

Deep Q-Learning Enhanced Variable Neighborhood Search for Influence Maximization in Social Networks

Afifeh Maleki Ghalghachi ^a, Mehdy Roayaei Ardakani ^{b*}

Department of Electrical and Computer Engineering, Tarbiat Modares University (TMU), Tehran, Iran;
m.afifeh@modares.ac.ir ^a, mroayaei@modares.ac.ir ^b

ABSTRACT

A social network consists of individuals and the relationships between them, which often influence each other. This influence can propagate behaviors or ideas through the network, a phenomenon known as influence propagation. This concept is crucial in applications like advertising, marketing, and public health. The influence maximization (IM) problem aims to identify key individuals in a social network who, when influenced, can maximize the spread of a behavior or idea. Given the NP-hard nature of IM, non-exact algorithms, especially metaheuristics, are commonly used. However, traditional metaheuristics like the variable neighborhood search (VNS) struggle with large networks due to vast solution spaces. This paper introduces DQVNS (Deep Q-learning Variable Neighborhood Search), which integrates VNS with deep reinforcement learning (DRL) to enhance neighborhood structure determination in VNS. By using DQVNS, we aim to achieve performance similar to population-based algorithms and utilize the information created step by step during the algorithm's execution. This adaptive approach helps the VNS algorithm choose the most suitable neighborhood structure for each situation and find better solutions for the IM problem. Our method significantly outperforms existing metaheuristics and IM-specific algorithms. DQVNS achieves a 63% improvement over population-based algorithms on various datasets. The results of implementation on different real-world social networks of varying sizes demonstrate the superiority of this algorithm compared to existing metaheuristic, IM-specific algorithms, and network-specific measures.


Keywords— Social Networks, Deep Reinforcement Learning, Influence Maximization, DQN.

1. Introduction

Social networks play an important role in disseminating information, thoughts, and ideas. A social network is a connected structure of elements formed for social interactions. Social influence occurs through the dissemination of information within the network. Nowadays, the analysis of social networks holds great importance in both theoretical and practical domains. This problem has been extensively studied in various fields, such as social sciences [1], psychology [2], marketing [3], and recommender systems [4]. One of the significant and common problems in social networks is identifying influential individuals within these networks, people who can exert the most influence on network members.

In social networks, the behavior of key players plays a significant role in analyzing these networks. There are various criteria for analyzing influential nodes in a social network. For example, clustering coefficient, density, different centrality measures, degree, page rank, and so on can be mentioned. Finding influential nodes (key players) in large-scale networks is a complex task because the size of these networks is constantly expanding.

A social network is defined as a graph $G(V, E)$ where V is the set of network nodes and E is the set of edges between them. The edge $(v_i, v_j) \in E$ in the network indicates the ability of node v_i to influence or activate node v_j , which occurs with a probability p determined by a specific propagation model that simulates how information spreads in the social

 <http://dx.doi.org/10.22133/ijwr.2024.459158.1219>

Citation A. Maleki Ghalghachi, M. Roayaei Ardakani, " Deep Q-Learning Enhanced Variable Neighborhood Search for Influence Maximization in Social Networks ", *International Journal of Web Research*, vol.7, no.2, pp.23-36, 2024, doi: <http://dx.doi.org/10.22133/ijwr.2024.459158.1219>.

*Corresponding Author

Article History: Received: 24 December 2023; Revised: 12 March 2024; Accepted: 18 March 2024.

Copyright © 2022 University of Science and Culture. Published by University of Science and Culture. This work is licensed under a Creative Commons Attribution-Noncommercial 4.0 International license (<https://creativecommons.org/licenses/by-nc/4.0/>). Noncommercial uses of the work are permitted, provided the original work is properly cited.

network. One of the most commonly used propagation models in literature is the Independent Cascade (IC) model, where a node can be active or inactive at any given moment. Initially, all nodes except those in the initial set (seed set) are inactive. Each active node (v_i) at time t has a chance to activate its inactive neighbor (v_j) with the probability of the edge weight between them. If v_i succeeds, then v_j will become an active node at time $t + 1$. This cascade process continues until no other active nodes emerge within a timestamp. The influence of a set S ($S \subseteq V$) is defined as the average number of activated nodes by set S , denoted by $I_G(S)$. The problem of finding influential nodes is thus defined as:

Influence Maximization (IM): Given a graph $G(V, E)$, the goal is to find a subset S with at most l nodes from the set V such that $I_G(S)$ is maximized.

Due to the NP-hard nature of the IM problem [5], non-exact methods such as metaheuristics are used to handle it. However, in the case of large networks, the solution space becomes excessively vast, posing a challenge for existing metaheuristics to yield satisfactory results. In such scenarios, it is crucial to explore innovative strategies or enhancements to existing algorithms to effectively tackle the complexity of IM in large-scale networks.

In this paper, we choose Variable Neighborhood Search (VNS) algorithm as a metaheuristic to enhance using deep reinforcement learning [6]. VNS is renowned for its ability to find high-quality solutions in optimization tasks, making it a suitable candidate for solving combinatorial optimization problems such as IM. Its adaptability allows for seamless integration with diverse problem domains, while its balance between exploration and exploitation ensures efficient navigation of the dynamic search space inherent in social networks. Moreover, VNS's flexibility in exploring different neighborhood structures enables it to escape local optima and discover diverse solutions, which is crucial for addressing the complexity of social network analysis.

VNS is a metaheuristic that sequentially explores different neighborhood structures to find new and diverse solutions. The main idea of the VNS algorithm is to define k_{max} different types of neighborhoods for a problem, and during the algorithm, these neighborhoods are used in sequence to improve the current solution. If the current neighborhood cannot improve the current solution, then the next neighborhoods are used sequentially. If the current neighborhood can improve the current solution, this solution is selected as the new current solution, and the search process continues from this solution, starting with first neighborhood structure (Figure 1).

```

Function VNS( $S, k_{max}, t_{max}$ ):
1  $S \leftarrow \text{Initial\_solution}()$ ;
2 repeat
3    $k \leftarrow 1$ ;
4   while  $k \leq k_{max}$  do
5      $S' \leftarrow \text{Shake}(S, k)$ ;           /* Shaking */
6      $S'' \leftarrow \text{LS}(S')$ ;           /* Local search */
7      $k \leftarrow k + 1$ ;               /* Next neighborhood */
8     if  $S''$  is better than  $S$  then
9        $S \leftarrow S''$ ;  $k \leftarrow 1$ ; /* Make a move */
    end
  end
10  $t \leftarrow \text{CpuTime}()$ ;
until  $t > t_{max}$ ;
11 Return  $S$ ;

```

Figure 1. VNS Basic Algorithm [7]

In this algorithm, k_{max} represents the maximum number of neighborhood structures, and t_{max} denotes the time condition for terminating the algorithm. The $\text{Shake}(S, k)$ function generates a random neighborhood for solution S based on k th neighborhood. Additionally, the $\text{LS}(S')$ function performs a local search to find the best local neighbor for the solution S' .

Metaheuristics offers flexible problem-solving methods that can be applied across various domains without requiring specialized knowledge. However, they frequently fail to fully exploit the data generated during their execution, which leaves opportunities for enhancing performance and efficiency, particularly in complex problems. They do not effectively utilize past experiences from previous iterations, which could greatly influence the selection of search strategies, behavioral adaptation, and parameter tuning. These overlooked possibilities have the potential to substantially improve convergence speed and the overall quality of the solutions generated.

Specifically, in VNS, a challenge is the order in which the defined neighborhoods are utilized in the search process. In the basic algorithm, the defined neighborhood structures are used sequentially, and if a neighborhood fails to produce an improvement in the current solution, the next neighborhood is used. A drawback of this approach is that VNS does not leverage the data generated in previous iterations to determine the appropriate order of neighborhood utilization. This means that considering the impact of different neighborhoods in previous search iterations, the search process could be enhanced, and at each iteration of VNS, priority should be given to the neighborhood structures that have shown better performance in previous iterations of the algorithm.

In this paper, deep reinforcement learning is utilized to tackle this challenge. Reinforcement learning (RL) is a category of machine learning methods in which agents interact with their environment, take actions, receive feedback (reward), and learn based on the rewards associated with each action. The goal of the proposed method is to learn the appropriate selection among different

neighborhood structures at each step of the search algorithm execution, leading to finding a better final solution in a shorter time.

Traditional RL algorithms often rely on state-action-value tables. These tables store the expected reward (value) for taking a specific action in a given state. This approach allows the agent to learn an optimal policy by selecting actions that maximize the expected future reward. However, maintaining and manipulating such tables becomes impractical in real-world scenarios with many possible states (i.e., high-dimensional state spaces). Deep learning offers a powerful alternative to state-action-value tables. It is possible for RL to learn value functions or policies directly from data by applying neural networks as function approximators.

Deep reinforcement learning (DRL) combines reinforcement learning with deep learning. In DRL, deep neural networks are integrated to enable agents to make decisions directly from unstructured input data without manual feature engineering. The algorithms handle significant inputs (such as video game screens) and optimize objectives (e.g., maximizing game scores). DRL has many applications in various fields, including robotics, video games, natural language processing, computer vision, education, transportation, finance, and healthcare.

DRL often encounters challenges related to learning efficiency and stability. An essential technique to address these issues is experience replay. Experience replay is a concept in Deep Q-Networks (DQN) where past experiences, such as states, actions, and rewards encountered during agent-environment interactions, are stored in a replay buffer. Experience replay does not discard data collected during agent-environment interactions; instead, it stores and utilizes this data for efficient learning.

During training, the DRL agent benefits from experience replay by sampling a diverse range of past interactions stored in the replay buffer. These experiences, encompassing state transitions, actions taken, and received rewards, are used to update the agent's decision-making strategy, represented by either the value function (e.g., Q-values) or the policy itself. The critical advantage of experience replay lies in the random sampling mechanism. This approach disrupts potential correlations that might exist within consecutive observations during training. Additionally, it smooths over any unexpected changes in the data distribution encountered by the agent. By learning from such a diverse array of experiences, the agent achieves greater stability and efficiency in its learning process.

The contributions of this paper can be summarized as follows:

- Modeling the neighborhood structure selection in VNS as a Markov Decision Process (MDP).
- Employing reinforcement learning to improve the process of neighborhood selection in VNS.
- Using deep reinforcement learning models to control the large state space of IM.
- Comparing the proposed method with other widely used algorithms on real-world network datasets.

The structure of the remainder of the paper is as follows: Section 2 reviews related works. Section 3 models the problem as an MDP. Section 4 presents the proposed method, called DQVNS. Section 5 evaluates the results on different datasets compared to previous methods. Finally, Section 6 concludes the paper.

2. Related Works

Kempe et al. [8] proved that IM is an NP-hard problem. Therefore, it is not possible to provide an exact polynomial-time algorithm for solving this problem. As a result, alternative methods such as approximation algorithms [5,9] and heuristic algorithms [10-15] have been proposed for solving this problem.

Also, various metaheuristic algorithms [16-19] have also been proposed to solve this problem. Among them, we can mention genetic algorithm [16,17], simulated annealing algorithm [18], particle swarm optimization algorithm [19], ant colony optimization algorithm [20], and bee colony algorithm [21]. Most of these algorithms are evolutionary algorithms, in which a solution or a set of initial solutions evolve over various iterations, and eventually, after the algorithm stops, the best generated solution is returned as the final solution.

One of the metaheuristic algorithms is the VNS algorithm, which attempts to improve the final solution for a problem based on selecting different neighborhood structures. Despite its simplicity, this algorithm produces excellent results and has been used in various studies to solve multiple problems, including facility location [22], finding the longest common subsequence [23], routing [24], and scheduling [25].

However, despite its usefulness, this algorithm has some shortcomings. Like many other algorithms, it does not utilize the data generated in previous iterations to enhance its performance. Specifically, it does not utilize the data generated in previous iterations to determine the order of using neighborhood structures in subsequent iterations. This selection is done non-intelligently and sequentially. However, considering the impact of different neighborhood structures in previous

iterations, it is possible to conduct the search process more intelligently and prioritize the use of neighborhood structures in each iteration of VNS based on the performance of these neighborhood structures in previous iterations.

To address this issue and improve this aspect of the VNS algorithm, this paper proposes a reinforcement learning-based approach using the deep Q-network (DQN), which selects an appropriate neighborhood structure in the current iteration that is likely to create better improvement in the current solution, based on the performance of different neighborhood structures in previous iterations.

In this section, studies that have utilized reinforcement learning to improve the neighborhood selection process in VNS are investigated. Additionally, only studies are referenced whose proposed method is independent of the specific problem, as the goal is to improve VNS as a problem-independent metaheuristic algorithm.

The VNS algorithm was introduced in 1997, and its main idea is to systematically change neighborhoods to find an optimal solution and escape local optima. Due to its simplicity, effectiveness, and versatility, this algorithm has been used in various fields such as network design [26], continuous optimization [27], job scheduling [28,29], and vehicle routing [30].

Dos Santos et al. [31] proposed one of the pioneering works to employ the Q-learning algorithm to enhance VNS. They proposed a method called Reactive Search to select an appropriate local search at each step of the search process. In this method, RL was utilized in two stages: initial solution selection and local search method selection. In their method, the action space consists of the same number of neighborhood structure, and the state at each step is the type of neighborhood structure used in the previous step.

Li et al. [32] presented a self-adaptive VNS algorithm. For each type of neighborhood structure, a probability is considered, and the algorithm, during the learning phase, increases or decreases the probability of selecting that type of neighborhood structure based on its performance in improving or not improving the current solution.

Todosijević et al. [33] assigned a score for each sequence of different neighborhood structures. If using that specific sequence of neighborhood structures results in an improvement in the current solution, the score of that sequence increases by a constant value; otherwise, it decreases by the same amount.

Shahrabi et al. [34] utilized the Q-learning algorithm to select estimated parameters of the VNS algorithm for solving the job shop scheduling

problem. In their proposed algorithm, the state space is defined based on problem features (the number of jobs and the average processing time of operations), comprising 20 different states. Additionally, the action space is based on VNS algorithm parameters (the number of VNS iterations, the maximum number of iterations in local search, and the improvement threshold in local search), comprising 8 different actions.

Thevenin et al. [35] introduced the LVNS method, where efforts were made to learn features that are often observed in good solutions during VNS execution and employ them in generating new neighbors.

Shahmardan et al. [36] used the Q-learning method to select appropriate neighborhood structures in different states in the simulated annealing algorithm. In this method, the state is equivalent to the number of times the current solution is not improved, and the action is the type of neighborhood structure selected by the agent.

Chen et al. [37] utilized a simple RL algorithm to learn the probability of using each neighborhood operator among 8 operators. In this algorithm, which applied to vehicle routing problem, the state represents the probability of selecting each neighborhood, and the action is the reduction or increase of these probabilities based on the improvement or non-improvement of the objective function.

Zhao et al. [38] used Q-learning-based for balancing exploration and exploitation in VNS for solving a scheduling problem. In this method, the state represents the current solution status, and the action represents the type of neighborhood structures.

Zhang et al. [39] employed DRL for learning a suitable strategy in selecting neighborhood structures in the facility location problem. In this approach, a part of the solution is defined as the state, and the selected neighborhood structures are considered as actions.

Alicastro et al [40] used the presented in [33] to improve the iterated local search metaheuristic. This method was applied to solve an additive manufacturing problem in 3D printers.

Gu et al. [41] utilized Q-learning for selecting appropriate neighborhood structures in the taboo search algorithm for solving the Max-mean dispersion problem. In this approach, the state represents the last element added to the current partial solution, and the action includes a set of valid elements that can be added to the current state.

Wang et al. [42] used Q-learning to enhance the search process in the bee colony algorithm for solving a scheduling problem. In their proposed method, the

state is defined based on the current status, and actions include various search and neighborhood operators.

Alrashidi et al. [43] offered an efficient approach combining VNS with RL to solve the Green Vehicle Routing Problem (GVRP). The state space includes route configurations and vehicle capacities, while the action space comprises route modifications. The reward signal reflects cost reduction.

Zhang et al. [44] addressed the distributed flow-shop scheduling problem (DFSP), aiming to minimize makespan and energy consumption. It introduced a Q-learning-based multi-objective particle swarm optimization (QL-MoPSO) algorithm. Enhanced PSO divides particles into subgroups for faster convergence, while Q-learning guides variable neighborhood search (VNS) for balanced exploration and exploitation.

Pugliese et al. [45] utilized Q-learning for the local search within VNS. The agent selects actions based on the current state and received rewards, aiming to maximize the cumulative reward. To boost exploration, a parameter ϵ determines whether the next action is selected randomly or based on the maximum value. This method allows dynamic adaptation of neighborhood structures as actions and states in the search process, promoting effective problem-solving strategies.

Overall, the main difference between the proposed method in this paper and the previous methods can be summarized as follows:

- In most of the existing methods, due to the limitation of the action space and state space, tabular methods like Q-learning have been utilized. In the proposed method, to improve the solution, better generalization in the state space and better learning, a deep reinforcement learning method called DQN (Deep Q-Network) is employed.
- In most of the existing methods, the similar and limited number of neighborhood structures have been used. To enhance the exploration of the solution space, six different exploration operators (equivalent to six different types of neighborhood structures) have been used to generate diverse and varied solutions.
- In most of the existing methods, the state is defined based on a single previous action or on a fixed sequence of neighborhood structures. However, such definitions do not facilitate finding an adaptive sequence of neighborhood structures that leads to better solutions. In the proposed method, to improve the learning of the effects of neighborhood operators at each step of the VNS algorithm, the state is considered as a sequence of operators performed in previous stages of the algorithm. This enables the selection of an

appropriate sequence of these neighborhood structures during algorithm execution.

- This method has been employed to solve the IM problem. This problem is inherently NP-hard, and due to the fact that the solution size is a very small subset of the network nodes, it has high complexity. Many types of neighborhood structures may not cause a change in the objective function, which adds to the complexity of the problem.

3. Formulating the problem as a Markov Decision Process (MDP)

RL is a subset of machine learning methods in which an agent attempts to learn a good policy through trial and error and interaction with the environment. The policy refers to the mapping of the current state to an appropriate action. In other words, the goal of the agent is to learn to map the current state to the appropriate action at each time step, maximizing the discounted sum of numerical rewards received from the environment, as illustrated in Figure 2.

In this approach, the agent is not told what action to take, but rather how good the selected action was. The selected actions not only affect instant rewards but also impact future states. When an agent takes an action a_t in state s_t at time t , it receives the outcome of its action with reward r_{t+1} at time $t + 1$ and transitions to state s_{t+1} .

Each problem in RL is defined using the Markov Decision Process (MDP) model and is structured as a quintuple $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$, where \mathcal{S} represents a finite state space, where $s_t \in \mathcal{S}$ indicates the state of an agent at time t . \mathcal{A} is a set of available actions for the agent, where $a_t \in \mathcal{A}$ denotes the action agent takes at time t . Also, $\mathcal{P}(s_t, a_t; s_{t+1}): \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$ is a Markovian transition function indicating the probability of the agent transitions from state s_t to state s_{t+1} after taking an action a_t . $R: \mathcal{S} \times \mathcal{A} \rightarrow R$ is a reward function that returns the immediate reward $R(s_t, a_t)$ after taking an action a_t

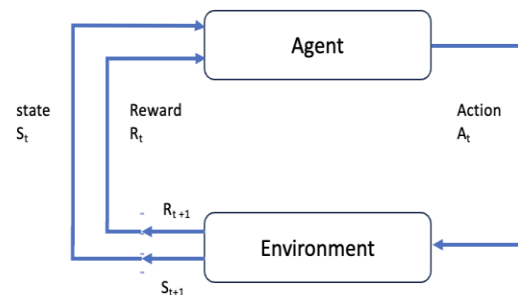


Figure 2. Main components of reinforcement learning

in state s_t . Additionally, $\gamma \in [0,1]$ is a discount factor indicating the rate of reward reduction over time. In a reinforcement learning problem, the goal is to maximize the total discounted rewards received by the agent (Eq. (1)).

$$r_1 + \gamma r_2 + \gamma^2 r_3 + \dots \quad (1)$$

where r_i is the reward received by the agent at time i .

The first step is to model the problem of selecting appropriate sequence of neighborhood structures in the VNS algorithm as an MDP model. In this modeling, the problem is defined as follows:

- **State:** An array with a length of n_s representing the previous actions used in the previous n_s iterations. In essence, the state represents a history of selected actions (neighborhood structures used) in previous iterations. In the proposed algorithm, $n_s = 6$.
- **Action:** The set of actions in the problem corresponds to the number of neighborhood structure defined in the VNS algorithm. In each iteration, depending on its current state, the agent selects one of the available neighborhood structures. The number of defined actions in this algorithm is set to $n_a = 6$.
- **Transition Function:** By selecting an action in each iteration, an attempt is made to produce better neighbors (solutions) for the current solution using the corresponding neighborhood structure. If this attempt is successful, the current solution is changed to the improved solution; otherwise, the current solution remains unchanged.
- **Reward:** The reward for each action is determined based on the difference between the value of the new solution generated by that action and the value of the current solution. If the new solution is better, the amount of this improvement is returned as a reward in the objective function; otherwise, a fixed value of -5 is returned as the reward.
- **Discount Factor:** The discount factor value is $\gamma = 0.95$.

4. Proposed Method

In this section, the proposed method (DQVNS) for solving IM using DRL-based VNS is presented. Each solution is represented as an array of length n (the number of graph vertices), where l elements (the number of influential nodes) take the value 1, and the remaining elements take the value 0.

The DQVNS has the following six types of neighborhood structures:

- **Crossover:** performs a crossover operation between the current solution and a random solution.
- **Mutation:** applies a mutation operation to the current solution.
- **PairSwap:** swaps the values of two elements in the current solution.
- **Inversion:** reverses the elements between two random positions in the array.
- **Insertion:** randomly inserts an element into the array and shifts the remaining elements to the right.
- **Displace:** selects a random subarray of the current solution, insert it at a random index, and shifts the remaining array to the right.

In selecting these six types of neighborhood structures, efforts have been made to choose operators that produce distinct and diverse solutions relative to each other. We will use the DQN algorithm [46], which is a combination of the Q-learning and deep neural networks. In the Q-learning algorithm [47], after performing each action, receiving a reward from the environment, and changing the state, the Q-value (the value of action a_t in state s_t) is updated according to Eq. (2):

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (2)$$

where α is the learning rate. In fact, $Q(s_t, a_t)$ is the average sum of rewards that the agent will receive from this state onwards and determines its value. In the DQN network, a deep neural network is used to approximate the value function Q. The architecture of this network for the proposed method is shown in Figure 3.

As shown in the Figure 3, an array with a length of n_s , representing the selected neighborhood structures in the previous n_s iterations, is provided as input to the network. The network consists of three fully connected hidden layers with a size of 24 and ReLU activation function. The output layer has a size of n_a , indicating the total number of neighborhood structures in the VNS algorithm, and has a Sigmoid activation function. The output layer returns the value of each neighborhood structures. The higher the value of a neighborhood structure, the more likely it will be selected for applying to the current solution. After this stage, an ϵ -greedy algorithm is used for action selection. This means that with a probability of $1 - \epsilon$, an action (neighborhood structure) with a higher value will be selected, and with a probability of ϵ , one of the other actions will be randomly selected.

After determining the action, the corresponding neighborhood structure is applied to the current solution, and the reward corresponding to the change in the objective function for this action is determined. Additionally, the current state is updated based on the selected action. The loss function for this network is

the difference between the objective value ($r_{t+1} + \gamma \max_a Q(s_{t+1}, a)$) and the estimated value by the network ($Q(s_t, a_t)$). Based on this loss function, the parameters of the network will be updated.

The proposed algorithm, as shown in Figure 4, is as follows. Here ϵ , α , and γ are parameters of the RL algorithm as introduced before, and max_it is the maximum number of iterations in the VNS algorithm, set to 100. Specifically, the innovation of this algorithm lies in line 4 (where DQN is used instead of Q-learning to control the large state space and better generalization), in line 6 (where the neighborhood structure is dynamically selected based on the current state of the algorithm), and in lines 14-15 (where the algorithm's state and DQN's state are adaptively updated to capture the performance of the algorithm in the current step). For a better understanding of the algorithm, its flowchart is presented in Figure 5. The novelty of the algorithm.

5. Results and Discussion

In this section, we compare the results of the proposed method (DQVNS) on a number of real-world networks with previous methods. The dataset

and their characteristics are shown in Table 1. In this implementation, the Independent Cascade (IC) model is used, where the probability of propagation in each network is specified in column p .

The ϵ -greedy algorithm used starts with an initial value of 1 for ϵ , which decreases during the algorithm execution by a factor of 0.99 until it reaches a minimum value of 0.1. The learning rate is set to $\alpha = 0.05$, and the discount factor is $\gamma = 0.95$. The results of the proposed algorithm on these six datasets are compared with well-known algorithms, which can be categorized as follows:

- **VNS**: The basic VNS algorithm.
- **Well-known metaheuristics** which have been applied on IM before including:
 - **GWO**: Grey Wolf Optimization Algorithm [51]
 - **GA**: Genetic Algorithm [52]
 - **PSO**: Particle Swarm Optimization Algorithm [53]
 - **ABC**: Artificial Bee Colony Algorithm [54]
 - **FF**: Firefly Optimization Algorithm [55]
 - **CS**: Cuckoo Search Algorithm [56]

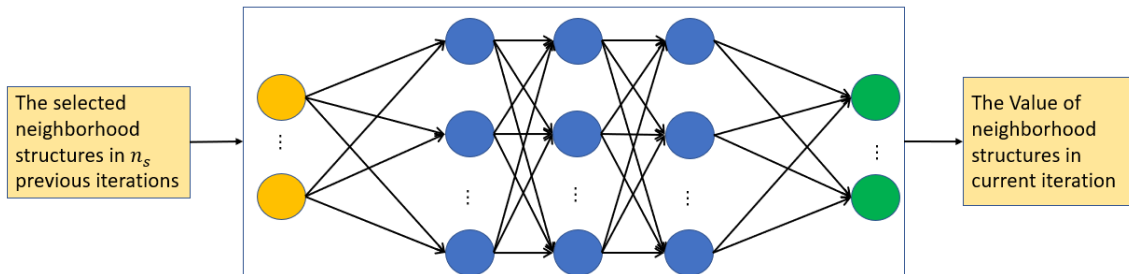


Figure 3. DQVNS Neural Network Architecture

Algorithm: DQVNS

```

1 Input:  $\epsilon, \alpha, \gamma, max\_it, n_s, n_A$ 
2 randomly initialize current solution S
3 initialize state
4 dqn= initialize DQN with parameters  $\epsilon, \alpha, \gamma, n_s, n_A$ 
5 For  $it : 1$  to  $max\_it$ :
6   action = select action using dqn and  $\epsilon$ -greedy algorithm given input state
7    $S' = Shake(S, action)$ 
8    $S'' = LS(S', action)$ 
9   If  $fitness(S'') > fitness(S)$ 
10     $S = S''$ 
11    reward =  $fitness(S'') - fitness(S)$ 
12   Else
13    reward = -5
14   nstate = update state
15   Update dqn w.r.t (state, action, nstate, reward)
16   state = nstate
17 return S

```

Figure 4. DQVNS Pseudo-code

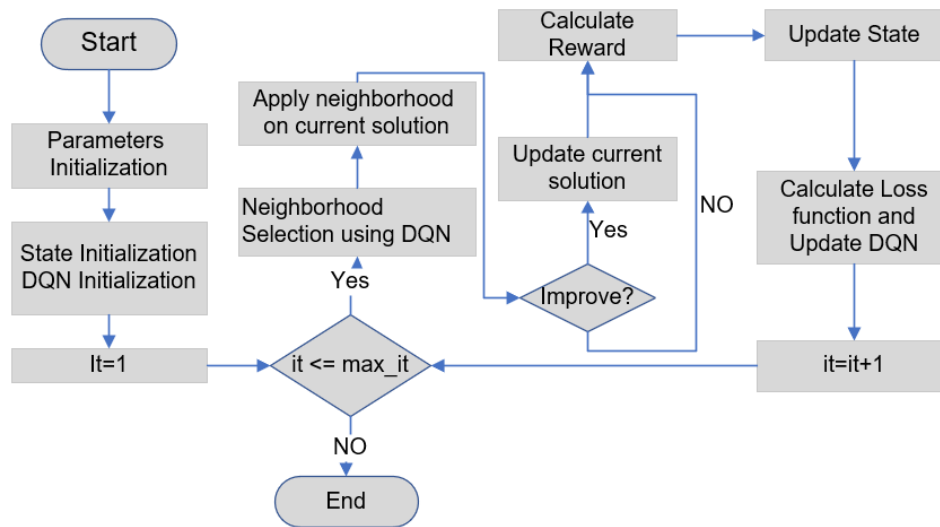


Figure. 5. DQVNS Flowchart

Table 1. Datasets Characteristics

#	dataset	p	#nodes	#edges	reference
1	HAM	0.03	2426	16631	[48]
2	EGO-Facebook	0.01	4039	88234	[48]
3	Wiki-votes	0.01	7115	103689	[49]
4	PGP	0.06	10680	24316	[50]
5	Hepph	0.1	12008	118521	[50]
6	NetHept	0.1	15223	31376	[50]

- **Centrality measures** including:
 - **Degree:** Degree centrality is a fundamental concept in network analysis. It measures the importance of a node within a network by counting the number of connections the node has to other nodes.
 - **Eigenvector:** This is another network analysis measure used to assess the influence or importance of a node within a network. Unlike degree centrality, which counts the number of connections, eigenvector centrality considers the quality of those connections. By considering not just the number of connections but also their influence, eigenvector centrality provides a more nuanced view of a node's importance within a network.
- **IM Methods:** IM-specific algorithms which have been applied on this problem including:
 - GWIM [57]
 - TOPSIS [58]
 - ID [59]
 - TI-SC [48]
 - PMC [50]

5.1. Influence Comparison

The results of comparing the proposed method with the mentioned algorithms are shown in Table 2. As evident in this table, The DQVNS algorithm demonstrated superior performance by achieving the highest influence scores in all six datasets.

Traditional metaheuristics often struggle to maintain an optimal balance between exploration and exploitation. DQVNS dynamically adjusts this balance through RL. This ensures that the algorithm effectively explores the search space while efficiently exploiting known-good solutions. Traditional metaheuristics rely on pre-defined heuristics and lack adaptive learning capabilities. These algorithms can become computationally expensive due to their reliance on population-based search strategies.

Unlike population-based algorithms, DQVNS maintains a low computational overhead by focusing on a single solution and enhancing it through intelligent learning mechanisms. This efficiency is crucial for large-scale datasets, where computational resources are a limiting factor. The proposed method integrates Deep Q-Learning, enabling it to adaptively refine its search strategy based on real-time feedback, thus improving performance over time. The consistent performance of DQVNS across diverse datasets highlights its robustness and generalizability. Whether dealing with small or large networks, DQVNS effectively maximizes influence, proving its applicability across different scenarios.

For the HAM dataset, DQVNS achieved an influence of 426, which is the best result among all algorithms. The second-best result was provided by GWO. The DQVNS algorithm outperforms all other algorithms by approximately at least 20%, which indicates its

superior efficiency. In the EGO-Facebook dataset, DQVNS achieved the highest influence score of 481, outperforming other algorithms, including GWO with a score of 464 and PSO with a score of 451.

For the Wiki-Votes dataset, DQVNS demonstrated a significant improvement in influence, achieving a score of 272. This score is notably higher than those of GA, which scored 161, and GWO, which scored 240. DQVNS also showed a 50% improvement over other population-based algorithms. The Eigenvector algorithm scored the lowest with a score of 50. The data in Table 2 indicate that all algorithms demonstrated generally poor performance when applied to the Wiki-Votes dataset, with a broad decrease in effectiveness. However, DQVNS exhibited relatively consistent performance across datasets, showing less sensitivity to the Wiki-Votes data compared to the other algorithms evaluated. For the PGP and NetHept datasets, DQVNS outperforms population-based algorithms by 25% and achieves a 50% improvement over PSO. In Hepph dataset, DQVNS achieved the highest influence score of 4789, while FF recorded the lowest score of 3991. In this case, our algorithm provided the best result.

Centrality measures for influence maximization, such as Degree and Eigenvector centrality, were also evaluated. These methods leverage graph-theoretic metrics to identify influential nodes in a network. These methods consistently scored lower than DQVNS because they are inherently static and do not adapt to the specific characteristics of the network during the influence maximization process. These rely on global graph properties that might not capture the nuanced interactions and dynamics of the network. These methods do not optimize, and nodes are ranked based on predefined metrics. Unlike DQVNS, these measures overlook more complex influence patterns that can be captured through optimization techniques. While these methods are simple and scalable, their simplicity limits their effectiveness in complex networks. DQVNS, on the other hand, employs sophisticated learning and

optimization techniques that can better handle the complexities of real-world networks.

The comparative analysis demonstrates the superiority of the DQVNS algorithm over traditional metaheuristics and statistical methods in influence maximization tasks. By integrating DQN with VNS, DQVNS effectively leverages adaptive learning, maintains computational efficiency, and achieves a balanced exploration-exploitation trade-off. By utilizing these capabilities, DQVNS consistently produces better results when applied to a variety of datasets, highlighting its effectiveness and robustness when applied to influence maximization in complex networks.

5.2. Running Time Comparison

To compare the running time of the algorithm with other methods, the population size in all methods is set to 100. Additionally, the running time is calculated for 100 iterations. As evident from the results presented in Table 3 **Error! Reference source not found.**, the proposed algorithm demonstrates significantly lower execution time compared to other algorithms, except VSN. This notable reduction in execution time can be attributed to the fact that, unlike other algorithms, the proposed method is not population-based. Population-based algorithms typically require considerable computational resources to manage and update multiple solutions simultaneously. In contrast, our algorithm focuses on a single solution, thus streamlining the computational process and reducing overhead.

The purpose of this comparison is to illustrate those intelligent mechanisms, when integrated into single-solution metaheuristics like VNS, can effectively enhance performance while preserving low running time. By leveraging these intelligent mechanisms, it is possible to achieve a balance where the algorithm operates efficiently and yields substantial results. This is particularly important in scenarios where computational efficiency is critical, such as real-time applications or large-scale optimization problems. Given the significant

Table 2. Influence Results

Algorithm /dataset	VNS	Degree	Eigenvector	IM Methods	CS	FF	ABC	PSO	GA	GWO	DQVNS
HAM	414	286	281	359 (ref. [58])	406	353	353	333	353	416	426
EGO-Facebook	460	370	251	383 (ref. [48])	455	449	438	451	324	464	481
Wiki-votes	252	246	50	250 (ref. [59])	208	186	174	122	161	240	272
PGP	605	429	331	530(ref. [57])	586	524	492	418	465	624	634
Hepph	4662	3986	3954	4410 (ref. [50])	4113	3991	4103	4118	3995	4499	4789
NetHept	1080	800	736	980 (ref. [50])	1020	885	939	702	840	1080	1119

reduction in running time without compromising on solution quality, it is reasonable to conclude that single- solution approaches provide a viable and efficient alternative to traditional population-based methods when augmented with adaptive and dynamic enhancements. This makes them highly suitable for practical applications in influence maximization and other complex optimization problems.

5.3. Scalability Comparison

Also, to compare the scalability of this algorithm with other algorithms, the growth of running time based on the dataset size (number of nodes) is shown in Figure 6. However, since besides the number of nodes, other network characteristics (such as the number of edges, network topology, etc.) also affect the running time of algorithms, the running time may not necessarily increase proportionally with the increase in the number of nodes. However, as shown, the DQVNS algorithm has better scalability compared to other algorithms (comparable to VSN), meaning that with an increase in the size of the input network, its running time increases with less slope.

5.4. VNS and DQVNS comparison

IM is crucial for optimizing strategies in networked systems such as social networks. The results are compared against various traditional metaheuristic algorithms, demonstrating the superiority of VNS and the further optimization

achieved by DQVNS. VNS and DQVNS outperformed traditional metaheuristics across most datasets. As a result of leveraging the strengths of VNS and enhancing them through RL, DQVNS achieved the highest influence scores. The VNS employs a multi-neighborhood search strategy in order to explore diverse regions of the search space effectively. This capability helps escape local optima and find better solutions, making it superior to single-neighborhood or less adaptive metaheuristics. By integrating Deep Q-Learning, DQVNS further enhances VNS performance. RL enables the algorithm to learn optimal strategies dynamically, improving the search process and resulting in better influence scores. For example, DQVNS improved the score from 460 (VNS) to 481 on the EGO-Facebook dataset.

DQVNS benefits from intelligent neighborhood selection guided by RL. This strategic selection helps focus the search on promising areas, enhancing the algorithm's efficiency and effectiveness. The comparative analysis highlights the effectiveness of VNS in influence maximization due to its multi-neighborhood search strategy, making it superior to other traditional metaheuristic algorithms. Further optimization through Deep Q-Learning significantly enhances VNS performance, leveraging reinforcement learning to select optimal neighborhoods and refine search strategies. This

Table 3. Running Time Results

<i>Algorithm /dataset</i>	<i>VNS</i>	<i>CS</i>	<i>FF</i>	<i>ABC</i>	<i>PSO</i>	<i>GA</i>	<i>GWO</i>	<i>DQVNS</i>
HAM	179	2004	7417	1512	1551	1545	910	323
EGO-Facebook	371	3218	13221	2619	2785	2411	2200	605
Wiki-votes	667	5412	23440	5296	5305	4690	4419	869
PGP	1085	8012	37500	7053	7032	6200	6032	1463
Hepph	1313	10561	50500	10204	10134	9353	9003	1913
NetHept	1429	11087	53500	10976	10929	8890	8835	2112

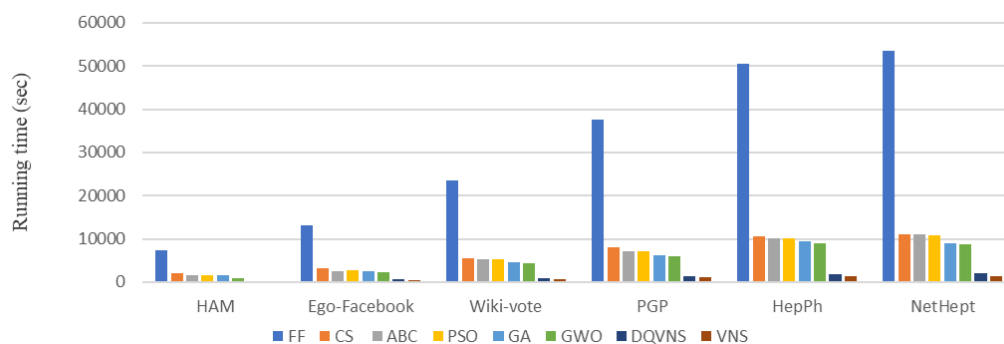


Figure 6. Scalability Results

combination not only outperforms traditional and population-based metaheuristics but also maintains computational efficiency, demonstrating the robustness and superiority of DQVNS for influence maximization tasks in complex networks.

In this section, the running times of the VNS algorithm are compared with its enhanced version, DQVNS, against traditional population-based metaheuristics. The results demonstrate how DQVNS retains the low running time of single-solution methods while offering significant performance improvements. Population-based algorithms are typically chosen for influence maximization tasks because they explore a broader solution space by maintaining multiple solutions simultaneously. This can be advantageous for finding influential nodes and maximizing the spread of influence but at the cost of higher running times. Our proposed method intelligently selects the most promising neighborhoods to explore. This adaptive enhancement improves the quality of the solutions while maintaining low computational costs. Despite the increase in running time compared to VNS, DQVNS's running time is still substantially lower than that of population-based methods. The DQVNS algorithm balances the efficiency of single-solution approaches and the comprehensive search capabilities of population-based methods. This makes it a viable influence maximization alternative, offering high performance and low computational overhead.

6. Conclusion and Future Work

The paper introduces a deep reinforcement learning-based method to enhance neighborhood structure selection in the VNS metaheuristic algorithm, which calculates the probability of selecting each neighborhood structure for the current iteration based on the performance of different neighborhood structures in previous iterations of the algorithm.

To evaluate the proposed algorithm, it was applied to the complex combinatorial problem of IM in social networks. The results obtained from this algorithm on various datasets of different sizes and the comparison of these results with various commonly used methods demonstrate the satisfactory performance of the proposed method in terms of accuracy and runtime.

The DQVNS algorithm demonstrates significant improvements in solving the IM problem; however, there are limitations to consider. Firstly, the algorithm was applied only to specific datasets, which may not fully capture the complexity of real-world social networks. As a result, generalizing these findings to networks with different structures remains uncertain.

Furthermore, integrating DRL with VNS introduces computational complexity. Although the algorithm's running time was manageable on the tested datasets and was even less than other population-based algorithms, the time complexity may increase when applied to datasets with different structures.

Another important consideration is that, although the proposed method was applied to the VNS algorithm in this paper, the underlying idea of intelligent neighborhood structure selection has broader applicability. This concept could be extended to other metaheuristic and state-space search methods, potentially enhancing their performance as well.

For future research, several suggestions are proposed. The first suggestion is to use RL to improve the performance of other metaheuristic algorithms. The second suggestion is to utilize RL to enhance other aspects besides neighborhood structure selection in metaheuristic algorithms. Additionally, the last suggestion is to employ multi-agent RL to enhance population-based search algorithms.

Declarations

Funding

This research did not receive any grant from funding agencies in the public, commercial, or non-profit sectors.

Authors' contributions

A. M. Ghalghachi: Study design, interpretation of the results, statistical analysis, drafting the manuscript;

M. R.: Study design, Conceptualization, interpretation of the results, statistical analysis, revision of the manuscript.

Conflict of interest

The authors declare that no conflicts of interest exist.

References

- [1] B. Razaghi, M. Roayaei, and N. M. Charkari, "On the Group-Fairness-Aware Influence Maximization in Social Networks," *IEEE Trans Comput Soc Syst*, vol. 10, no. 6, pp. 3406–3414, Dec. 2023, <https://doi.org/10.1109/TCSS.2022.3198096>.
- [2] H. Li, S. S. Bhowmick, A. Sun, and J. Cui, "Conformity-aware influence maximization in online social networks," *The VLDB Journal*, vol. 24, no. 1, pp. 117–141, Feb. 2015, <https://doi.org/10.1007/s00778-014-0366-x>.
- [3] H. Huang, H. Shen, Z. Meng, H. Chang, and H. He, "Community-based influence maximization for viral marketing," *Applied Intelligence*, vol. 49, no. 6, pp. 2137–2150, Jun. 2019, <https://doi.org/10.1007/s10489-018-1387-8>.
- [4] F. Coró, G. D'angelo, and Y. Velaj, "Link Recommendation for Social Influence Maximization," *ACM Trans Knowl Discov Data*, vol. 15, no. 6, pp. 1–23, Jun. 2021, <https://doi.org/10.1145/3449023>.

- [5] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining -KDD '03*, 2003, p. 137-146. <https://doi.org/10.1145/956750.956769>.
- [6] P. Hansen, N. Mladenović, J. Brimberg, and J. A. M. Pérez, "Variable neighborhood search," in *International Series in Operations Research and Management Science*, vol. 272, Springer New York LLC, 2019, pp. 57-97. https://doi.org/10.1007/978-3-319-91086-4_3.
- [7] N. Mladenović, A. Alkandari, J. Pei, R. Todosijević, and P. M. Pardalos, "Less is more approach: basic variable neighborhood search for the obnoxious p-median problem," *International Transactions in Operational Research*, vol. 27, no. 1, pp. 480-493, Jan. 2020, <https://doi.org/10.1111/itor.12646>.
- [8] D. Kempe, J. Kleinberg, and É. Tardos, "Influential Nodes in a Diffusion Model for Social Networks," In *Automata, Languages and Programming: 32nd International Colloquium, ICALP 2005*, Berlin Heidelberg, Springer, 2005, pp. 1127-1138. https://doi.org/10.1007/11523468_91.
- [9] H. Nguyen and R. Zheng, "On Budgeted Influence Maximization in Social Networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 6, pp. 1084-1094, Jun. 2013, <https://doi.org/10.1109/JSAC.2013.130610>.
- [10] C. Wilson, A. Sala, K. P. N. Puttaswamy, and B. Y. Zhao, "Beyond Social Graphs," *ACM Transactions on the Web*, vol. 6, no. 4, pp. 1-31, Nov. 2012, <https://doi.org/10.1145/2382616.2382620>.
- [11] B. M. Tabak, M. Takami, J. M. C. Rocha, D. O. Cajueiro, and S. R. S. Souza, "Directed clustering coefficient as a measure of systemic risk in complex banking networks," *Physica A: Statistical Mechanics and its Applications*, vol. 394, pp. 211-216, Jan. 2014, <https://doi.org/10.1016/j.physa.2013.09.010>.
- [12] S. Brin and L. Page, "Reprint of: The anatomy of a large-scale hypertextual web search engine," *Computer Networks*, vol. 56, no. 18, pp. 3825-3833, Dec. 2012, <https://doi.org/10.1016/j.comnet.2012.10.007>.
- [13] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, USA: ACM, Jun. 2009, pp. 199-208. <https://doi.org/10.1145/1557019.1557047>.
- [14] A. Goyal, W. Lu, and L. V. S. Lakshmanan, "SIMPACT: An Efficient Algorithm for Influence Maximization under the Linear Threshold Model," in *2011 IEEE 11th International Conference on Data Mining*, Vancouver, BC, Canada, IEEE, Dec. 2011, pp. 211-220. <https://doi.org/10.1109/ICDM.2011.132>.
- [15] R. Narayanam and Y. Narahari, "A Shapley Value-Based Approach to Discover Influential Nodes in Social Networks," *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 1, pp. 130-147, Jan. 2011, <https://doi.org/10.1109/TASE.2010.2052042>.
- [16] D. Bucur and G. Iacca, "Influence Maximization in Social Networks with Genetic Algorithms," 2016, pp. 379-392. https://doi.org/10.1007/978-3-319-31204-0_25.
- [17] C.-W. Tsai, Y.-C. Yang, and M.-C. Chiang, "A Genetic NewGreedy Algorithm for Influence Maximization in Social Network," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, Hong Kong, China, IEEE, Oct. 2015, pp. 2549-2554. <https://doi.org/10.1109/SMC.2015.446>.
- [18] D. Li, C. Wang, S. Zhang, G. Zhou, D. Chu, and C. Wu, "Positive influence maximization in signed social networks based on simulated annealing," *Neurocomputing*, vol. 260, pp. 69-78, Oct. 2017, <https://doi.org/10.1016/j.neucom.2017.03.003>.
- [19] M. Gong, J. Yan, B. Shen, L. Ma, and Q. Cai, "Influence maximization in social networks based on discrete particle swarm optimization," *Inf Sci (N Y)*, vol. 367-368, pp. 600-614, Nov. 2016, <https://doi.org/10.1016/j.ins.2016.07.012>.
- [20] S. S. Singh, K. Singh, A. Kumar, and B. Biswas, "ACO-IM: maximizing influence in social networks using ant colony optimization," *Soft comput*, vol. 24, no. 13, pp. 10181-10203, Jul. 2020, <https://doi.org/10.1007/s00500-019-04533-y>.
- [21] R. Cantini, F. Marozzo, S. Mazza, D. Talia, and P. Trunfio, "A Weighted Artificial Bee Colony algorithm for influence maximization," *Online Soc Netw Media*, vol. 26, p. 100167, Nov. 2021, <https://doi.org/10.1016/j.osnem.2021.100167>.
- [22] S. Rahdar, R. Ghanbari, and K. Ghorbani-Moghadam, "Tabu search and variable neighborhood search algorithms for solving interval bus terminal location problem," *Appl Soft Comput*, vol. 116, p. 108367, Feb. 2022, <https://doi.org/10.1016/j.asoc.2021.108367>.
- [23] M. Djukanović, A. Kartelj, D. Matić, M. Grbić, C. Blum, and G. R. Raidl, "Graph search and variable neighborhood search for finding constrained longest common subsequences in artificial and real gene sequences," *Appl Soft Comput*, vol. 122, p. 108844, Jun. 2022, <https://doi.org/10.1016/j.asoc.2022.108844>.
- [24] P. Kalatzantonakis, A. Sifaleras, and N. Samaras, "A reinforcement learning-Variable neighborhood search method for the capacitated Vehicle Routing Problem," *Expert Syst Appl*, vol. 213, p. 118812, Mar. 2023, <https://doi.org/10.1016/j.eswa.2022.118812>.
- [25] X. Zhang and L. Chen, "A general variable neighborhood search algorithm for a parallel-machine scheduling problem considering machine health conditions and preventive maintenance," *Comput Oper Res*, vol. 143, p. 105738, Jul. 2022, <https://doi.org/10.1016/j.cor.2022.105738>.
- [26] H. Amrani, A. Martel, N. Zufferey, and P. Makeeva, "A variable neighborhood search heuristic for the design of multicommodity production-distribution networks with alternative facility configurations," *OR Spectrum*, vol. 33, no. 4, pp. 989-1007, Oct. 2011, <https://doi.org/10.1007/s00291-009-0182-7>.
- [27] M. Bierlaire, M. Thémans, and N. Zufferey, "A Heuristic for Nonlinear Global Optimization," *INFORMS J Comput*, vol. 22, no. 1, pp. 59-70, Feb. 2010, <https://doi.org/10.1287/ijoc.1090.0343>.
- [28] B. F. Rosa, M. J. F. Souza, S. R. de Souza, M. F. de França Filho, Z. Ales, and P. Y. P. Michelon, "Algorithms for job scheduling problems with distinct time windows and general earliness/tardiness penalties," *Comput Oper Res*, vol. 81, pp. 203-215, May 2017, <https://doi.org/10.1016/j.cor.2016.12.024>.
- [29] C.-J. Liao and C.-C. Cheng, "A variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date," *Comput Ind Eng*, vol. 52, no. 4, pp. 404-413, May 2007, <https://doi.org/10.1016/j.cie.2007.01.004>.
- [30] A. Stenger, D. Vigo, S. Enz, and M. Schwind, "An Adaptive Variable Neighborhood Search Algorithm for a Vehicle Routing Problem Arising in Small Package Shipping," *Transportation Science*, vol. 47, no. 1, pp. 64-80, Feb. 2013, <https://doi.org/10.1287/trsc.1110.0396>.
- [31] J. P. Queiroz dos Santos, J. D. de Melo, A. D. Duarte Neto, and D. Aloise, "Reactive Search strategies using Reinforcement Learning, local search algorithms and Variable Neighborhood Search," *Expert Syst Appl*, vol. 41,

- no. 10, pp. 4939–4949, Aug. 2014, <https://doi.org/10.1016/j.eswa.2014.01.040>.
- [32] K. Li and H. Tian, “A two-level self-adaptive variable neighborhood search algorithm for the prize-collecting vehicle routing problem,” *Appl Soft Comput*, vol. 43, pp. 469–479, Jun. 2016, <https://doi.org/10.1016/j.asoc.2016.02.040>.
- [33] R. Todosijević, M. Mladenović, S. Hanafi, N. Mladenović, and I. Crévits, “Adaptive general variable neighborhood search heuristics for solving the unit commitment problem,” *International Journal of Electrical Power & Energy Systems*, vol. 78, pp. 873–883, Jun. 2016, <https://doi.org/10.1016/j.ijepes.2015.12.031>.
- [34] J. Shahrabi, M. A. Adibi, and M. Mahootchi, “A reinforcement learning approach to parameter estimation in dynamic job shop scheduling,” *Comput Ind Eng*, vol. 110, pp. 75–82, Aug. 2017, <https://doi.org/10.1016/j.cie.2017.05.026>.
- [35] S. Thevenin and N. Zufferey, “Learning Variable Neighborhood Search for a scheduling problem with time windows and rejections,” *Discrete Appl Math (1979)*, vol. 261, pp. 344–353, May 2019, <https://doi.org/10.1016/j.dam.2018.03.019>.
- [36] A. Shahmardan and M. S. Sajadih, “Truck scheduling in a multi-door cross-docking center with partial unloading – Reinforcement learning-based simulated annealing approaches,” *Comput Ind Eng*, vol. 139, p. 106134, Jan. 2020, <https://doi.org/10.1016/j.cie.2019.106134>.
- [37] B. Chen, R. Qu, R. Bai, and W. Laesanklang, “A variable neighborhood search algorithm with reinforcement learning for a real-life periodic vehicle routing problem with time windows and open routes,” *RAIRO - Operations Research*, vol. 54, no. 5, pp. 1467–1494, Sep. 2020, <https://doi.org/10.1051/ro/2019080>.
- [38] F. Zhao, L. Zhang, J. Cao, and J. Tang, “A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem,” *Comput Ind Eng*, vol. 153, p. 107082, Mar. 2021, <https://doi.org/10.1016/j.cie.2020.107082>.
- [39] Z. Zhang, Z. Huang, and L. Zou, “Neighborhood Search Acceleration Based on Deep Reinforcement Learning for SSCFLP,” in *Artificial Intelligence: First CAAI International Conference, CICA 2021*, Hangzhou, China, Springer International Publishing, June 5–6, 2021, pp. 202–212. https://doi.org/10.1007/978-3-030-93046-2_18.
- [40] M. Alicastro, D. Ferone, P. Festa, S. Fugaro, and T. Pastore, “A reinforcement learning iterated local search for makespan minimization in additive manufacturing machine scheduling problems,” *Comput Oper Res*, vol. 131, p. 105272, Jul. 2021, <https://doi.org/10.1016/j.cor.2021.105272>.
- [41] X. Gu, S. Zhao, and Y. Wang, “Reinforcement learning enhanced multi-neighborhood tabu search for the max-mean dispersion problem,” *Discrete Optimization*, vol. 44, p. 100625, May 2022, <https://doi.org/10.1016/j.disopt.2021.100625>.
- [42] J. Wang, D. Lei, and J. Cai, “An adaptive artificial bee colony with reinforcement learning for distributed three-stage assembly scheduling with maintenance,” *Appl Soft Comput*, vol. 117, p. 108371, Mar. 2022, <https://doi.org/10.1016/j.asoc.2021.108371>.
- [43] M. Alrashidi and M. A. Ghamdi, “Variable Neighborhood Search Based on Reinforcement Learning for Green Vehicle Routing Problem,” *2024 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*, Osaka, Japan, 2024, pp. 530–537, <https://doi.org/10.1109/ICAIIIC60209.2024.10463347>.
- [44] W. Zhang, H. Geng, C. Li, M. Gen, G. Zhang, and M. Deng, “Q-learning-based multi-objective particle swarm optimization with local search within factories for energy-efficient distributed flow-shop scheduling problem,” *Journal of Intelligent Manufacturing*, Oct. 2023, <https://doi.org/10.1007/s10845-023-02227-9>.
- [45] L. D. P. Pugliese, D. Ferone, P. Festa, F. Guerriero, and G. Macrina, “Combining variable neighborhood search and machine learning to solve the vehicle routing problem with crowd-shipping,” *Optim. Lett.*, vol. 17, no. 9, pp. 1981–2003, Dec. 2023, <https://doi.org/10.1007/s11590-021-01833-x>.
- [46] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, <https://doi.org/10.1038/nature14236>.
- [47] J. Clifton and E. Laber, “Q-Learning: Theory and Applications,” *Annu Rev Stat Appl*, vol. 7, no. 1, pp. 279–301, Mar. 2020, <https://doi.org/10.1146/annurev-statistics-031219-041220>.
- [48] H. A. Beni and A. Bouyer, “TI-SC: top-k influential nodes selection based on community detection and scoring criteria in social networks,” *J Ambient Intell Humaniz Comput*, vol. 11, no. 11, pp. 4889–4908, Nov. 2020, <https://doi.org/10.1007/s12652-020-01760-2>.
- [49] D. Bucur and G. Iacca, “Influence Maximization in Social Networks with Genetic Algorithms,” in *Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016*, Porto, Portugal, Springer International Publishing, 2016, pp. 379–392. https://doi.org/10.1007/978-3-319-31204-0_25.
- [50] A. Arora, S. Galhotra, and S. Ranu, “Debunking the Myths of Influence Maximization,” in *Proceedings of the 2017 ACM International Conference on Management of Data*, New York, NY, USA: ACM, May 2017, pp. 651–666. <https://doi.org/10.1145/3035918.3035924>.
- [51] M. Roayaci, “On the binarization of Grey Wolf optimizer: a novel binary optimizer algorithm,” *Soft comput*, vol. 25, no. 23, pp. 14715–14728, Dec. 2021, <https://doi.org/10.1007/s00500-021-06282-3>.
- [52] Md. M. Kabir, Md. Shahjahan, and K. Murase, “A new local search based hybrid genetic algorithm for feature selection,” *Neurocomputing*, vol. 74, no. 17, pp. 2914–2928, Oct. 2011, <https://doi.org/10.1016/j.neucom.2011.03.034>.
- [53] R. Bello, Y. Gomez, A. Nowe, and M. M. Garcia, “Two-Step Particle Swarm Optimization to Solve the Feature Selection Problem,” in *Seventh International Conference on Intelligent Systems Design and Applications (ISDA 2007)*, IEEE, Oct. 2007, pp. 691–696. <https://doi.org/10.1109/ISDA.2007.101>.
- [54] C. J. Santana, M. Macedo, H. Siqueira, A. Gokhale, and C. J. A. Bastos-Filho, “A novel binary artificial bee colony algorithm,” *Future Generation Computer Systems*, vol. 98, pp. 180–196, Sep. 2019, <https://doi.org/10.1016/j.future.2019.03.032>.
- [55] K. K. Bhattacharjee and S. P. Sarmah, “A binary firefly algorithm for knapsack problems,” in *2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, IEEE, Dec. 2015, pp. 73–77. <https://doi.org/10.1109/IEEM.2015.7385611>.
- [56] Y. Kaya, “Feature selection using binary cuckoo search algorithm,” in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, Izmir, Turkey, IEEE, May 2018, pp. 1–4. <https://doi.org/10.1109/SIU.2018.8404843>.
- [57] A. Zareie, A. Sheikhamadi, and M. Jalili, “Identification of influential users in social network using gray wolf optimization algorithm,” *Expert Syst Appl*, vol. 142, p. 112971, Mar. 2020, <https://doi.org/10.1016/j.eswa.2019.112971>.
- [58] A. Zareie, A. Sheikhamadi, and K. Khamforoosh, “Influence maximization in social networks based on

TOPSIS,” *Expert Syst Appl*, vol. 108, pp. 96–107, Oct. 2018, <https://doi.org/10.1016/j.eswa.2018.05.001>.

- [59] A.-S. T. Olanrewaju, R. Ahmad, and M. Mahmudin, “Influence Maximization Towards Target Users on Social Networks for Information Diffusion,” In *Recent Trends in Information and Communication Technology: Proceedings of the 2nd International Conference of Reliable Information and Communication Technology (IRICT 2017)*, Springer International Publishing, 2018, pp. 842–850. https://doi.org/10.1007/978-3-319-59427-9_87.



Mehdy Roayaei received his B.S., M.S., and Ph.D. in Computer Engineering from Amirkabir University of Technology (AUT) in 2008, 2010, 2016. He is currently an Assistant Professor of Computer Engineering at Tarbiat Modares University. He is interested in using reinforcement learning approaches for handling complex problems in real-world environments.



Afifeh Maleki Ghalghachi is a Master's student at Tarbiat Modares University. Her research focuses on using reinforcement learning to enhance metaheuristic algorithms.