Mon		Tue		Wed		Thu		Fri		Sat		Sun	
Nurse1	L	C	E	C	-	-	-	-	-	-	L	C	L	C
Nurse2	E	T	-	-	N	C	N	T	N	T	N	C	N	C
Nurse3	-	-	E	C	L	H	L	H	L	H	-	-	-	-

Fig. 7: Example of the change-shift operator.



	Mon		Tue		Wed		Thu		Fri		Sat		Sun	
Nurse1	L	C	E	C	-	-	-	-	-	-	L	C	L	C
Nurse2	E	T	-	-	N	C	N	T	N	T	N	C	N	C
Nurse3	-	-	E	C	L	H	L	H	L	H	-	-	-	-

	Mon		Tue		Wed		Thu		Fri		Sat		Sun	
Nurse1	E	T	E	C	-	-	-	-	-	-	L	C	L	C
Nurse2	L	C	-	-	N	C	N	T	N	T	N	C	N	C
Nurse3	-	-	E	C	L	H	L	H	L	H	-	-	-	-

Fig. 8: Example of the swap-shift operator.

Table 2
The combination of neighborhoods for each agent

Agents	*Role*	*Combination of neighborhoods*
Agent1	Comprehensive Exploration Agent	M_1, M_2, M_3, M_4, M_5, M_6
Agent2	Focused Exploitation Agent	M_1, M_2, M_3
Agent3	Swap Exploitation Agent	M_4, M_5,
Agent4	Disruptive Exploration Agent	M_6, M_7, M_8

which can keep the total number of shifts per day unchanged. This improves the exploitation of the current solution.

4. **Agent4-Disruptive Exploration Agent:** Agent4 combines three neighborhood operators ($M_6 - M_8$), all of which employ a strategy of partially ruining and then recreate the current solution. This disruptive strategy helps to jump out of the local optimal solution and provides a powerful exploration mechanism.

The neighborhood combination strategies of these four agents have their advantages, which ensure comprehensive exploration in the vast solution space while exploiting and optimizing high-quality solutions once identified. The MAS we designed effectively merges the strengths of exploration and exploitation, greatly increasing the quality of the solution.

5. Experimental study

In this section, we first enumerate the characteristics of the data set and the corresponding instances generated. Then we present a concise overview of the computational outcomes achieved by MDQN-MA on a set of instances and verify its effectiveness by comparing it with state-of-the-art algorithms. Finally, we conduct ablation experiments to further demonstrate the effectiveness of our proposed DQN and multi-agent framework.

5.1. Dataset analysis and experimental parameter settings

The INRC-II organizers coordinated benchmark tests comprising a collection of 17 datasets. Each dataset contained one scenario file, three initial history files, and ten weekdata files. The weekdata files were reusable within the same instance, enabling the creation of different test instances of the same dataset. Table 3 presents an analysis of the dataset characteristics and the instances generated by this dataset. An instance is a specific scenario, an initial history, and a sequence of 4 (or 8) week data files, all of which belong to the same dataset. For example, the instance n030w4_0_8-8-3-6 is one with 30 nurses over 4 weeks, where the index for the history file is 0, and the indices for the four week data files are 8, 8, 3, and 6 respectively. To ensure a fair comparison between algorithms, we established a standardized approach. We set the number of searches to 4500 for one iteration on all algorithms and a total of 20 iterations were performed. Implementing a fixed number of searches allowed each algorithm to have an equal opportunity to explore the search space and discover optimal or nearly optimal solutions. This approach minimizes the impact of random fluctuations and guarantees an unbiased performance comparison.

Table 3
Characteristics of the 17 datasets and corresponding instances.

Dataset	$\|N\|$	$\|k\|$	$\|S\|$	$\|W\|$	$\|C\|$	Problem	Instance
D1	5	2	3	4	2	P_1	n005w4_0_1-2-3-3
D2	12	2	4	8	2	P_2	n012w8_1_7-7-0-8-9-3-2-6
D3	21	4	4	4	3	P_3	n021w4_0_5-5-1-5
D4	30	4	4	4	3	P_4	n030w4_0_8-8-3-6
D5	30	4	4	8	3	P_5	n030w8_1_2-8-3-5-7-0-0-3
D6	40	4	4	4	3	P_6	n040w4_0_3-8-5-2
D7	40	4	4	8	3	P_7	n040w8_2_2-3-3-4-2-9-8-1
D8	50	4	4	4	3	P_8	n050w4_0_3-1-8-5
D9	50	4	4	8	3	P_9	n050w8_0_8-8-7-5-5-3-4
D10	60	4	4	4	4	P_{10}	n060w4_0_9-9-4-9
D11	60	4	4	8	3	P_{11}	n060w8_0_2-0-2-6-5-9-6-1
D12	80	4	4	4	4	P_{12}	n080w4_1_6-1-9-8
D13	80	4	4	8	4	P_{13}	n080w8_1_2-6-5-8-2-2-6-3
D14	100	4	4	4	4	P_{14}	n100w4_0_7-3-7-9
D15	100	4	4	8	4	P_{15}	n100w8_1_0-8-0-6-1-1-5-3
D16	120	4	4	4	3	P_{16}	n120w4_0_1-0-4-9
D17	120	4	4	8	3	P_{17}	n120w8_2_6-2-5-4-1-3-6-2

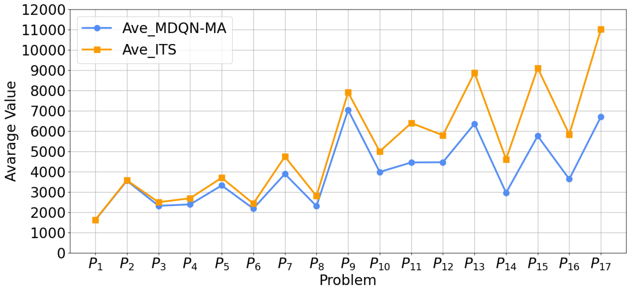


Fig. 9: The average values for 10 independent runs obtained by MDQN-MA and ITS on 17 instances.

5.2. Comparison with the best metaheuristic algorithm in the INRC-II

To demonstrate the effectiveness of our proposed MDQN-MA, we selected the iterative tabu search algorithm (ITS) proposed by the Hust.Smart group [25], which won an award in the competition and ranked first in the category of metaheuristic algorithms. By comparing our proposed MDQN-MA with this highly successful metaheuristic algorithm, we can validate the effectiveness and advancement of MDQN-MA. For a fair comparison, we also set the stopping condition of the two algorithms to perform 20 iterations, with each iteration performing 4500 searches. The average values of ten independent runs of MDQN-MA and ITS on 17 instances are plotted in Fig. 9. We can see that the two algorithms only show similar performances on P_1 and P_2, while MDQN-MA performs better than the ITS in subsequent instances. One interesting observation is that the performance gap between MDQN-MA and ITS increases with the increasing complexity of NRP. This is because DQN can make more global decisions by learning knowledge during training, and multi-agent systems can learn richer information through interactions between agents, better guiding the search process. This result demonstrates the effectiveness of MDQN-MA in tracking the NRP, especially in complex instances.

5.3. Comparison with the state-of-the-art metaheuristic algorithms

To further demonstrate the effectiveness of MDQN-MA, we conducted comparative analyses with several competitive metaheuristic algorithms recently proposed in the field of COP. We selected three state-of-the-art algorithms: an adaptive large-scale neighborhood search framework with a simulated annealing acceptance mechanism (ALNS-SA) proposed in [23], a hybrid tabu search - variable neighborhood descent (HTS-VND) meta-heuristic algorithm introduced in [24], and a smart general variable neighborhood search with adaptive local search (SGVNSALS) proposed in [56]. We adapted the neighborhoods of these three methods to suit the NRP and compared them with the MDQN-MA under fair conditions. Each algorithm performs 20 iterations, with each iteration performing 4500 searches. Table 4 displays the best and average values obtained by independently running the four algorithms (ALNS-SA, HTS-VND, SGVNSALS, and MDQN-MA) 10 times for each instance. The best result for each instance is highlighted in bold font within the table. Additionally, to ensure a statistically meaningful comparison, we conducted the Wilcoxon rank-sum test at a 95% confidence level on the experimental results. The symbols $+$, $\approx$, and $-$ indicate that the compared algorithm statistically outperforms, performs equally with, or underperforms MDQN-MA, respectively.

From the experimental results, it is evident that MDQN-MA outperformed ALNS-SA, HTS-VND, and SGVNSALS in most instances. This suggests that MDQN-MA possesses significant advantages over these state-of-the-art metaheuristic algorithms in tackling combinatorial optimization problems. One explanation for the superior performance of MDQN-MA could be its ability to leverage deep reinforcement learning techniques, which enable it to make more informed and adaptive decisions based on learned knowledge. This adaptability allows MDQN-MA to effectively explore the solution space and discover high-quality solutions even in complex and challenging instances of the NRP. Furthermore, the multi-agent nature of MDQN-MA facilitates information sharing and collaborative learning among agents, leading to a more comprehensive exploration of the solution space. By leveraging the collective intelligence of multiple agents, MDQN-MA can exploit synergies between different search trajectories and avoid getting trapped in local optima, thereby enhancing its search efficiency and effectiveness.

5.4. The effectiveness of the DQN

We have chosen the neighborhood selection methods ANS and VND from two state-of-the-art algorithms ALNS-SA [23] and HTS-VND [24] mentioned in section 5.3 to replace the neighborhood selection method of each agent (line 3 of algorithm 2). These two comparison algorithms are named MANS-MA, and MVND-MA, corresponding to two neighborhood selection strategies: ANS and VND. By comparing MDQN-MA with these highly successful multi-agent metaheuristic algorithms with state-of-the-art neighborhood selection methods, we can assess the effectiveness of DQN for neighborhood search in tackling the NRP.

We conduct experiments in 17 instances for the datasets and list the best and average values in Table 5 obtained by running the three algorithms (MDQN-MA, MANS-MA, and MVND-MA) independently 10 times for each instance. The best result of each instance is highlighted using the bold font in the table. Moreover, to obtain a statistically sound comparison, the Wilcoxon rank-sum test with 95% confidence level was conducted on the experimental results. The comparison results indicate that MDQN-MA outperforms both MANS-MA and MVND-MA across all 17 instances of the datasets. This suggests that the DQN-based approach for neighborhood search in tackling the NRP is more effective compared to the multi-agent metaheuristic algorithms with ANS and VND neighborhood selection methods. DQN's ability to learn from experience and calculate long-term reward expectations enables it to effectively guide the search process, thus contributing to the superior performance observed in experiments. These findings highlight the robustness and strategic advantages of incorporating DQN into a multi-agent framework.

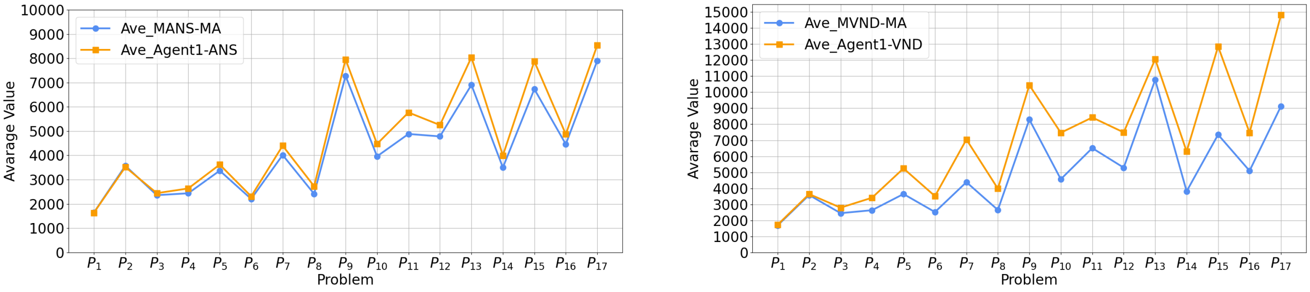
5.5. The effectiveness of multi-agent

To verify the effectiveness of the multi-agent framework we proposed, we run the single-agent algorithm for comparative analysis. In addition, we ran a single agent (Agent5) with all eight neighborhoods ($M_1 - M_8$) we designed, excluding the four agents mentioned above, for comparative analysis. We conduct experiments in 17 instances for the datasets and list the average values in Table 6 obtained by running the five algorithms (Agent1, Agent2, Agent3, Agent4, Agent5, and Multi-Agent) independently 10 times for each instance. We can see that Multi-Agent shows better performance in all instances which validates that the cooperation between agents can improve the performance of our algorithm. The coordination among agents allows for a more comprehensive exploration of the solution space, enabling the algorithm to discover better solutions more efficiently.

Additionally, each agent's performance and functionality are different. The performance of Agent5 falls short of that of Agent1 and the multi-agent, and in some cases, even lags behind Agent4. This indicates that the performance of the single agent encompassing all neighborhood structures cannot surpass that of multi-agents, and sometimes even lags behind agents such as Agent1 and Agent4, which include only specific neighborhoods. This highlights the importance of neighborhood combinations and further validates the effectiveness of designing multi-agent frameworks with different neighborhood structures. While some agents may excel in specific contexts, the collective effort of multiple agents within the multi-agent framework ensures robustness and adaptability across different problem instances. To further validate the generality of our proposed multi-agent

Table 4
Numerical results of ALNS-SA, HTS-VND, SGVNSALS, and MDQN-MA ("Ave" and "Best" respectively denote the average cost and the best cost across 10 independent runs).

Problem	NRP Instance	ALNS-SA		HTS-VND		SGVNSALS		MDQN-MA	
		Ave	Best	Ave	Best	Ave	Best	Ave	Best
P_1	n005w4_0_1-2-3-3	1636.5(≈)	1615	1728.5(−)	1605	1645.5(≈)	1605	**1615.5**	**1595**
P_2	n012w8_1_7-7-0-8-9-3-2-6	**3518.5**(≈)	3415	3640.0(−)	3530	3535.0(≈)	**3310**	3550.0	3400
P_3	n021w4_0_5-5-1-5	2442.5(≈)	2315	2793.0(−)	2635	2354.5(≈)	2270	**2317.0**	**2210**
P_4	n030w4_0_8-8-3-6	2636.5(−)	2490	3413.0(−)	3265	2560.5(≈)	2440	**2385.0**	**2290**
P_5	n030w8_1_2-8-3-5-7-0-0-3	3619.5(−)	3420	5241.5(−)	5000	3447.0(≈)	3280	**3319.0**	**3220**
P_6	n040w4_0_3-8-5-2	2310.5(≈)	2160	3514.0(−)	3200	2257.0(≈)	2185	**2180.0**	**2100**
P_7	n040w8_2_2-3-3-4-2-9-8-1	4399.0(−)	4290	7046.5(−)	6675	4479.0(−)	4265	**3887.5**	**3800**
P_8	n050w4_0_3-1-8-5	2728.5(−)	2565	3980.5(−)	3750	2601.5(−)	2490	**2298.0**	**2145**
P_9	n050w8_0-8-8-7-5-5-3-4	7955.0(−)	7605	10439.0(−)	10125	8020.5(−)	7550	**7043.5**	**6815**
P_{10}	n060w4_0_9-9-4-9	4471.5(−)	4090	7468.0(−)	6785	4784.5(−)	4610	**3983.5**	**3885**
P_{11}	n060w8_0_2-0-2-6-5-9-6-1	5768.5(−)	5555	8420.5(−)	8105	5755.0(−)	5380	**4454.0**	**4150**
P_{12}	n080w4_1_6-1-9-8	5250.0(−)	5080	7490.0(−)	6675	5666.5(−)	5085	**4463.0**	**4350**
P_{13}	n080w8_1_2-6-5-8-2-2-6-3	8042.0(−)	7795	12060.0(−)	11510	9470.0(−)	8905	**6360.0**	**6190**
P_{14}	n100w4_0_7-3-7-9	3996.5(−)	3840	6296.0(−)	6025	6650.5(−)	5635	**2958.0**	**2825**
P_{15}	n100w8_1_0-8-0-6-1-1-5-3	7876.0(−)	7510	12852.0(−)	12210	12653.0(−)	12010	**5758.5**	**5685**
P_{16}	n120w4_0_1-0-4-9	4870.0(−)	4535	7472.0(−)	6920	9346.0(−)	8345	**3631.5**	**3435**
P_{17}	n120w8_2_6-2-5-4-1-3-6-2	8540.0(−)	8350	14824.0(−)	14615	18479.5(−)	15685	**6702.5**	**6335**
(+/≈/−)		(0/4/16)		(0/0/17)		(0/6/11)		—	
(*best*/*all*)		(1/17)		(0/17)		(1/17)		(15/17)	
Average Ranking		2.29		3.82		2.76		1.12	



(a) **(b)**

Fig. 10: The average values for 10 independent runs obtained by MANS-MA, MVND-MA, Agent1-ANS, and Agent1-VND on 17 instances.

framework combined with metaheuristic algorithms, we employ the previously mentioned ANS and VND metaheuristic algorithms to replace **Algorithm** 2. We integrate these algorithms with our multi-agent framework and conduct a comparative study with the best-performing single agent, Agent1. Specifically, we compare MANS-MA and MVND-MA with Agent1, which utilizes ANS or VND for neighborhood selection with neighborhood combinations from M_1 to M_6. The average values of ten independent runs of MANS-MA, MVND-MA, Agent1-ANS, and Agent1-VND on 17 instances are plotted in Fig. 10.

6. Conclusion

This paper has proposed MDQN-MA, a novel approach for tackling the NRP. We have shown that leveraging a multi-agent framework coupled with metaheuristics is a promising strategy for tackling the NRP. This innovative approach allows us to harness the collective intelligence of multiple agents to solve complex scheduling problems efficiently. Then, we introduce the concept of modeling the neighborhood selection process as an MDP and integrating DQN into this process. This integration has proven to be an effective way to maximize long-term rewards, leading to improved

Table 5
Numerical results of MDQN-MA, MANS-MA, AND MVND-MA ("Ave" and "Best" respectively denote the average cost and the best cost across 10 independent runs).

Problem	Instance	MANS-MA		MVND-MA		MDQN-MA	
		Ave	Best	Ave	Best	Ave	Best
P_1	n005w4_0_1-2-3-3	1641.5(≈)	1625	1687.5(≈)	1680	**1615.5**	**1595**
P_2	n012w8_1_7-7-0-8-9-3-2-6	3572.5(≈)	3510	3580.0(≈)	3515	**3550.0**	**3400**
P_3	n021w4_0_5-5-1-5	2360.0(≈)	2305	2450.0(−)	2415	**2317.5**	**2215**
P_4	n030w4_0_8-8-3-6	2437.5(−)	2380	2635.0(−)	2570	**2385.0**	**2290**
P_5	n030w8_1_2-8-3-5-7-0-0-3	3370.0(≈)	3335	3645.0(−)	3560	**3319.0**	**3280**
P_6	n040w4_0_3-8-5-2	2205.0(≈)	2105	2520.0(−)	2440	**2180.0**	**2100**
P_7	n040w8_2_2-3-3-4-2-9-8-1	4015.0(−)	4005	4385.0(−)	4375	**3887.5**	**3800**
P_8	n050w4_0_3-1-8-5	2412.5(−)	2355	2650.0(−)	2580	**2298.0**	**2145**
P_9	n050w8_0-8-8-7-5-5-3-4	7277.5(−)	7135	8292.5(−)	7935	**7043.0**	**6815**
P_{10}	n060w4_0_9-9-4-9	4135.0(−)	3925	4570.0(−)	4430	**3983.5**	**3885**
P_{11}	n060w8_0_2-0-2-6-5-9-6-1	4882.5(−)	4795	6510.0(−)	6130	**4454.0**	**4150**
P_{12}	n080w4_1_6-1-9-8	4782.5(−)	4770	5290.0(−)	5100	**4463.0**	**4350**
P_{13}	n080w8_1_2-6-5-8-2-2-6-3	6905.0(−)	6885	10770.0(−)	10490	**6360.0**	**6190**
P_{14}	n100w4_0_7-3-7-9	3500.0(−)	3420	3815.0(−)	3735	**2958.0**	**2825**
P_{15}	n100w8_1_0-8-0-6-1-1-5-3	6732.5(−)	6385	7350.0(−)	7200	**5758.5**	**5685**
P_{16}	n120w4_0_1-0-4-9	4460.0(−)	4260	5092.5(−)	5060	**3631.5**	**3435**
P_{17}	n120w8_2_6-2-5-4-1-3-6-2	7900.0(−)	7850	9127.5(−)	8880	**6702.5**	**6335**
(+/≈/−)		(0/5/12)		(0/2/15)		—	

Table 6
Numerical results of Agent1, Agent2, Agent3, Agent4, Agent5 and Multi-Agent

Problem	Instance	Agent1 $(M_1 - M_6)$	Agent2 $(M_1 - M_3)$	Agent3 (M_4, M_5)	Agent4 $(M_6 - M_8)$	Agent5 $(M_1 - M_8)$	Multi-Agent
P_1	n005w4_0_1-2-3-3	1620.5	2340.5	2567.5	1655.0	1626.5	**1615.5**
P_2	n012w8_1_7-7-0-8-9-3-2-6	3560.0	4462.5	7217.5	3552.5	3570.0	**3550.0**
P_3	n021w4_0_5-5-1-5	2335.5	6100.0	5742.5	2412.5	2371.0	**2317.0**
P_4	n030w4_0_8-8-3-6	2490.0	6797.5	5540.0	2545.0	2577.5	**2385.0**
P_5	n030w8_1_2-8-3-5-7-0-0-3	3380.0	13875.0	9142.5	3562.5	3523.5	**3319.0**
P_6	n040w4_0_3-8-5-2	2251.5	11725.0	7447.5	2302.5	2545.0	**2180.0**
P_7	n040w8_2_2-3-3-4-2-9-8-1	4100.0	25937.5	13377.5	5137.5	5021.5	**3887.5**
P_8	n050w4_0_3-1-8-5	2351.5	14232.5	7017.5	2620.0	2568.5	**2298.0**
P_9	n050w8_0-8-8-7-5-5-3-4	7373.5	32535.0	15817.5	7567.5	7730.0	**7043.5**
P_{10}	n060w4_0_9-9-4-9	4085.5	19600.0	13045.0	4837.5	4572.5	**3983.5**
P_{11}	n060w8_0_2-0-2-6-5-9-6-1	5022.0	24265.0	17215.0	5687.5	5725.0	**4454.0**
P_{12}	n080w4_1_6-1-9-8	4683.5	23342.5	13142.5	4897.5	4953.5	**4463.0**
P_{13}	n080w8_1_2-6-5-8-2-2-6-3	6903.5	55962.5	21917.5	7280.0	7222.5	**6360.0**
P_{14}	n100w4_0_7-3-7-9	3290.0	30185.0	10335.0	3652.5	3528.5	**2958.0**
P_{15}	n100w8_1_0-8-0-6-1-1-5-3	6514.5	65297.0	20675.0	7070.0	7266.0	**5758.5**
P_{16}	n120w4_0_1-0-4-9	3890.5	24295.0	15270.0	4705.0	4763.5	**3631.5**
P_{17}	n120w8_2_6-2-5-4-1-3-6-2	7325.0	48992.5	30440.0	9510.0	9503.0	**6702.5**

solution quality. Our extensive experiments and analyses have consistently demonstrated the superior performance of MDQN-MA when compared to state-of-the-art algorithms such as MANS-MA, MVND-MA, and ITS. Notably, as the complexity of the instances increases, the performance gap between MDQN-MA and its competitors widens, further emphasizing its suitability for addressing large-scale NRP instances in real-world scenarios.

The promising results obtained from our experiments highlight the potential for further improvements and advancements in this field. One avenue for future research is to explore ways to improve communication and cooperation between agents within the MAS that can potentially lead to more efficient solutions and better scalability. Furthermore, we will explore other RL-based models to enhance the algorithm's performance. For instance, neural architecture search (NAS) techniques could be employed to design network models that are better suited for the NRP [57]. In addition, investigating the impact of different training techniques on the performance of neural networks could further improve our overall effectiveness.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant

62376163; in part by the National Natural Science Foundation of China under Grant 62203308; in part by the Guangdong Regional Joint Foundation Key Project under Grant 2022B1515120076; and in part by the Shenzhen Science and Technology Program under Grant JCYJ20220531101411027. The work of Carlos A. Coello Coello was supported by CONACyT Grant (Investigación en Fronteras de la Ciencia 2016) under Grant 2016-01-1920.

CRediT authorship contribution statement

Xinzhi Zhang: Conceptualization, Methodology, Software, Writing – original draft. **Yeming Yang:** Conceptualization, Writing – original draft. **Qingling Zhu:** Conceptualization, Supervision, Writing-reviewing and editing. **Qiuzhen Lin:** Conceptualization, Supervision, Validation, Writing-reviewing and editing. **Weineng Chen:** Supervision, Validation. **Jianqiang Li:** Funding acquisition, Supervision. **Carlos A. Coello Coello:** Supervision, Validation.

References

[1] Edmund Burke, Peter Cowling, Patrick De Causmaecker, and Greet Vanden Berghe. A memetic approach to the nurse rostering problem. *Applied intelligence*, 15(3):199–214, 2001.

[2] Tse-Chiu Wong, Mai Xu, and Kwai-Sang Chin. A two-stage heuristic approach for nurse scheduling problem: A case study in an emergency department. *Computers & Operations Research*, 51:99–110, 2014.

[3] 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.

[4] Haroldo G Santos, Túlio AM Toffolo, Rafael AM Gomes, and Sabir Ribas. Integer programming techniques for the nurse rostering problem. *Annals of Operations Research*, 239:225–251, 2016.

[5] Broos Maenhout and Mario Vanhoucke. Branching strategies in a branch-and-price approach for a multiple objective nurse scheduling problem. *Journal of scheduling*, 13:77–93, 2010.

[6] Fang He and Rong Qu. A constraint programming based column generation approach to nurse rostering problems. *Computers & Operations Research*, 39(12):3331–3343, 2012.

[7] Uwe Aickelin and Kathryn A Dowsland. An indirect genetic algorithm for a nurse-scheduling problem. *Computers & operations research*, 31(5):761–778, 2004.

[8] Burak Bilgin, Peter Demeester, Mustafa Mısır, Wim Vancroonenburg, G 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.

[9] Uwe Aickelin and Jingpeng Li. An estimation of distribution algorithm for nurse scheduling. *Annals of Operations Research*, 155:289–309, 2007.

[10] Edmund K Burke, Timothy Curtois, Rong Qu, and Greet Vanden Berghe. A time predefined variable depth search for nurse rostering. *INFORMS Journal on Computing*, 25(3):411–419, 2013.

[11] Simone Zanda, Paola Zuddas, and Carla Seatzu. Long term nurse scheduling via a decision support system based on linear integer programming: a case study at the university hospital in cagliari. *Computers & Industrial Engineering*, 126:337–347, 2018.

[12] A Legrain and J Omer. A dedicated pricing algorithm to solve a large family of nurse scheduling problems with branch-and-price. *Les Cahiers du GERAD ISSN*, 711:2440, 2023.

[13] Alex M Andrew. Modern heuristic search methods. *Kybernetes*, 1998.

[14] Zhipeng Lü and Jin-Kao Hao. Adaptive neighborhood search for nurse rostering. *European Journal of Operational Research*, 218(3):865–876, 2012.

[15] Ioannis X Tassopoulos, Ioannis P Solos, and Grigorios N Beligiannis. A two-phase adaptive variable neighborhood approach for nurse rostering. *Computers & Operations Research*, 60:150–169, 2015.

[16] Antoine Legrain, Jérémy Omer, and Samuel Rosat. A rotation-based branch-and-price approach for the nurse scheduling problem. *Mathematical Programming Computation*, 12:417–450, 2020.

[17] Ebtisam Sharif, Masri Ayob, and Mohammed Hadwan. Hybridization of heuristic approach with variable neighborhood descent search to solve nurse rostering problem at universiti kebangsaan malaysia medical centre (ukmmc). In *2011 3rd Conference on Data Mining and Optimization (DMO)*, pages 178–183. IEEE, 2011.

[18] Ziran Zheng, Xiyu Liu, and Xiaoju Gong. A simple randomized variable neighbourhood search for nurse rostering. *Computers & Industrial Engineering*, 110:165–174, 2017.

[19] Federico Della Croce and Fabio Salassa. A variable neighborhood search based matheuristic for nurse rostering problems. *Annals of Operations Research*, 218(1):185–199, 2014.

[20] Say Leng Goh, Nasser R Sabar, Salwani Abdullah, Graham Kendall, et al. A 2-stage approach for the nurse rostering problem. *IEEE Access*, 10:69591–69604, 2022.

[21] Mohammad Reza Hassani and Javad Behnamian. A scenario-based robust optimization with a pessimistic approach for nurse rostering problem. *Journal of Combinatorial Optimization*, 41:143–169, 2021.

[22] Nagaraj V Dharwadkar, Gautami G Shingan, Sandeep Uttam Mane, and Santosh Joshi. Enhanced parallel-particle swarm optimization (ep-pso) approach for solving nurse rostering problem: Enhanced parallel-particle swarm optimization (ep-pso) algorithm. *International Journal of Swarm Intelligence Research (IJSIR)*, 13(1):1–17, 2022.

[23] Xuening Chang, Jianmai Shi, Zhihao Luo, and Yao Liu. Adaptive large neighborhood search algorithm for multi-stage weapon target assignment problem. *Computers & Industrial Engineering*, 181:109303, 2023.

[24] Nikolaos A Kyriakakis, Ioannis Sevastopoulos, Magdalene Marinaki, and Yannis Marinakis. A hybrid tabu search–variable neighborhood descent algorithm for the cumulative capacitated vehicle routing problem with time windows in humanitarian applications. *Computers & Industrial Engineering*, 164:107868, 2022.

[25] SU Zhouxing, WANG Zhuo, Lü Zhipeng, et al. Weighted tabu search for multi-stage nurse rostering problem. *Scientia Sinica Informationis*, 46(7):834–854, 2016.

[26] Celia A Glass and Roger A Knight. The nurse rostering problem: A critical appraisal of the problem structure. *European Journal of Operational Research*, 202(2):379–389, 2010.

[27] Sara Ceschia, Nguyen Dang, Patrick De Causmaecker, Stefaan Haspeslagh, and Andrea Schaerf. The second international nurse rostering competition. *Annals of Operations Research*, 274(1):171–186, 2019.

[28] Susana Fernandes and H Lourenço. Hybrids combining local search heuristics with exact algorithms. In *V Congreso Espanol sobre Metaheurısticas, Algoritmos Evolutivos y Bioinspirados*, pages 269–274, 2007.

[29] Erfan Rahimian, Kerem Akartunalı, and John Levine. A hybrid integer programming and variable neighbourhood search algorithm to solve nurse rostering problems. *European Journal of Operational Research*, 258(2):411–423, 2017.

[30] Aykut Melih Turhan and Bilge Bilgen. A hybrid fix-and-optimize and simulated annealing approaches for nurse rostering problem. *Computers & Industrial Engineering*, 145:106531, 2020.

[31] Fu Luo, Xi Lin, Fei Liu, Qingfu Zhang, and Zhenkun Wang. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. *Advances in Neural Information Processing Systems*, 36, 2024.

[32] SA ILOG. Ilog cplex 7.1 user's manual. *ILOG S. A: Gentilly, France*, 2001.

[33] Edmund K Burke and Tim Curtois. New approaches to nurse rostering benchmark instances. *European Journal of Operational Research*, 237(1):71–81, 2014.

[34] Jie-jun Wu, Ying Lin, Zhi-hui Zhan, Wei-neng Chen, Ying-biao Lin, and Jian-yong Chen. An ant colony optimization approach for nurse rostering problem. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1672–1676. IEEE, 2013.

[35] Frederik Knust and Lin Xie. Simulated annealing approach to nurse rostering benchmark and real-world instances. *Annals of Operations Research*, 272(1-2):187–216, 2019.

[36] Sara Ceschia, Rosita Guido, and Andrea Schaerf. Solving the static inrc-ii nurse rostering problem by simulated annealing based on large neighborhoods. *Annals of Operations Research*, 288(1):95–113, 2020.

[37] Razamin Ramli, Siti Nurin Ima Ahmad, Syariza Abdul-Rahman, and Antoni Wibowo. A tabu search approach with embedded nurse preferences for solving nurse rostering problem. *International Journal for Simulation and Multidisciplinary Design Optimization*, 11:10, 2020.

[38] David Meignan and Sigrid Knust. A neutrality-based iterated local search for shift scheduling optimization and interactive reoptimization. *European Journal of Operational Research*, 279(2):320–334, 2019.

[39] Toni I Wickert, Pieter Smet, and Greet Vanden Berghe. The nurse rerostering problem: Strategies for reconstructing disrupted schedules. *Computers & Operations Research*, 104:319–337, 2019.

[40] Zhenkun Wang, Shunyu Yao, Genghui Li, and Qingfu Zhang. Multiobjective combinatorial optimization using a single deep reinforcement learning model. *IEEE Transactions on Cybernetics*, 2023.

[41] Zhongju Yuan, Genghui Li, Zhenkun Wang, Jianyong Sun, and Ran Cheng. Rl-csl: a combinatorial optimization method using reinforcement learning and contrastive self-supervised learning. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2022.

[42] Wei Qin, Zilong Zhuang, Zizhao Huang, and Haozhe Huang. A novel reinforcement learning-based hyper-heuristic for heterogeneous vehicle routing problem. *Computers & Industrial Engineering*, 156:107252, 2021.

[43] Jingwen Li, Yining Ma, Ruize Gao, Zhiguang Cao, Andrew Lim, Wen Song, and Jie Zhang. Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem. *IEEE Transactions on Cybernetics*, 52(12):13572–13585, 2021.

[44] Zhi Zheng, Shunyu Yao, Genghui Li, Linxi Han, and Zhenkun Wang. Pareto improver: Learning improvement heuristics for multi-objective route planning. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[45] Yanjie Song, Luona Wei, Qing Yang, Jian Wu, Lining Xing, and Yingwu Chen. Rl-ga: A reinforcement learning-based genetic algorithm for electromagnetic detection satellite scheduling problem. *Swarm and Evolutionary Computation*, 77:101236, 2023.

[46] Rongkai Zhang, Cong Zhang, Zhiguang Cao, Wen Song, Puay Siew Tan, Jie Zhang, Bihan Wen, and Justin Dauwels. Learning to solve multiple-tsp with time window and rejections via deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[47] Rui Li, Wenyin Gong, Ling Wang, Chao Lu, and Chenxin Dong. Co-evolution with deep reinforcement learning for energy-aware distributed heterogeneous flexible job shop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2023.

[48] Xiao-long Chen, Jun-qing Li, and Ying Xu. Q-learning based multiobjective immune algorithm for fuzzy flexible job shop scheduling problem considering dynamic disruptions. *Swarm and Evolutionary Computation*, 83:101414, 2023.

[49] Yi Zhang, Haihua Zhu, Dunbing Tang, Tong Zhou, and Yong Gui. Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems. *Robotics and Computer-Integrated Manufacturing*, 78:102412, 2022.

[50] Anand J Kulkarni and Kang Tai. Probability collectives: a multi-agent approach for solving combinatorial optimization problems. *Applied Soft Computing*, 10(3):759–771, 2010.

[51] Maria Amélia Lopes Silva, Sergio Ricardo de Souza, Marcone Jamilson Freitas Souza, and Moacir Felizardo de Franca Filho. Hybrid metaheuristics and multi-agent systems for solving optimization problems: A review of frameworks and a comparative analysis. *Applied Soft Computing*, 71:433–459, 2018.

[52] Xiao-long Zheng and Ling Wang. A multi-agent optimization algorithm for resource constrained project scheduling problem. *Expert Systems with Applications*, 42(15-16):6039–6049, 2015.

[53] Maria Amélia Lopes Silva, Sérgio Ricardo de Souza, Marcone Jamilson Freitas Souza, and Ana Lúcia C Bazzan. A reinforcement learning-based multi-agent framework applied for solving routing and scheduling problems. *Expert Systems with Applications*, 131:148–171, 2019.

[54] David Meignan, Jean-Charles Créput, and Abderrafiâa Koukam. An organizational view of metaheuristics. In *First International Workshop on Optimisation in Multi-Agent Systems, AAMAS*, volume 8, pages 77–85. Citeseer, 2008.

[55] Mehmet Emin Aydin. Coordinating metaheuristic agents with swarm intelligence. *Journal of intelligent manufacturing*, 23:991–999, 2012.

[56] Sinaide Nunes Bezerra, Marcone Jamilson Freitas Souza, and Sérgio Ricardo de Souza. A variable neighborhood search-based algorithm with adaptive local search for the vehicle routing problem with time windows and multi-depots aiming for vehicle fleet reduction. *Computers & Operations Research*, 149:106016, 2023.

[57] Xun Zhou, A Kai Qin, Maoguo Gong, and Kay Chen Tan. A survey on evolutionary construction of deep neural networks. *IEEE Transactions on Evolutionary Computation*, 25(5):894–912, 2021.