Hindawi
Applied Computational Intelligence and Soft Computing
Volume 2023, Article ID 9640807, 27 pages
https://doi.org/10.1155/2023/9640807

Research Article

Chemical Reaction Optimization for Minimum Weight Dominating Set

A. Pritam Khan Boni and B. Md. Rafiqul Islam

Khulna University, Khulna, Bangladesh

Correspondence should be addressed to A. Pritam Khan Boni; pritamku130218@gmail.com

Received 1 July 2022; Revised 16 December 2022; Accepted 20 December 2022; Published 11 January 2023

Academic Editor: Upaka Rathnayake

Dominating set of a graph can be defined as the set of vertices that can cover all other vertices of the graph. The minimum weight dominating set (MWDS) is the minimum number of vertices in the dominating set with minimum total weight. In recent times, the chemical reaction optimization algorithm (CRO) has shown its supremacy in solving these types of problems. Therefore in this paper, a novel approach based on CRO has been proposed to solve the MWDS problem. The proposed method uses a repair-based technique to generate a molecule. To make the solution feasible by covering all vertices and to get better results, three supporting operators are implemented along with the CRO operators. Besides this, two repair operators are introduced. In the first repair operator, the searching procedure works based on the scaling properties of vertices, and the second one is a unique method for eliminating common neighbors of vertices of the dominating set. The performance of the proposed method is better than any other existing related algorithms. The performance is measured from different graphs of the benchmark datasets. It can be mentioned that the proposed method takes minimal running time to obtain the minimum weight compared to other benchmark methods.

1. Introduction

In graph theory, the dominating set of a graph is the set of vertices that are either members or neighbors of the member vertices. If in dominating set the number of vertices is minimum, then it is called the minimum dominating set (MDS). A minimum weight dominating set (MWDS) is a vulgarized version of the minimum dominating set (MDS). Finding the minimum dominating set with the minimum total weight of the vertices can be defined as the MWDS problem. Here, weight is an associated positive value for every vertex of the graph.

The MWDS problem is a rising research topic nowadays due to its immense application area. The study of social networks and influence propagation [1], protein interaction networks [2, 3], design of wireless sensor networks [4], and covering codes [5] are the application areas where the concept of MWDS is used. To construct a backbone for ad-hoc wireless networks, Wang et al. [6] proposed a centralized and distributed algorithm for the MWDS problem. On the other hand, to generate an MWDS for a unit disk graph (UDG), a $(5+\epsilon)$-approximation algorithm was proposed [7]. However, for the MWDS problem, a $(4+\epsilon)$-approximation dynamic programming algorithm has been suggested, which is an enormous upsurge as the approximation factor for UDG without smooth weights [8]. In short, it can be said that both mathematical and engineering fields have the importance of the study of the MWDS problem. For example, to lessen the number of full wavelength converters in the deployment of wavelength division multiplexing all-optical networks the MWDS problem is used to solve the sparse wavelength converters placement problem [9]. This problem is also indulged in ad-hoc network problems where the nodes have to be determined and intrusion detection software for squandering detection needs to be installed [10]. MWDS problem is also redesigned for web databases to find an optimal query selection plan [11]. However, many approximation algorithms were introduced for special graphs such as UDG problems [7]. However, for general graphs, it is not applicable, as the time complexity of the approximation algorithm is always less than O (ln*n*) until P = NP [12]. Thus the heuristic algorithm is widely evaluated for solving the MWDS problem to get high-quality solutions. Since the more data will be generated in real-world problems, the more vertices will be in the graph. To cope with these handsome amounts of data in a minimum time metaheuristic algorithm is inexorable.

There are various metaheuristic algorithms that are used in different combinatorial optimization-based problems. In recent times, the mixed-variable differential evolution approach [13], the self-learning discrete Jaya algorithm [14], the two-stage cooperative evolutionary algorithm [15], the cooperative water wave optimization algorithm [16], and the cooperative multistage hyper-heuristic algorithm [17] have helped solve many combinatorial optimization-based problems significantly. Using a self-adaptive differential evolution algorithm [18], the batch-processing machine problem has been solved. On the other hand, the self-learning discrete Jaya algorithm [14] has been used in solving the energy-efficient distributed no-idle flow-shop scheduling problem (FSP) in a heterogeneous factory system (HFS-EEDNIFSP). Besides, the no-wait flow-shop scheduling problem is introduced and solved by implementing a two-stage cooperative evolutionary algorithm [15] with problem-specific knowledge. The no-idle flow shop scheduling problem is also solved using a cooperative water wave optimization algorithm [16] that is hybridized with reinforcement learning. The problems solved using these optimization algorithms are mainly combinatorial-based optimization problems. These algorithms show their supremacy in solving these problems effectively.

As MDS and MWDS are both defined as NP-hard problems [19, 20], it is quite hard to find a near-optimal solution for large instances in reasonable CPU time [20]. Thus, for getting an optimal solution to the MWDS problem, methods using metaheuristic algorithms are very much effective [21]. Inspired by this knowledge, many researchers suggested metaheuristic algorithms to obtain a better solution. They proposed different approaches based on various metaheuristic algorithms including binary particle swarm optimization (BPSO) algorithm [22], ant colony optimization (ACO) algorithm [23], hybrid ACO-PP-LS algorithm [24], a randomized population-based iterated greedy approach R-PBIG [25], hybrid tabu search algorithm [21], hybrid memetic algorithm (HMA) [26], and hybrid genetic algorithm (HGA) [24]. However, the efficiencies of existing metaheuristic algorithms are still not up to the mark, especially for hard and large-scale instances [27].

Although there are many metaheuristic algorithms that were proposed to solve the MWDS problem, here we are proposing an approach based on the chemical reaction optimization (CRO) algorithm. This algorithm has an efficient and effective gain in solving NP-hard problems [28, 29]. The reason behind the dominance of this algorithm is its flexible architecture for both local and global searches, multiple operators, and the ability to work with variables in both discrete and continuous domains. Islam et al. [30] showed these diversified qualities and robustness of the CRO algorithm in their work. CRO was applied to the different optimization problems [28, 29, 31–45]. In the proposed approach, the population is generated using a ranking-based procedure along with the parameter-based random method. Next, the operators of CRO work on the population. Three new operators are applied as the supporting operators along with the CRO operators. To intensify the search, a new repair operator is used on the best solution obtained after CRO. After that, to the improvement of the obtained results, a common neighbor elimination procedure is applied. Contributions to this work are as follows:

(1) For solving the MWDS problem using CRO, the operators of CRO are redesigned to fit especially for this problem.

(2) A novel population generation procedure is introduced where the parameter-based and ranking-based searching procedures are merged. This combination improves the efficiency of searching in the search space.

(3) After population generation, molecule intensification procedure, a novel operator is introduced which can lead to obtaining a better solution.

(4) Along with the four operators of CRO, three supporting operators such as neighborhood search, local search, and redundant vertices elimination procedure, are designed and applied to make the solution feasible and to obtain better results. Among these operators, the concept of neighborhood search operator is unique which helps to find a minimum weighted neighbor.

(5) We have designed a novel repair operator where three properties of nodes are introduced which are uniquely identified to solve the MWDS problem. This operator can find the minimum weighted node with a maximum number of neighbors in minimum time.

(6) In our proposed method another state-of-the-art repair operator is introduced to eliminate common neighbors of the vertices of dominating set. This operator helps to prevent taking similar nodes into the dominating set which are the neighbors of another node of dominating set.

The rest of this paper is organized as follows: The recently developed approaches with their pros and cons are described in Section 2. Section 3 describes the chemical reaction optimization algorithm. The proposed technique for solving the MWDS problem using the chemical reaction optimization algorithm is explained in Section 4. Section 5 shows the results of the experiments with proper graphical and tabular representations. The conclusion of our work is delineated in Section 6.

1.1. Basic Concept. MWDS is a graph-based problem where it can be related to the set-covering problem [46]. According to the MWDS problem, it is a set of vertices of a large graph that are connected with every other vertex of the graph (i.e., covered all vertices of the graph) and the weight of the set of vertices is minimum.

Let $G = (V, E)$ be an undirected graph with $V = \{v_1, v_2, v_3, \ldots, v_n\}$ being the set of vertices and $E = \{e_1, e_2, e_3, \ldots, e_m\}$ being the set of edges. Each edge $e = \{u, v\}$ joins two vertices u and v, and these vertices represent the endpoints of e. In order to acquire the smallest dominating set, we must know the neighbor of any vertex and include those vertices with the greatest number of neighbors.

The following equations are used to obtain the minimum weight dominating set, where $N\ [i]$ denotes the closed neighbor of vertex i (i.e., $N[i] = N(i)\text{cup}i$), D denotes the dominating set and $D\text{in}V$, omega represents the weight of each vertex, and x is the decision variable, which is set to 1 if the vertex is included in the dominating set; otherwise, it is 0 [12]. For getting the minimum weight dominating set, equation (2) is introduced, and equations (3) and (4) have been used as constraints.

$$T_\omega = \sum_{v \in D} \omega(v) x(v), \tag{1}$$

$$D_{m\omega} = \min(T_\omega), \tag{2}$$

$$\sum_{k \in N[[i]} x(k) \geq 1, \quad \forall_i \in D, \tag{3}$$

$$x(v) \in \{0, 1\}. \tag{4}$$

Here, T_ω denotes the total weight of the members of the dominating set and $D_{m\omega}$ denotes the minimum weight dominating set.

Figure 1 shows an example of the MDS problem where the graphs are unweighted. Both Figures 1(a) and 1(b) depict the minimum dominating set. As in Figure 1(a), the vertex set $\{v_1, v_2\}$ is selected where $N[v_1] = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and $N[v_2] = \{v_1, v_2, v_3, v_4\}$ and in Figure 1(b) $\{v_1, v_4\}$ is selected in which $N[v_1] = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and $N[v_4] = \{v_1, v_2, v_3, v_4\}$. Though Figure 1(c) denotes a dominating set, it is not a minimum one as in this set three vertices are selected where the set is $\{v_4, v_5, v_6\}$.

Figure 2 shows an example of the MWDS problem where the weights of the graphs are introduced. In Figure 2(b) total weight, $T_{\omega 1} = (14 \times 1 + 10 \times 1 + 80 \times 0 + 60 \times 0 + 75 \times 0 + 45 \times 0 = 24)$. In Figure 2(a) the dominating set is $\{v1, v4\}$ and total weight, $T_{\omega 2} = (14 \times 1 + 10 \times 0 + 80 \times 0 + 60 \times 1 + 75 \times 0 + 45 \times 0 = 74)$. Now, $D_{\text{mow}} = \min\ _\omega(T_{\omega 1},\ T_{\omega 2}) = \min\ (24, 74) = 24$, so Figure 2(b) shows the minimum weight dominating set.

2. Literature Review

In recent years, MWDS has demonstrated its importance in addressing a wide range of social data and network-related issues. Many academics have been drawn to it because it is nearly impossible to work with massive amounts of data in a graph if each vertex should always be traversed one by one using an exact algorithm. As a result, researchers presented algorithms that can be divided into two groups to handle this problem; one of them is approximation algorithms, and another is metaheuristic algorithms [22]. According to Alon et al. [12] if $P \neq NP$ then set-cover is inapproximable polynomially within a factor c ln$n$, where c is a constant and $1 > r\text{bin}c > r\text{bin }0$. The MWDS problem is ambiguous with the set-cover problem in polynomial cases [46]. Therefore, it is also inapproximable for the case when $P \neq NP$.

However, several researchers focused on redesigning heuristic algorithms to solve general graphs, more precisely the MWDS problem, to acquire optimal solutions. In recent times, to overcome the MWDS problem, Lin et al. [26] presented a hybrid memetic method. Their first step was the formulation of the MWDS problem as a constrained 0-1 programming problem and then transfigured to an equivocal unconstrained 0-1 problem using an adaptive penalty function. A hybrid memetic algorithm (HMA) was used on this resulting unconstrained 0-1 problem to obtain optimal dominating sets. In this hybrid algorithm, they merged the greedy randomized adaptive construction procedure, a tabu local search procedure, a crossover operator, a population updating method, and a path relinking procedure. Their proposed algorithm had a comparison with RAKA-ACO [23], HGA, ACO-LS, and ACO-PP-LS [24], where the computation time was almost six times faster in HMA. However, for different instances, the average objective function values were not optimal in this algorithm.

Another algorithm used by researchers to deal with the MWDS problem is CC^2FS. Wang et al. [27] proposed a new form of the configuration checking heuristic algorithm called two-level configuration checking (CC^2), which they paired with a novel scoring function based on the frequency of vertices being uncovered. This hybrid algorithm is named as CC^2FS. According to their intended algorithm, the experimental result was quite mesmerizing. In every instance, this algorithm outperformed any other algorithms. However, another newly generated optimization algorithm, as well as some other versions of configuration checking such as strong configuration checking, was not tested on the MWDS problem according to them. On the other hand, they were not sure enough to get good results according to their experiment in larger graphs.

Following the sequence of solving the MWDS problem using a variant of metaheuristic algorithms, in 2018, two methods were introduced. One was a Binary Particle Swarm Optimization (BPSO). In this proposed methodology, Lin and Guan [22] redesigned BPSO for the MWDS problem by combining Tabu search, where tabu search was introduced to enhance the solution quality rapidly. They also used stochastic strategies to prevent premature convergence and diversify the search. According to their experimental result, the average CPU time needed for every instance was far less than previously proposed algorithms, though they did not show their solution as the benchmark as in some cases this algorithm cannot find the best solution. Another algorithmic model for solving the

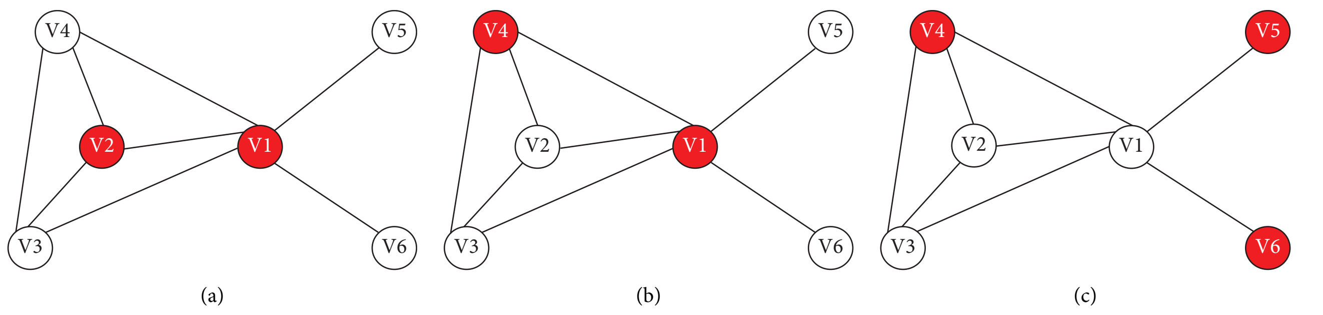


FIGURE 1: An example of a minimum dominating set.

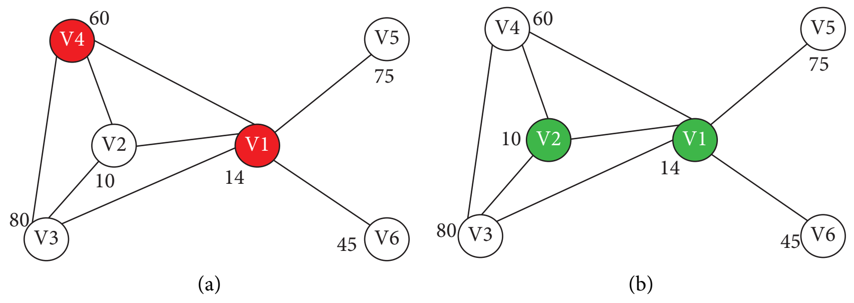


FIGURE 2: An example of a minimum weight dominating set.

MWDS problem initiated in 2018 was an efficient metaheuristic algorithm. In this model, Albuquerque and Vidal [21] suggested a metaheuristic model, whereas metaheuristic refers to algorithms in which metaheuristic algorithms are linked with mixed-integer programming (MIP) strategies and software [47, 48]. According to their suggested architecture, four strategies were combined to reproduce a novel solution. Tabu search had been initiated for searching the efficient neighborhood, and to promote exploration perturbation had been introduced. The method is introduced as a hybrid tabu search metaheuristic (HTS-DS). In their architecture, they used a node elimination procedure which elapsed more CPU time, though according to their experimental result, the CPU time was much less than other proposed systems as well as their suggested methodology outperforms any other previous architecture.

At the beginning of 2019, Yuan et al. [49] came up with a fresh architecture using a hybrid algorithm to solve the minimum total dominating set (MTDS) problem. In this proposed architecture, a local search and genetic algorithm have been merged. Score based heuristic algorithm was generated to enhance the search effectiveness and then a repair-based crossover operation was implemented to get viable solutions. Though this algorithm is better for some instances, it cannot help to get a good result for every instance.

In 2020, Shetgaonkar and Singh [50] introduced hybridized method using an artificial bee colony algorithm with the estimation of distribution algorithm (ABC-EDA). In this method, they hybridized the algorithms by merging them and used guided mutation as a repair operator. Albeit of the minimum run time, they could not be able to obtain the minimum weight of the dominating set.

3. Chemical Reaction Optimization

The inspiration for the chemical reaction optimization algorithm comes from the behavior of chemical reactions. It is a population-based metaheuristic optimization algorithm. The property of a chemical reaction is to transform any unstable molecule into a stable one by a natural process. This process is similar to a step-wise search for optimization problems [51].

Like other metaheuristics, the CRO algorithm consists of three stages. They are the initialization, iteration, and termination stages [32]. The initialization stage promotes the initial population besides the parameters such as population size (PopSize), kinetic energy loss rate (KElossRate), MoleColl, Buffer, initial kinetic energy (InitialKE), and two thresholds (α and β).

The initialization stage begins with the initialization of the different parameters [51]. In the iteration step, one elementary reaction from four runs in each iteration. A random number b between 0 and 1 determines whether the reaction operator is unimolecular or not. If $b >$ rbin MoleColl, there is a unimolecular reaction otherwise, there is a bimolecular reaction. At the end of each iteration, we have to identify the conclusion principle. The last step controls when the conclusion principle of the iteration step is fulfilled. In the final stage, the algorithm aborts, and the premier solution is the outcome.

The unimolecular collision has two types of reactions which are on-wall ineffective collision and decomposition. Similarly, the intermolecular collision has two reaction operators, which are intermolecular ineffective collision and synthesis.

The molecules in CRO have several parameters, some of which are optional to the basic operations. Table 1 depicts the different parameters of CRO.

4. MWDS Using Chemical Reaction Optimization

The chemical reaction optimization algorithm (CRO) is used in this study to get a minimum weighted dominating set. Initial values for different CRO parameters are introduced at the preliminary stage, and populations are produced. Following initialization, one of the four operators is used to begin the iteration process to acquire a better solution from solution space. To attain a better solution, we also used repair operators besides the operators of CRO. The iteration process runs until the termination criteria are met. After termination, the repair operator and integer linear programming (ILP) are used to obtain the further minimum weighted solution. The workflow of the proposed architecture to find the minimum weight dominating set is as follows:

(1) Initially from the data file, the graph structure has evolved and then the initial population has been generated
(2) Molecule intensification procedure has been applied to every molecule of the population to obtain the initial minimum weighted dominating set
(3) Based on the comparison of a random number between 0 and 1 and the parameter value of molecule collision rate (MoleColl), one or more molecule(s) has/have been selected
(4) Based on the condition of decomposition (for a single molecule) or synthesis (for more than one molecule) operator has been chosen and new molecule(s) are found
(5) Local search procedure is applied to the newly found molecule(s) to cover up all the vertices
(6) If the fitness value of generated molecule is better than the original molecule then the original one is replaced with the up-to-date molecule in the population set
(7) Otherwise, the original molecule remains in the population set
(8) Steps 3 to 7 to be repeated till it is unworthy for termination condition
(9) Repair operator is applied on the molecules found from early steps
(10) Common neighbor elimination procedure is applied to eliminate common neighbors of vertices that are the members of dominating set

4.1. Generation of Population. The technique of obtaining a feasible solution to the MWDS problem from search space can be divided into three approaches. These approaches are penalty-based procedure [21], preserving-based approach [24], and repair-based approach [49]. From these three approaches to obtain the feasible population, repair based approach is initiated in our proposed method. The technique of generating a molecule is divided into three distinct stages. The first stage of this technique is to obtain a feasible solution based on the cost of the node and the second one is to obtain a feasible solution based on the weight of the node and the degree of the node. Finally, the solutions of both of these stages are merged. This merging process occurs based on the similarity between taken nodes. If the nodes are taken in both solutions then the nodes are kept as the member of the dominating set.

The representation of a solution (molecule) in our proposed architecture is based on 0 and 1. If any vertex is taken as the member of dominating set, then in the molecule, it is represented by the value 1 and reported as an active node. Otherwise, the value with 0 is represented as an inactive node. Figure 3 shows the molecule representation.

The final feasible molecule according to the representation in Figure 3 is generated after processing three stages of molecule generation which are the initial stage, middle stage, and final stage. The population set is generated by the iteration of this process twenty times as the taken population size of this work is twenty.

4.1.1. Initial Stage. Primarily, the vertices with minimum weight are considered by comparing cost with a specific value with the help of randomness. If the cost of any vertex is less than the difference between the average cost (μ) and the product of epsilon (ϵ) and the standard deviation (σ) of the cost then the vertex is selected randomly. Here, epsilon (ϵ) is a new parameter that is introduced in the initial stage of the population generation method to obtain better random molecules for further operation. The cost is calculated as follows:

$$c_i = \frac{\omega_i}{d_i}. \tag{5}$$

Here, c_i, ω_i, and d_i refer to the cost, weight, and degree of vertex *i,* respectively.

After this procedure, to make the solution feasible by covering all vertices and reducing search space, the properties of vertices are checked. The vertex of minimum weight and maximum uncovered neighbors properties is the highly prioritized vertex to count as the member of dominating set. The priority of the vertex depends on the probability score, which is calculated as follows:

$$P_i = \frac{PC_i}{\sum_{j\in N[i]} PC_j} \tag{6}$$

Here, P_i refers to the probability score and PC_i denotes the probable cost of vertex *i*. PC_j is the probable cost of neighboring vertices of vertex *i*. The probability cost of vertex *i* is determined as follows:

$$PC_i = \frac{d_i \times \sum_{j\in N[i]} \omega_j}{\omega_i}. \tag{7}$$

After the initial process to cover up the uncovered nodes, priority-based nodes are selected randomly from the first 30% of highly prioritized nodes if the ratio of the number of vertices and the number of edges is greater than the constant value of 0.3.

Table 1: Parameters of CRO.

No.	Symbol	Algorithmic definition
1	PopSize	The number of molecules in a population or simply the population size
2	KELossRate	The loss rate of kinetic energy (KE) during the reaction
3	MoleColl	A parameter varying between 0 and 1 decides whether the chemical reaction is unimolecular or intermolecular
4	Buffer	The initial energy in the surroundings
5	InitialKE	The initial energy
6	α and β	Two parameters controlling the intensification and diversification
7	Number of hits (NumHit)	The total number of hits a molecule has taken
8	Minimum structure (MinStruct)	The molecule structure that has minimum potential energy
9	MinimumPE (MinPE)	When a molecule attains its MinStruct, MinPE is its corresponding PE
10	Minimum hit number (MinHit)	It is the number of hits when a molecule has MinStruct. It is an abstract notation of the time when MinStruct is achieved

0	0	1	0	1	0	1	1	1	0	1	1	0	0	0	0

Figure 3: Example of a molecule.

Otherwise, the nodes are selected based on the percentage of the ratio. For example, if there is a sample graph with 500 vertices and 20000 edges, then the ratio is 500/20000 = 0.025. According to the delineation of the process, the active nodes are chosen from the first 2.5% of highly prioritized nodes. After the selection process, the highly dominating nodes are considered active nodes. Despite selecting highly dominating nodes, it cannot be able to cover all the nodes. Again, to find a feasible solution, the nodes that are not covered are picked as spared nodes and put in a list. From this list, active nodes are selected randomly until all nodes are covered. Algorithm 1 shows the initial stage of the population generation procedure.

4.1.2. Middle Stage. In this stage, to minimize the computational time for large graphs we check for the node with zero, one, and two degrees. The node with zero degree means the node is not connected with any other nodes and automatically will be the member of dominating set. The node with degree one means the node is connected with only one neighbor. Hence from these two nodes which one is minimum weighted is taken as the member of dominating set. The node with two degrees is referred to as the triangle node. These types of the node have two neighbors. From these three nodes, one node is selected as the member of dominating set based on weight. The minimum weighted node is taken as the dominating member. To take the node into dominating set the property of neighbors are also checked to determine whether the neighbor nodes are the member of dominating set or not. Let, q, r and p are three nodes and $\{q, r\} \in N(p)$ and p is the node with minimum weight, so, if D is the dominating set, D' is the dominated set and $\{p\} \notin D'$ then, $\{p\} \in D$. These nodes are taken in a molecule as a permanent node.

After accomplishing this procedure there are some nodes to be covered that still remain uncovered. From the list of the uncovered nodes, the size of common nodes is checked which technique helps to find the nodes that can cover the maximum number of nodes. If for all of the uncovered nodes the total number of uncovered neighbor nodes is one then the node with minimum weight is taken as the member of dominating set. After this process, if any of the nodes remains uncovered then the neighbor node with a minimum weight of the uncovered node is taken as the member of the dominating set. Algorithm 2 shows the initial stage of the population generation procedure.

4.1.3. Final Stage. In this stage, two different techniques of generating molecules are merged together to obtain feasible molecules for searching using CRO. The nodes which are common in both molecules are considered active nodes. This procedure can make the molecule infeasible. To make this molecule feasible, a local search procedure has been applied and finally, the molecule is prepared for deployment in the searching paradigm. The whole process from the initial stage to the final stage works twenty times to generate twenty different molecules as members of the population.

4.2. Molecule Intensification Procedure. After generating molecules, to intensify searching, a molecule intensification procedure is introduced. Every molecule has gone through this procedure to find the optimal or best solution so far. This procedure is also a ranking-based system of the nodes. After the production of the molecules, some nodes are the neighbors of the neighbors, of active nodes. Initially, they are not considered active nodes deliberately. However, there is some chance to get an optimal solution or intensified searching space for taking those nodes. For the selection of the nodes for which the molecules will be optimal, the molecules have been made infeasible intentionally by removing active nodes, which are low-ranked based on equation (7). The size of the neighbors of each node is counted based on the list of low-ranked nodes. Then for each node, the cost is calculated as follows:

$$c_i = \frac{L(NC(N(i)))}{\omega_i}. \tag{8}$$

Here, c_i refers to cost, and $L(NC(N(i)))$ refers to the length of the uncovered neighbors of the vertex i from the

```
    Input: properties of graphs.
    Output: feasible solution.
(1) M ⟵ ∅
(2) for $i \longleftarrow 0$ to $len(V)$ do
(3)     if $c[i] < \mu - \epsilon \times \sigma$ and random value $\geq 0.5$ then
(4)         M[i] ⟵ 1
(5)     end
(6)     else
(7)         M[i] ⟵ 0
(8)     end
(9) end
(10) Calculate the probability score using equation (6)
(11) Select high prioritized node based on the probability score
(12) Create a list of spared nodes
(13) Select nodes from spared nodes until all nodes are covered
```

Algorithm 1: The initial stage of population generation.

```
    Input: properties of graphs.
    Output: feasible solution.
(1) M ⟵ ∅
(2) Add nodes with zero degree, one degree, and two degrees as active nodes into M.
(3) for $uc \neq \varnothing$ do//$uc$ refers to the list of uncovered nodes
(4)     for $v_i \in uc$ do
(5)         $cn \longleftarrow N(v_i) \cap uc$; //$cn$ refers to the list of common nodes between the neighbor node of $v_i$ and the list of uncovered nodes
(6)         if $cn \neq \varnothing$ then
(7)             $pn \longleftarrow v_i$; //$pn$ is the probable member of dominating set
(8)             $cn_l \longleftarrow cn$; //$cn$ is the list of the common node
(9)             $Len \longleftarrow len(cn)$
(10)        end
(11)        else
(12)            Continue
(13)        end
(14)    end
(15)    if $Len \neq \varnothing$ then
(16)        $ML \longleftarrow \max(\text{Len})$; //$ML$ is the maximum size from all the common nodes' size
(17)        if $ML > 1$ then
(18)            $M[v_m] \longleftarrow 1$; //$v_m$ is the node that can cover the maximum number of uncovered nodes
(19)        end
(20)        else
(21)            if $\omega_i > \omega_{N[i]}$ then
(22)                $M[v_i] \longleftarrow 1$
(23)            end
(24)            else
(25)                $M[v_{N[i]}] \longleftarrow 1$
(26)            end
(27)        end
(28)    end
(29)    else
(30)        Find uncovered nodes.
(31)        Find the minimum weighted node ($v_{\min}$) associated with uncovered nodes.
(32)        $M[v_{\min}] \longleftarrow 1$
(33)    end
(34) end
```

Algorithm 2: The second stage of population generation.

```
     Input: one molecule, $M$
     Output: updated molecule, $M'$
(1)  Count total active nodes $(T_n)$
(2)  total search, $S_n \longleftarrow 0$
(3)  while $True$ do
(4)      An active node $(V_i)$ has been chosen randomly from the molecule
(5)      $\omega_{\min} \longleftarrow \omega_i$
(6)      $V_{\text{prev}} \longleftarrow V_i$ for $j \in N[i]$ do
(7)          if $\omega_j < \omega_{\min}$ then
(8)              $\omega_{\min} \longleftarrow \omega_j V_{\text{taken}} \longleftarrow V_j$
(9)          end
(10)     end
(11)     $S_n \longleftarrow S_n + 1$
(12)     if $V_{\text{prev}} \neq V_{\text{taken}}$ then
(13)         $M'[\text{taken}] \longleftarrow 1$
(14)         $M'[\text{prev}] \longleftarrow 0$
(15)         break
(16)     end
(17)     if $T_n = S_n$ then
(18)         break
(19)     end
(20)     else
(21)         continue
(22)     end
(23) end
```

Algorithm 3: Neighborhood search.

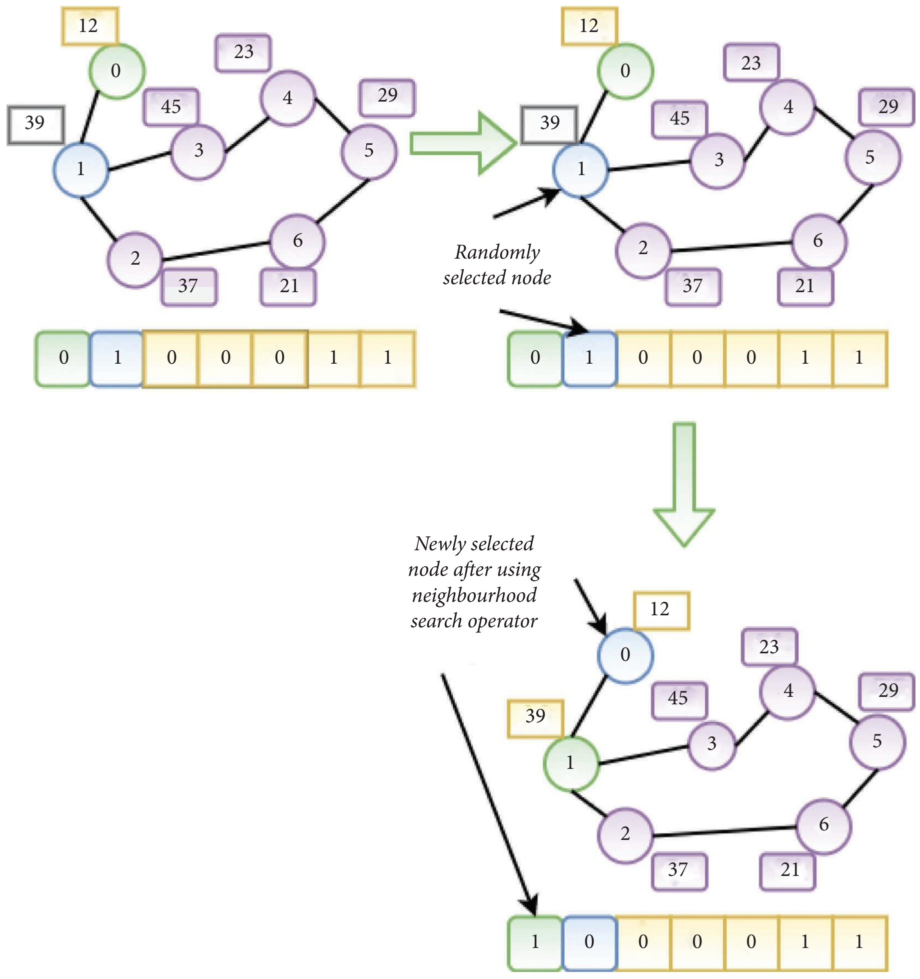


Figure 4: Illustration of a neighborhood search operator.

```
     Input: infeasible molecule, $M$
     Output: feasible molecule, $M'$
(1)  while $NC \neq \emptyset$ do
(2)    Calculate the greedy value for $v_i \in NC$
(3)    $g_{\text{all}} \longleftarrow$ greedy values of $v_i \in NC$
(4)    $g_{\max} \longleftarrow \max(g_i)$
(5)    for $gv \in g_{\text{all}}$ do
(6)      if $gv \geq \delta \times g_{\max}$ then
(7)        $M'[gv] \longleftarrow 1$
(8)        break
(9)      end
(10)     else
(11)       continue
(12)     end
(13)   end
(14)   Check for uncovered vertices
(15)   update $NC$
(16) end
```

Algorithm 4: Local search.

```
     Input: molecule, $M$
     Output: updated molecule, $M'$
(1)  while $R \neq \emptyset$ do
(2)    $\omega_{\max} \longleftarrow 0$
(3)    $v_{\text{eliminate}} \longleftarrow 0$
(4)    for $rv \in R$ do
(5)      if $\omega_{\max} < \omega_{rv}$ then
(6)        $\omega_{\max} \longleftarrow \omega_{rv}$
(7)        $v_{\text{eliminate}} \longleftarrow rv$
(8)      end
(9)      else
(10)       continue
(11)     end
(12)   end
(13)   $M' \longleftarrow 0$ Check for redundant vertices
(14)   update $R$
(15) end
```

Algorithm 5: Redundant vertices elimination.

low-ranked list. From these vertices, the cost with the maximum value is selected as active nodes. This process is iterated till all the vertices are covered.

4.3. Operator Design. In the proposed method, four operators of CRO and three supportive operators have been designed to solve the MWDS problem. The delineation of operators is as in the following subsections.

4.3.1. Neighborhood Search Operator. A small amount of change occurs due to the reaction of both on-wall ineffective collision and intermolecular ineffective collision, which is similar to the local search of any other evolutionary algorithms. The inspiration for the neighborhood search operator comes from this concept. In this operator, the neighbor node with minimum weight has been searched from any active nodes. An active node is picked randomly, and the searching occurs among all neighbor nodes inclusive of the active node for the lowest weight. If the active node is not the lowest weighted, then the lowest-weighted neighbor is activated, the active node becomes deactivated, and the process terminates. Otherwise, the same procedure occurs to find another active node randomly. This process runs until obtaining a molecule different from the given one. The final termination occurs due to the unavailability of any new node after checking all active nodes. Algorithm 3 shows the neighborhood search procedure.

Figure 4 depicts the search procedure of the neighborhood search operator.

4.3.2. Local Search Procedure. After using neighborhood search on on-wall ineffective collision, intermolecular ineffective collision, decomposition, or synthesis, some vertices of

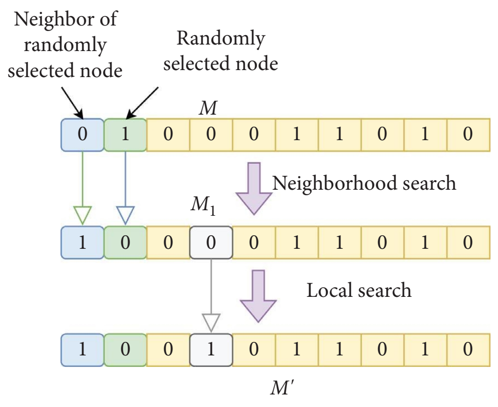


Figure 5: Illustration of on-wall ineffective collision.

```
     Input: one molecule, $M$
     Output: updated molecule, $M'$
(1)  Copy $M$ to $M'$
(2)  Randomly select a position $i$ from $M'$
(3)  if $M'[i] = 1$ then
(4)      Run neighborhood search operator
(5)  end
(6)  else
(7)      $M'[i] \longleftarrow 1$
(8)  end
(9)  Run the local search.
(10) Run redundant vertices elimination.
```

Algorithm 6: On-wall ineffective collision.

a molecule can be uncovered, and the solution becomes infeasible. To make the solution feasible by covering all the uncovered vertices, a technique suggested by Potluri and Singh [24] has been used in our method. They proposed four greedy approaches for searching a node from the set of vertices. From these four approaches, the assimilation of the fourth approach occurs in our system due to its better performance than the other three approaches. The greedy approach follows the following equation to calculate the value of a vertex (v_i):

$$g_i = d_i \times \frac{\sum_{j\in N(i)} \omega_j}{\omega_i}. \quad (9)$$

Here, $\sum_{j\in N(i)} \omega_j$ denotes the summation of all uncovered neighbors of a single vertex (v_i), ω_i represents the weight, and d_i is the degree of v_i.

The calculation of greedy values of every uncovered vertex takes place by using equation (9). Then a vertex is activated if $g_i \geq \delta \times \max(g_i); \forall\{v_i \in NC\}$. Here, NC represents the set of uncovered vertices, and δ is a controlling parameter for obtaining a better vertex. The procedure runs until all vertices are covered. Algorithm 4 shows the local search procedure.

4.3.3. Redundant Vertices Elimination. If any vertex (v_i) is a member of dominating set and all of its neighbors inclusive of the vertex are the neighbors of one or more members of dominating set, then the vertex is defined as the redundant vertex. Removal of redundant vertices can minimize the total weight of the dominating set. Thus the redundant vertices with maximum weights have been removed from the molecule to obtain the minimum weighted solution. If R is a set of redundant vertices where $v_i \in R$ and ω_i is maximum, then v_i is removed from the molecule, and then further redundant vertices are checked and removed. This process runs until the removal of all the redundant vertices. The procedure for eliminating redundant vertices has been shown in Algorithm 5.

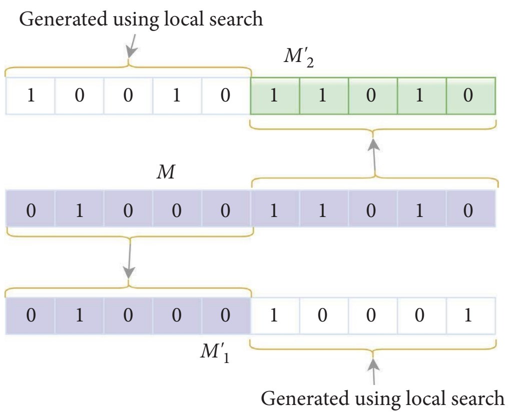


Figure 6: Illustration of decomposition.

```
    Input: one molecule, $M$
    Output: updated molecules, $M_1'$ and $M_2'$
(1) for $i \longleftarrow 0$ to $len\ (V)/2$ do
(2)     $M_1'[i] \longleftarrow M[i]$
(3) end
(4) Run a local search on $M_1'$
(5) for $i \longleftarrow len\ (V)\ /2$ to $len\ (V)$ do
(6)     $M_2'[i] \longleftarrow M[i]$
(7) end
(8) Run the local search on $M_2'$
```

Algorithm 7: Decomposition.

4.3.4. On-Wall Ineffective Collision. It is a unimolecular reaction. So in this operator, initially, a molecule has been selected randomly. Then the molecule is modified to generate a new optimal molecule. Primarily, a random selection of a position from the molecule takes place for propagating the latest molecule. If the node of the randomly selected position is active, then the position changes according to the neighborhood search operator. Otherwise, the inactive node becomes active. Finally, for any uncovered nodes, the nodes are covered using the local search procedure. Likewise, for any redundant nodes, the redundant vertices elimination procedure takes place to eradicate redundancy. Figure 5 depicts the MWDS problem's on-wall ineffective collision, where M denotes the molecule (i.e., solution) and M' denotes the newly created molecule after applying both neighborhood and local search. The procedure of on-wall ineffective collision is shown in Algorithm 6.

4.3.5. Decomposition. It is also a unimolecular reaction where the molecule can be divided into two or more. To find a better solution for the MWDS problem, the search happens in the search space according to this reaction by dividing the parent molecule into two. Both two new molecules inherit a portion from the parent molecule. The parent molecule is divided from the middle. Then the first portion of the parent molecule becomes the first portion of the first child molecule, and the other portion of the parent molecule becomes the last portion of another child molecule. Then for both molecules, uncovered vertices are covered using a local search procedure. Figure 6 shows the decomposition procedure. Here, in the first child molecule (M_1'), the first portion is similar to the first portion of the parent molecule, and in the second child molecule (M_2'), the last portion is the same as the last portion of the parent molecule. Algorithm 7 depicts the working procedure of decomposition.

4.3.6. Intermolecular Ineffective Collision. It is a multimolecular reaction operator where two or more parent molecules can react with each other and generate new molecule(s) where the change in the new molecule is little. This operator searches locally in search space such as the on-wall ineffective collision. In the proposed method, two molecules are used to generate new molecules by implementing this operator. A single position (i.e., vertex) from both parent molecules is considered randomly. Then the values from both positions are interchanged in child molecules. Then uncovered nodes are covered, and redundant vertices are removed from both child molecules. Figure 7 depicts the intermolecular ineffective collision where M_1 and M_2 are the parent molecules and M_1' and M_2' are denoted as child molecules. Algorithm 8 shows the working procedure of intermolecular ineffective collision.

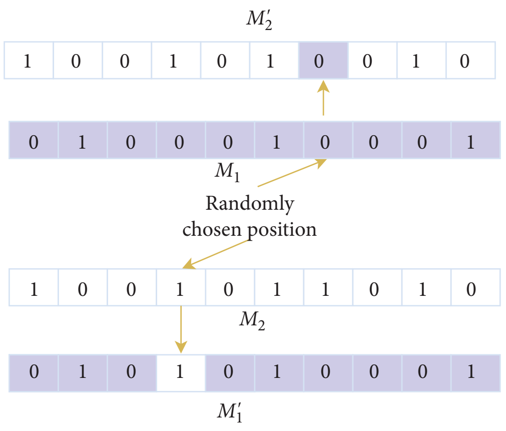


Figure 7: Illustration of intermolecular ineffective collision.

4.3.7. Synthesis. It is also a multimolecular reaction operator. In this operator, the amalgamation of two molecules generates a child molecule to obtain a better offspring. Let, there are two molecules M_1 and M_2 in the population set. If any vertex in both parent molecules is active, then the vertex remains active in the child molecule (M'). Other vertices of the child molecule become inactive. Then local search procedure is applied to the child molecule to obtain a feasible solution. Figure 8 shows the illustration of the synthesis reaction, and Algorithm 9 depicts the working procedure of synthesis.

4.4. Operator Selection. Some comparisons occur to select one of the four operators of CRO for the MWDS problem. The intermolecular collision occurred based on the comparison between a random value and MoleColl. If the random value between 0 and 1 is less than MoleColl, then the collision is unimolecular. Otherwise, the reaction is

```
Input: two molecules, $M_1$ and $M_2$
Output: updated molecules, $M_1'$ and $M_2'$
(1) Copy $M_1$ to $M_1'$
(2) Copy $M_2$ to $M_2'$
(3) Select a vertex $v_i$ from $M_1$ randomly
(4) Select a vertex $v_j$ from $M_2$ randomly
(5) $M_1'[i] \longleftarrow M_2[j]$
(6) $M_2'(j) \longleftarrow M_1[i]$
(7) Run a local search on $M_1'$
(8) Run redundant vertices elimination on $M_1'$
(9) Run the local search on $M_2'$
(10) Run redundant vertices elimination on $M_2'$
```

Algorithm 8: Intermolecular ineffective collision.

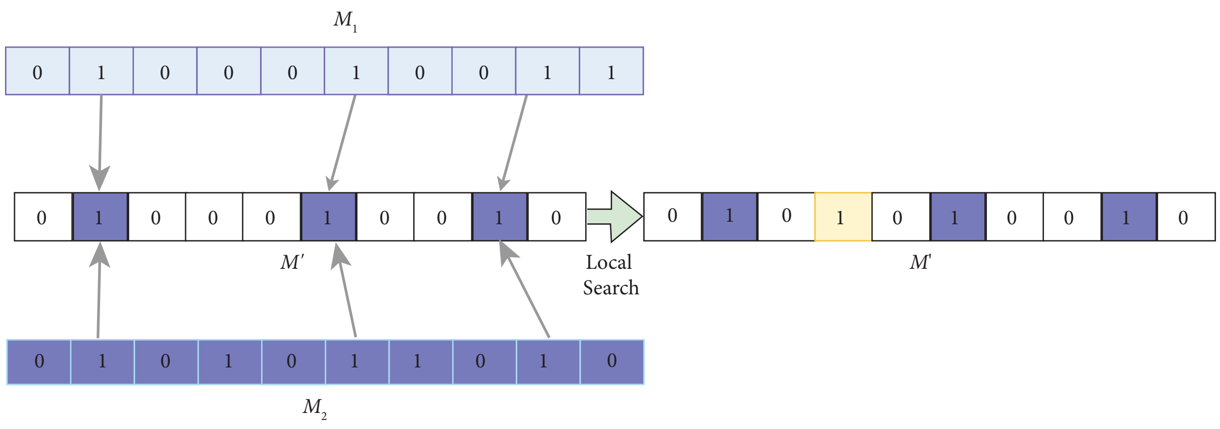


Figure 8: Illustration of synthesis.

intermolecular. If it is unimolecular, then the collision can be an on-wall ineffective collision or decomposition. The decision of on-wall or decomposition reaction comes from the following equation.

$$\text{hit\_diff} > \alpha. \quad (10)$$

Here, hit_diff refers hit difference of the selected molecule. If equation (10) is satisfied, decomposition occurs. Otherwise, we get an on-wall ineffective collision. In the same manner, synthesis or intermolecular ineffective reaction can occur if the random value is greater than the value of MoleColl. The decision of synthesis or intermolecular ineffective reaction is taken based on the following equation:

$$KE \leq \beta. \quad (11)$$

Here, KE is the kinetic energy of the molecule. If the kinetic energy of all molecules is less than or equal to β, then synthesis occurs. Otherwise, the reaction is an intermolecular ineffective collision. To get rid of the local minima synthesis is triggered based on the parameter values.

By following this approach, better offspring is found in which the number of vertices is minimum. Then using the repair operator and ILP, the offspring is checked for a better solution. If the offspring is better than the previous one then the candidate solution is updated, and the vertices of the candidate solution are taken as the member of the minimum weighted dominating set.

The reaction operators of CRO for the MWDS problem have been triggered based on the parameter values of Table 2, and an optimal solution has been obtained. The proposed flowchart of the CRO algorithm of the MWDS problem is shown in Figure 9.

4.5. Repair Operator. After using CRO operators with other supporting operators, there are some operations to obtain a better solution from the search space. As in the local search procedure, minimum weighted nodes are bound to inherit the solution; the search space is confined and cannot search for a further better solution. Alongside this drawback, there is another pitfall. In previous searching procedures, the properties of nodes are not updated after including the best node. Thus nodes are included every time by using old properties. For example, let a vertex (v_1) is taken as the best active node in the solution and the neighbors of v_1 are v_6, v_8, and v_9. If v_8 be the second-best node with the highest greedy value then this node is taken as the active node in the solution. But as v_8 is already dominated by v_1, by eliminating this, a better solution can be obtained.

To get rid of the aforementioned problems, a new repair operator has been proposed in our method with some other

```
Input: two molecules, $M_1$ and $M_2$
Output: updated molecule, $M'$
(1) for $i \longleftarrow 0$ to $len\ (V)$ do
(2)     if $M_1[i] = 1$ and $M_2[i] = 1$ then
(3)         $M'[i] \longleftarrow 1$
(4)     end
(5)     else
(6)         $M'[i] \longleftarrow 0$
(7)     end
(8) end
(9) Run the local search on $M'$
```

Algorithm 9: Synthesis.

Table 2: Initialization of parameters for proposed MWDS_CRO algorithm.

Parameters	Initialized values
Epsilon (ϵ)	0.9
Delta (δ)	1.45
Total iteration	100
Initial population size	20
KELossRate	0.1
MoleColl	0.1
Energy buffer	0
Hit_diff	0
Alpha (α)	10000
Beta (β)	1000
Kinetic energy (KE)	100

properties of nodes. As the feasible solution has been obtained from the CRO operators to apply repair operator on the solution, the solution is converted into an infeasible solution by uncovering some vertices. Forty percent (40%) of vertices with maximum weights from the dominating set are removed intentionally to make the infeasible solution. Then to find the better solution vertices are searched based on three properties. The properties of a vertex (v_i) are the weight of the vertex, uncovered neighbors of the vertex, and the ratio of weight and uncovered neighbor nodes. Then these properties are scaled based on minimum or maximum values of corresponding properties according to the following equations: (12)–(14).

$$S_{\omega_i} = \frac{\omega_{\min}}{\omega_i}. \tag{12}$$

$$S_{NC(N(i))} = \frac{NC(N(i))}{NC(N(\max))}, \tag{13}$$

$$S_{r_i} = \frac{r_{\min}}{r_i}. \tag{14}$$

Here, ω_i refers to the weight of the uncovered node v_i and $\omega_{\min}$ is the node with minimum weight from all uncovered nodes. S represents the scaled property, $NC(N(i))$ represents the number of total uncovered neighbors of the node v_i, and $NC(N(\max))$ represents the maximum number of all uncovered neighbors. The ratio (r_i) of the node v_i is calculated as follows:

$$r_i = \frac{\omega_i}{NC(N(i))}. \tag{15}$$

All these three properties of a vertex are considered by combining these properties. To make the solution feasible based on these properties, if any vertex has two or more uncovered neighbors, the vertex with the maximum aggregated value of the properties is selected as an active vertex. Otherwise, the vertex is selected as active based on weight. If the weight is minimum then the vertex becomes active. This procedure terminates when all the vertices are covered. The operator can find the minimum weighted node with the maximum number of neighbor nodes which helps obtain a better solution in minimal time.

Figure 10 shows the procedure of the repair operator. Let after using all of the previous operators on the graph of Figure 10 the selected nodes are 1, 3, and 6 and the dominating set is $\{1, 3, 6\}$. The weights of these three nodes are 6, 12, and 5 and the total weight is 23. As the nodes with maximum weights are removed, nodes 1 and 3 are removed from the dominating set and 6 remains in the dominating set. Now, three properties of uncovered nodes are checked. After removing nodes 1 and 3, nodes 1, 2, 3, and 5 become uncovered. The weights of these four nodes are 6, 10, 12, and 5, and the total number of uncovered neighbors of these four nodes are 1, 2, 2, and 1. Here, from the four uncovered nodes, the minimum weight is 5, and the maximum number of uncovered neighbors is 2. Now according to equation (12), the values of nodes 1, 2, 3, and 5 are 5/6 = 0.83, 5/10 = 0.5,

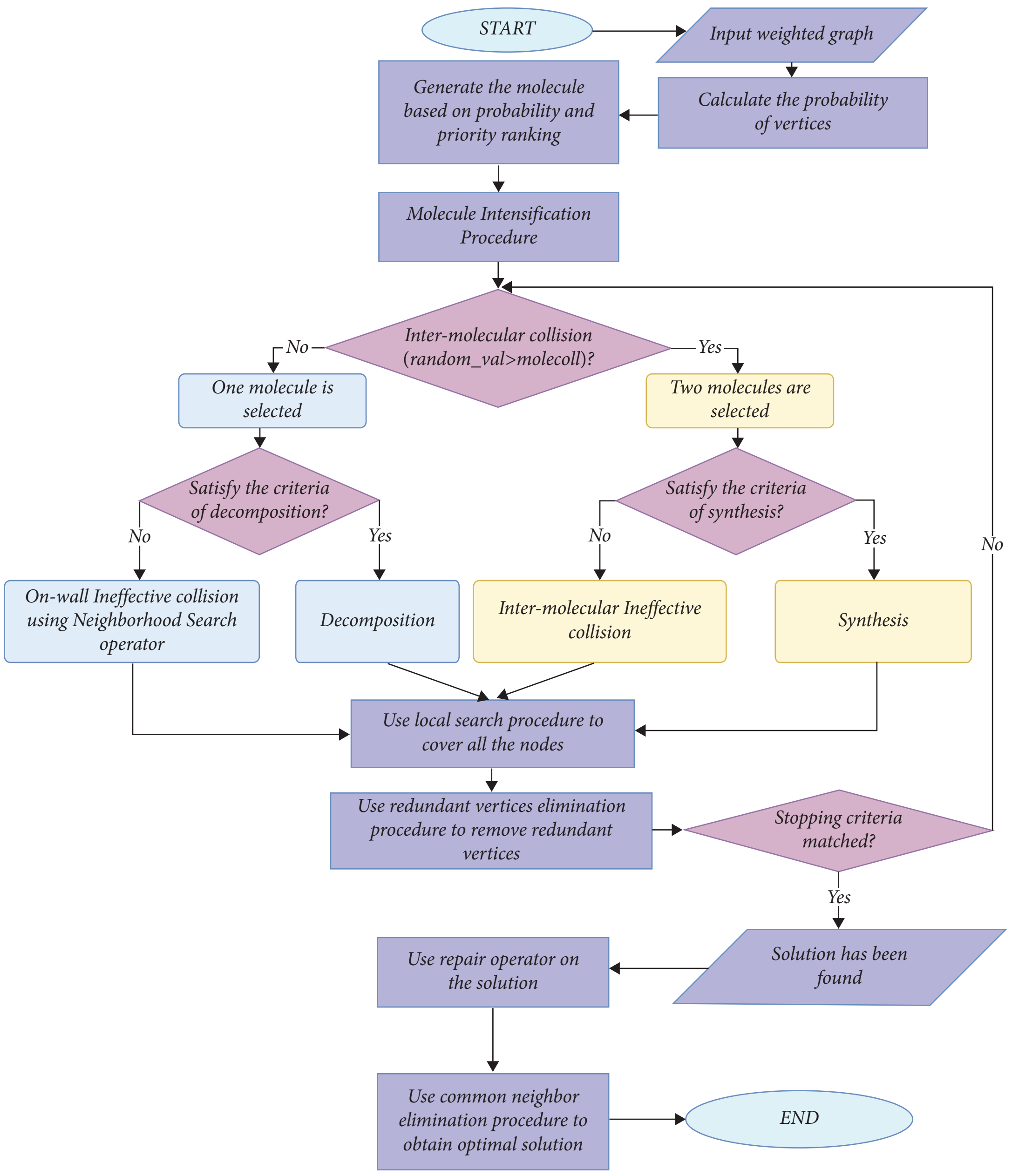
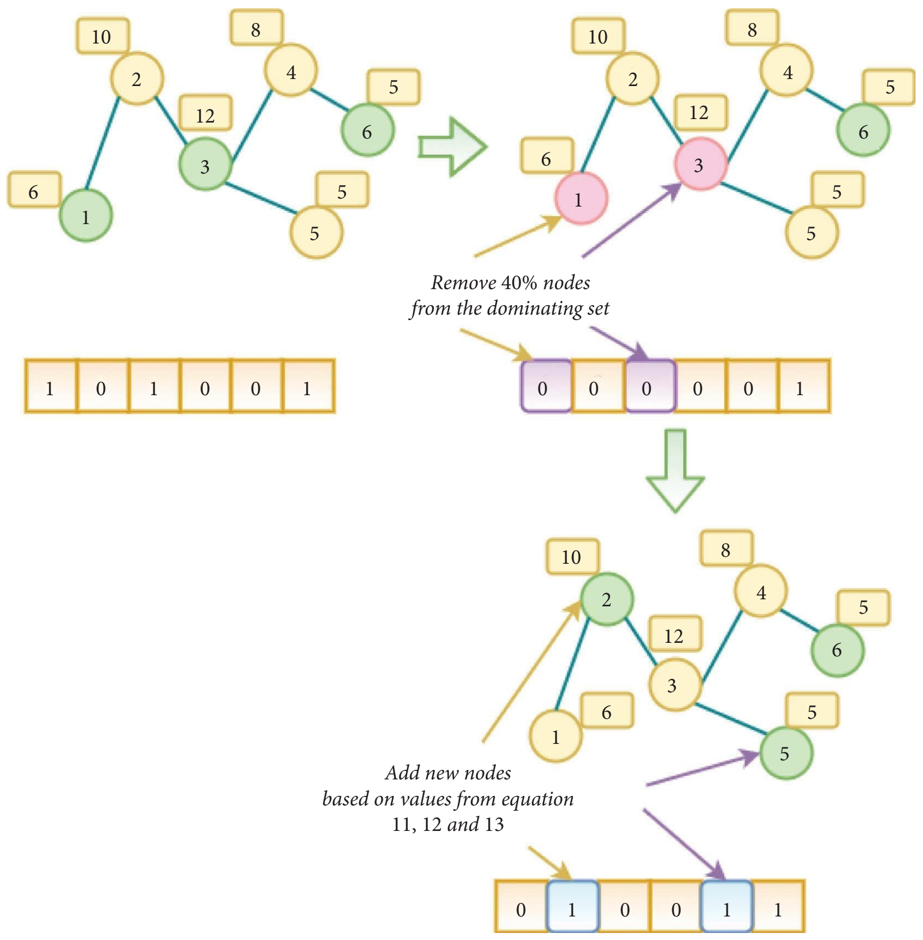

START
Input weighted graph
Calculate the probability of vertices
Generate the molecule based on probability and priority ranking
Molecule Intensification Procedure
Inter-molecular collision (random_val>molecoll)?
No
Yes
One molecule is selected
Two molecules are selected
Satisfy the criteria of decomposition?
Satisfy the criteria of synthesis?
No
Yes
No
Yes
No
On-wall Ineffective collision using Neighborhood Search operator
Decomposition
Inter-molecular Ineffective collision
Synthesis
Use local search procedure to cover all the nodes
Use redundant vertices elimination procedure to remove redundant vertices
Stopping criteria matched?
Yes
Solution has been found
Use repair operator on the solution
Use common neighbor elimination procedure to obtain optimal solution
END

Figure 9: Flowchart of CRO for MWDS problem.

5/12 = 0.417, and 5/5 = 1. According to equation (13), the values of nodes 1, 2, 3, and 5 are 1/2 = 0.5, 2/2 = 1, 2/2 = 1, and 1/2 = 0.5 =. The ratios according to equation (15) are 6/1 = 6, 10/2 = 5, 12/2 = 6, and 5/1 = 5. According to equation (14), the values of nodes 1, 2, 3, and 5 are 5/6 = 0.83, 5/5 = 1, 5/6 = 0.833, and 5/5 = 1. The combined values of these three properties for nodes 1, 2, 3, and 5 are 2.16, 2.5, 2.25, and 2.5. Here the values of nodes 2 and 5 are maximum which is 2.5 for both cases. So for the first iteration from nodes 2 and 5, one of them is selected randomly. Then to cover up all the nodes, another node is selected. Finally, the minimum weighted dominating set is found, which is $\{2, 5, 6\}$ and the weight of this set is 20. The weight of the previous set was 23. So it is clear that repair operator helps to provide better results.

4.6. Common Neighbor Elimination. This method eliminates nodes that are common in the neighbor of any node and dominating set based on total weights. The process iterates for every node in a graph. In this process, the intersection between dominating set and the neighbor of a node has been produced for obtaining the total weight of the nodes of the intersection set. If the total weight of the intersection set is greater than the weight of the node then the node becomes an active node. Then the nodes of the intersection set have been deactivated by checking the covering condition. If any node becomes uncovered then it remains active. Finally, nodes in the intersection set have been removed from dominating set one by one by checking the objective value. If the objective value decreases then the molecule is updated with a new molecule, otherwise, it remains the same.



Remove 40% nodes from the dominating set
Add new nodes based on values from equation 11, 12 and 13

Figure 10: Illustration of repair operator.

Algorithm 10 demonstrates the procedure of eliminating neighbor members of dominating set.

5. Experimental Results

In this section, experimental results of CRO on given datasets are reported. The proposed CRO was run on a computer with a 2.7 GHz Intel Core i5 processor and 8 GB of RAM. The algorithm was run using python 3.7. Firstly, we introduce the used benchmark datasets. Then, we describe the method for setting the parameter values. Finally, computational results and comparisons are reported.

5.1. Dataset. In our proposed system two types of benchmark datasets were used which were used by different researchers to solve the MWDS problem using heuristic algorithms [21–24, 27]. The benchmark data are of two types (TI and TII), and another class is BHOSLIB. These datasets can be found in [21]. Each vertex weight of the type I is randomly generated in between 20 to 70. For the instances in type II, the weight of vertex i is randomly generated in (1, degree$(i)^2$), where degree(i) is the degree of vertex i. Both types I and II have 53 groups. There are 10 instances (the names of these 10 instances are n_m_0 to n_m_9) in each group. Instances with the same number of vertices (n) are classified as the same group. On the other hand, the complement graphs of the remaining instances in the BHOSLIB dataset were taken as useable graphs for the MWDS problem. To obtain the weights of the nodes of graphs, for each node, it is calculated as $\omega(i) = (i\%200) + 1$. Here, $\omega(i)$ refers to the weight of a single node i.

5.2. Setting Parameters. Because the initialization of parameters in CRO is not guaranteed, obtaining an appropriate parameter value for the optimal molecule is a difficult process. In our study, we strive to discover the best molecule with the smallest weight by adjusting various factors, and we provide the findings for several parameter sets. In the population generation stage, a new parameter (ϵ) is introduced. Figure 11 shows the tuning results of this parameter on some graph data of the TI instance. Besides this, Figures 12–14 refer to the different parameter tuning results of CRO which are the kinetic energy, alpha, and beta tuning. In all the graphs X-axis represents the total weight of the dominating set, and the Y-axis represents the parameter values. From the results of tuning, the parameter values are shown in Table 2.

```
     Input: one molecule, list of neighbors $(N)$, dominating set $(D)$, the weight of vertices.
     Output: updated molecule.
(1)  for $N(v_i) \in N$ do
(2)    $CN \longleftarrow N(v_i) \cap D$;// $CN$ refers to the list of common neighbors between the neighbor node of $v_i$ and dominating set
(3)    $T_\omega \longleftarrow \sum \omega(v_j \in CN)$
(4)    if $T_\omega > \omega(v_i)$ then
(5)      $M[v_i] \longleftarrow 1$ for $v_j \in CN$ do
(6)        $M[v_j] \longleftarrow 0$
(7)        Check the feasibility of the molecule.
(8)        if infeasible then
(9)          $M[v_j] \longleftarrow 1$
(10)       end
(11)       else
(12)         check the objective value of both the previous and updated molecule.
(13)         if infeasible then
(14)           $f \longleftarrow 0$
(15)         end
(16)         else
(17)           $f \longleftarrow 1$
(18)         end
(19)       end
(20)     end
(21)   end
(22)   if $f = 1$ then
(23)     updated molecule $\longleftarrow$ molecule with minimum objective value.
(24)   end
(25)   else
(26)     continue.
(27)   end
(28) end
```

Algorithm 10: Common neighbor elimination.

5.3. Performance Analysis. To analyze the performance of the proposed algorithm the minimum weights of the dominating set and the run time of the algorithm were considered. The existing related algorithms used different CPUs and different programming languages to obtain the output. Thus to compare the running time we implemented the HTS-DS approach in our implementation environment. As the HTS-DS approach obtained the best time compared with other algorithms, we show only the comparison between the proposed CRO-based method and the HTS-DS approach.

The comparison regarding minimum weights between MWDS_CRO with other algorithms has been shown in Tables 3–7. These tables depict the outcomes for types TI, TII, and BHOSLIB datasets, respectively. For both types TI and TII, the instances are divided into two classes which are small and medium particle instances (SMPI) and large particle instances (LPI). Here, the graphs with (50–250) nodes are classified as SMPI class while the others are represented as LPI class. Table 3 illustrates the minimum weights of class SMPI of type TI which are obtained after the implementation of different algorithms. Here, $|V|$ and $|E|$ refer to the number of vertices and edges in a graph. The outcome regarding weights of different algorithms of LPI class from TI type can be obtained from Table 4. Tables 5 and 6 refer to the minimum weights of various algorithms of classes SMPI and LPI of type TII. Table 7 shows the comparison of performance between HTS-DS and MWDS_CRO for the BHOSLIB dataset.

Tables 8 and 9 demonstrate the run times for all instances of types TI and TII successively for every individual algorithm.

From the tables, it is clear that the average results, as well as the run times for different instances of the proposed algorithm, are better than all other existing related algorithms.

Table 10 shows the total number of the best-known results obtained by each algorithm for every dataset. For example, the CC^2 FS algorithm has the best-known results in 24 instances out of 32 instances in the data set type TI, class-SMPI. Here, $'-'$ denotes the comparison of results with respect to best-known results that are not available for CC^2 FS, FBPSO, and ABC-EDA algorithms for the BHOSLIB dataset. In the column with the caption “improvement%” we compute the results as follows:

$$\text{Improvement\%} = \frac{(\text{TBR\_MWDS\_CRO} - \text{TBR\_HTS} - DS)}{TS} \times 100, \tag{16}$$

where TBR_MWDS_CRO means the total number of best-known results by the MWDS_CRO method and TBR_HTS $-$ DS denotes the total number of best-known

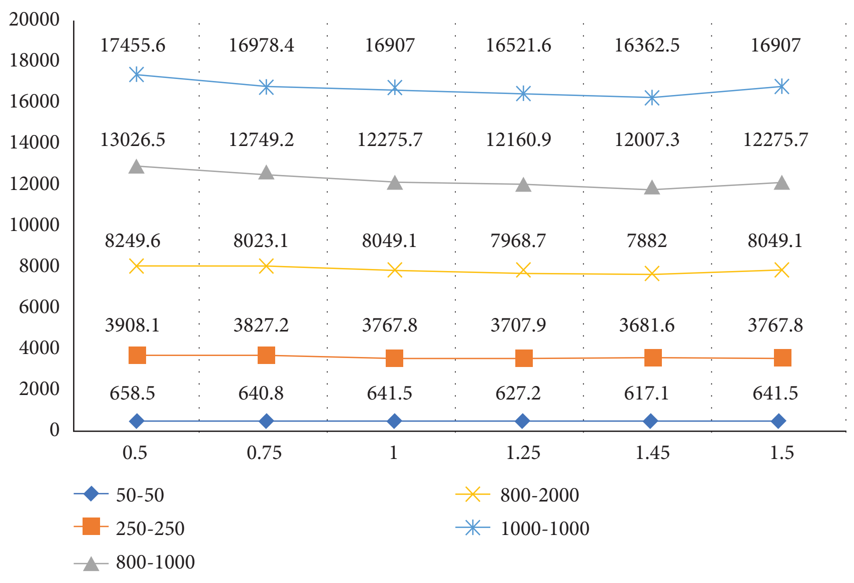


Figure 11: Epsilon (ϵ) tuning.

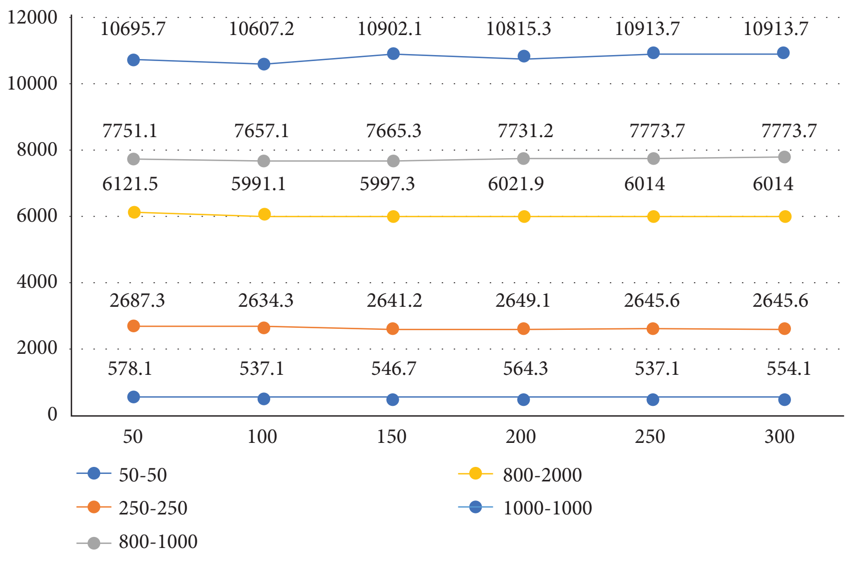


Figure 12: Kinetic energy (KE) tuning.

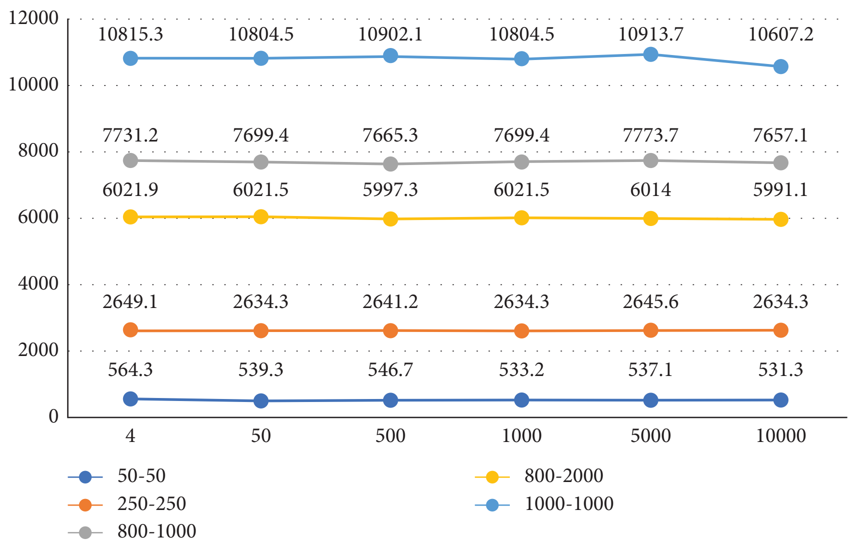


Figure 13: Alpha (α) tuning.

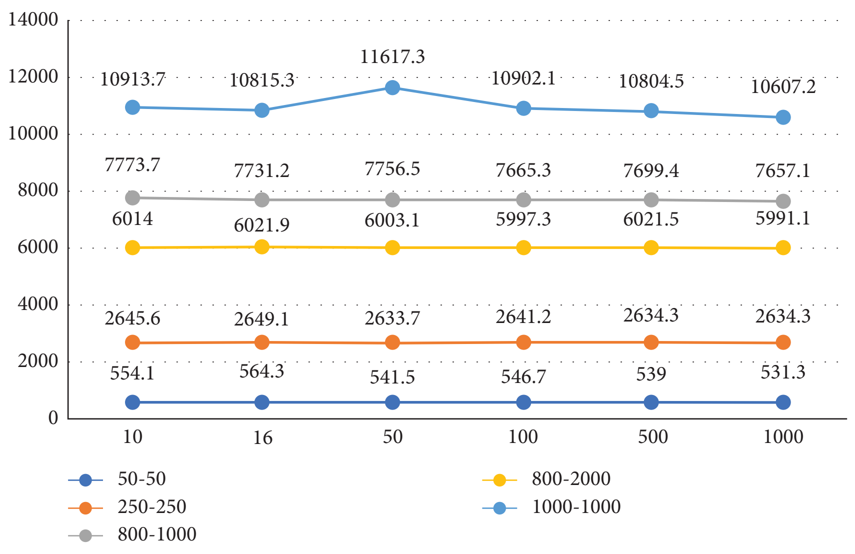


Figure 14: Beta (β) tuning.

Table 3: Type TI class SMPI-comparison of MWDS_CRO for various parameters with recent state-of-the-art algorithms.

$\|V\|$	$\|E\|$	CC^2 FS	FBPSO	HTS-DS	ABC-EDA	MWDS_CRO	Best known
50	50	531.3	531.3	531.3	531.3	531.3	531.3
	100	370.9	370.98	370.9	370.9	370.9	370.9
	250	175.7	175.7	175.7	175.7	175.7	175.7
	500	94.9	94.9	94.9	94.9	94.9	94.9
	750	63.3	63.16	63.1	63.1	63.1	63.1
	1000	41.5	41.5	41.5	41.5	41.5	41.5
100	100	1061.0	1062.99	1061.0	1061.4	1061.0	1061.0
	250	618.9	619.48	618.9	621.1	618.9	618.9
	500	355.6	355.74	355.6	355.8	355.6	355.6
	750	255.8	255.95	255.8	258.0	255.8	255.8
	1000	203.6	203.6	203.6	205.2	203.6	203.6
	2000	107.4	107.88	107.4	107.9	107.4	107.4
150	150	1580.5	1585.71	1580.5	1583.8	1580.5	1580.5
	250	1218.2	1223.31	1218.2	1223.2	1218.2	1218.2
	500	744.6	747.45	744.6	747.5	744.6	744.6
	750	546.1	548.15	546.1	549.2	546.1	546.1
	1000	432.9	433.93	432.8	434.8	432.8	432.8
	2000	240.8	241.47	240.8	243.7	240.8	240.8
	3000	166.9	167.53	166.9	169.3	166.9	166.9
200	250	1910.4	1918.82	1909.7	1916.7	1909.7	1909.7
	500	1232.8	1239.71	1232.8	1243.4	1232.8	1232.8
	750	911.2	918.47	911.2	924.2	911.2	911.2
	1000	724.0	727.09	723.5	729.0	723.5	723.5
	2000	412.7	415.46	412.9	417.9	412.9	412.7
	3000	292.8	294.03	292.8	298.1	292.8	292.8
250	250	2633.4	2649.73	2633.0	2647.0	2633.0	2633.0
	500	1805.9	1813.03	1805.9	1817.9	1805.9	1805.9
	750	1362.2	1375.32	1361.9	1382.6	1361.9	1361.9
	1000	1091.1	1099.65	1090.1	1107.7	1089.9	1089.9
	2000	621.9	625.41	621.9	628.6	621.6	621.6
	3000	447.9	452.45	448.0	457.1	448.0	447.9
	5000	289.5	291.18	289.6	297.4	289.5	289.5

Table 4: Type TI class LPI-comparison of MWDS_CRO for various parameters with recent state-of-the-art algorithms.

$\|V\|$	$\|E\|$	CC^2 FS	FBPSO	HTS-DS	ABC-EDA	MWDS_CRO	Best known
300	300	3178.6	3203.62	3175.4	3193.0	3175.4	3175.4
	500	2438.1	2453.57	2435.6	2461.3	2435.6	2435.6
	750	1854.6	1870.8	1853.9	1876.0	1853.8	1853.8
	1000	1495.0	1506.56	1494.1	1516.7	1494.0	1494.0
	2000	862.5	871.89	862.4	878.8	862.4	862.4
	3000	624.3	628.24	624.4	636.8	624.1	624.1
	5000	406.1	409.08	406.3	417.7	406.1	406.1
500	500	5305.7	5392.45	5304.7	5370.7	5304.7	5304.7
	1000	3607.8	3659.87	3608.6	3652.1	3607.6	3607.3
	2000	2181.0	2210.28	2177.8	2225.0	2176.8	2176.8
	5000	1043.3	1054.88	1044.2	1073.5	1042.3	1042.3
	10000	587.2	594.32	587.2	605.8	587.2	587.2
800	1000	7663.4	7771.52	7655.0	7755.7	7655.0	7655.0
	2000	4982.1	5060.73	4991.7	5079.1	4987.3	4982.1
	5000	2441.2	2470.38	2435.8	2492.2	2432.6	2432.6
	10000	1395.6	1417.16	1395.1	1440.0	1393.7	1393.7
1000	1000	10585.3	10785.37	10574.4	10766.4	10574.4	10574.4
	5000	3671.8	3713.22	3662.7	3757.4	3656.6	3656.6
	10000	2109.0	2132.76	2102.2	2180.8	2099.8	2099.8
	15000	1521.5	1542.64	1521.9	1573.6	1519.7	1519.7
	20000	1203.6	1215.98	1205.6	1242.1	1200.9	1200.9

results by HTS-DS, and TS is the total number of instances of the respective data set. We considered HTS-DS for comparisons because this algorithm is a state-of-the-art method.

In Table 11, we have shown the comparison in average time improvement for all the test datasets. The time improvement in % has been computed using by using the following equation:

$$\text{Time improvementin\%} = \frac{\text{Avg}\left(\text{T}_{\text{HTS-DS}}\right) - \text{Avg}\left(\text{T}_{\text{MWDS\_CRO}}\right)}{\text{Avg}\left(\text{T}_{\text{HTS-DS}}\right)} \times 100, \tag{17}$$

where Avg$(T_{\text{HTS}-DS}$ refers to the average running time in second(s) for HTS-DS and Avg$(T_{\text{MWDS_CRO}})$ denotes the average running time in second(s) for MWDS_CRO. Here we have considered the time of HTS-DS and MWDS-CRO algorithms. Since these two algorithms have the best-known results for most of the instances of all datasets. Our method improves average computational time by 92.97% for the TI-SMPI dataset which is the highest compared to all datasets, whereas the lowest improvement is 71.43% which is for the TII-SMPI dataset.

Table 12 shows the outcome for different components of the Type TI dataset. According to the table, it is clear that from the basic CRO operators, the best outcome is obtained for most instances. However, there are some instances in which the best results cannot be found after the basic CRO operators. In these cases, repair operators and common neighbor elimination procedure work to obtain the best results.

The time complexity of metaheuristic algorithms for searching the global optimal solution cannot be determined because these algorithms do not guarantee the finding of the global optimal solution within a given time limit. The metaheuristic algorithms such as CRO, GA, PSO, and ACO have no predefined end. Some metaheuristics need a few hundred or thousand iterations (with a big computational effort for each iteration) to obtain good or best results, while others need several million iterations (with only a tiny computational effort for each iteration).

In the proposed algorithm, the CRO performed 100 iterations in one run. Moreover, in one full iteration, the algorithm performs 20 evaluations (popSize = 20). So, the number of evaluations in one run is $100 \times 20 = 2000$. After performing the basic CRO operators, the obtained solutions went through repair operators and a common neighbor elimination procedure. Both of these methods run 1 time on the solution to obtain the best solution. For repair operator time complexity is $\mathcal{O}(n)$ and for the common neighbor elimination procedure it is $\mathcal{O}(mn)$. Here, n refers to the total number of nodes in a graph and m refers to the total number of neighbors of any node. So finally the total number of evaluations is $100 \times 20 + 1 = 2001$.

5.4. Statistical Significance Test.

The statistical significance of MWDS_CRO over other algorithms is shown in this subsection by showing Wilcoxon signed-rank test. From Tables 13 and 14, it is clear that MWDS_CRO is more significant than any other algorithms for both TI and TII instances. For the Wilcoxon test, for both instances, the value of the significance level is 0.05, and the hypothesis is two-tailed.

Due to the insufficient sample size of MWDS_CRO and HTS-DS in Table 11 for the type TII instance, mean (W), standard deviation (W), and p value can not be calculated. Thus the significance of the result cannot be shown as the significance of the result depends on the p value. If the p value is less than 0.05 then the result is significant.

Table 5: Type TII class SMPI-comparison of MWDS_CRO for various parameters with recent state-of-the-art algorithms.

$\|V\|$	$\|E\|$	CC^2 FS	FBPSO	HTS-DS	ABC-EDA	MWDS_CRO	Best known
50	50	60.8	60.8	60.8	60.8	60.8	60.8
	100	90.3	90.3	90.3	90.3	90.3	90.3
	250	146.7	146.7	146.7	146.7	146.7	146.7
	500	179.9	179.9	179.9	179.9	179.9	179.9
	750	171.1	171.1	171.1	171.1	171.1	171.1
	1000	146.5	146.5	146.5	146.5	146.5	146.5
100	100	123.5	123.5	123.5	123.5	123.5	123.5
	250	209.2	209.23	209.2	209.6	209.2	209.2
	500	305.7	305.72	305.7	305.7	305.7	305.7
	750	384.5	384.5	384.5	384.5	384.5	384.5
	1000	427.3	427.3	427.3	427.3	427.3	427.3
	2000	550.6	550.6	550.6	550.6	550.6	550.6
150	150	184.5	184.59	184.5	184.5	184.5	184.5
	250	232.8	233.04	232.8	233.1	232.8	232.8
	500	349.5	349.68	349.5	349.5	349.5	349.5
	750	452.4	452.4	452.4	452.8	452.4	452.4
	1000	547.2	547.2	547.2	547.2	547.2	547.2
	2000	720.1	720.1	720.1	720.1	720.1	720.1
	3000	792.4	792.48	792.4	792.4	792.4	792.4
200	250	271.7	272.06	271.7	271.7	271.7	271.7
	500	386.7	386.77	386.7	386.8	386.7	386.7
	750	497.1	497.15	497.1	497.3	497.1	497.1
	1000	596.8	597.21	596.8	597.3	596.8	596.8
	2000	884.6	884.63	884.6	884.6	884.6	884.6
	3000	1019.2	1019.45	1019.2	1022.4	1019.2	1019.2
250	250	306.1	306.79	306.0	306.1	306.0	306.0
	500	440.7	441.72	440.7	440.8	440.7	440.7
	750	567.4	568.63	567.4	568.2	567.4	567.4
	1000	668.6	669.26	668.6	669.5	668.6	668.6
	2000	1007.0	1007.91	1007.0	1010.2	1007.0	1007.0
	3000	1250.6	1251.57	1250.6	1250.6	1250.6	1250.6
	5000	1464.2	1464.2	1464.2	1464.2	1464.2	1464.2

Table 6: Type TII class LPI-comparison of MWDS_CRO for various parameters with recent state-of-the-art algorithms.

$\|V\|$	$\|E\|$	CC^2 FS	FBPSO	HTS-DS	ABC-EDA	MWDS_CRO	Best known
300	300	369.9	370.63	369.9	369.9	369.9	369.9
	500	477.8	478.32	477.8	478.4	477.8	477.8
	750	613.3	614.73	613.3	613.9	613.3	613.3
	1000	737.9	738.93	737.7	738.1	737.7	737.7
	2000	1093.8	1094.23	1093.8	1095.2	1093.8	1093.8
	3000	1358.5	1359.57	1358.5	1361.3	1358.5	1358.5
	5000	1682.7	1683.2	1682.7	1682.7	1682.7	1682.7
500	500	623.6	627.24	623.6	623.7	623.6	623.6
	1000	899.8	904.24	899.6	901.6	899.6	899.6
	2000	1363.3	1373.33	1362.2	1367.3	1362.2	1362.2
	5000	2333.7	2335.41	2326.6	2342.1	2326.6	2326.6
	10000	3211.5	3211.8	3211.5	3214.5	3211.5	3211.5
800	1000	1104.3	1113.03	1103.9	1104.5	1103.9	1103.9
	2000	1632.3	1641.87	1630.8	1636.1	1630.8	1630.8
	5000	2878.5	2901.62	2876.6	2888.1	2876.0	2876.1
	10000	4105.6	4107.19	4104.0	4121.5	4102.1	4102.8
1000	1000	1237.7	1249.7	1237.5	1239.4	1237.5	1237.5
	5000	3178.7	3206.7	3180.0	3194.4	3172.9	3172.9
	10000	4711.8	4733.58	4709.9	4748.1	4704.5	4704.8
	15000	5874.2	5896.91	5862.4	5897.6	5856.4	5856.4
	20000	6662.1	6681.64	6657.9	6691.5	6655.1	6655.1

Table 7: BHOSLIB-comparison of MWDS_CRO for various parameters with the recent state-of-the-art algorithm.

Instances			HTS-DS		MWDS_CRO		Best known
Name	$\|V\|$	$\|E\|$	Avg	T (s)	Avg	T (s)	—
frb30-15-1	450	17827	212.0	41.6	212.0	8.12	212.0
frb30-15-2	450	17874	242.0	14.32	242.0	4.56	242.0
frb30-15-3	450	17809	175.0	21.67	175.0	5.51	175.0
frb30-15-4	450	17831	166.0	14.57	166.0	2.23	166.0
frb30-15-5	450	17794	160.0	17.3	160.0	3.59	160.0
frb35-17-1	595	27856	274.0	44.55	274.0	10.76	274.0
frb35-17-2	595	27847	208.0	18.17	208.0	3.02	208.0
frb35-17-3	595	27931	201.0	19.61	201.0	6.63	201.0
frb35-17-4	595	27842	286.0	52.87	286.0	13.33	286.0
frb35-17-5	595	28143	295.0	12.74	295.0	2.41	295.0
frb40-19-1	760	41314	262.0	20.22	262.0	5.60	262.0
frb40-19-2	760	41263	243.0	80.41	243.0	12.51	243.0
frb40-19-3	760	41095	250.0	78.71	250.0	16.06	250.0
frb40-19-4	760	41605	249.1	71.87	249.0	14.18	249.0
frb40-19-5	760	41619	272.0	63.69	272.0	10.87	272.0
frb45-21-1	945	59186	328.0	76.98	328.0	13.96	328.0
frb45-21-2	945	58624	259.2	87.38	259.0	22.34	259.0
frb45-21-3	945	58245	233.0	132.06	233.0	30.53	233.0
frb45-21-4	945	58549	399.0	79.67	399.0	19.49	399.0
frb45-21-5	945	58579	312.5	92.06	312.0	20.79	312.0
frb50-23-1	1150	80072	261.0	159.38	261.0	37.11	261.0
frb50-23-2	1150	80851	277.0	161.75	277.0	38.93	277.0
frb50-23-3	1150	81068	281.0	125.9	281.0	25.07	281.0
frb50-23-4	1150	80258	265.0	191.06	265.0	40.58	265.0
frb50-23-5	1150	80035	404.3	108.5	408.0	12.10	404.0
frb53-24-1	1272	94227	229.0	86.89	229.0	7.99	229.0
frb53-24-2	1272	94289	298.0	348.12	298.0	52.68	298.0
frb53-24-3	1272	94127	182.0	212.98	182.0	41.26	182.0
frb53-24-4	1272	94308	189.0	284.44	189.0	58.56	189.0
frb53-24-5	1272	94226	204.0	78.23	204.0	16.21	204.0
frb56-25-1	1400	109676	229.0	107.66	229.0	12.90	229.0
frb56-25-2	1400	109401	319.0	145.09	319.0	39.02	319.0
frb56-25-3	1400	109379	336.0	163.23	336.0	48.88	336.0
frb56-25-4	1400	110038	265.0	169.65	265.0	51.09	265.0
frb56-25-5	1400	109601	411.4	128.27	408.0	32.88	408.0
frb59-26-1	1534	126555	262.6	192.54	262.0	54.39	262.0
frb59-26-2	1534	126163	386.6	159.25	383.0	33.74	383.0
frb59-26-3	1534	126082	246.7	348.38	246.0	92.40	246.0
frb59-26-4	1534	127011	248.0	357.5	248.0	86.66	248.0
frb59-26-5	1534	125982	288.0	396.44	288.4	98.23	288.0
frb100-40	4000	572774	350.0	963.58	350.0	181.16	350.0

Table 8: Type TI-run time comparison in second of MWDS_CRO with the recent state-of-the-art algorithm.

$\|V\|$	$\|E\|$	HTS-DS	MWDS_CRO
50	50	0.42	0.36
	100	0.42	0.36
	250	1.10	1.07
	500	2.01	1.81
	750	1.82	2.25
	1000	2.59	3.01
100	100	0.64	0.90
	250	1.61	1.01
	500	2.93	2.13
	750	3.25	3.18
	1000	6.89	5.39
	2000	12.93	13.41
150	150	1.61	1.01
	250	1.93	0.77
	500	9.63	1.24
	750	22.47	2.79
	1000	26.00	4.14
	2000	25.04	14.66
	3000	30.08	26.79
200	250	1.93	1.19
	500	6.42	1.58
	750	25.04	11.56
	1000	25.04	19.92
	2000	25.68	16.04
	3000	23.11	12.09
250	250	3.53	1.38
	500	7.70	1.81
	750	26.32	18.20
	1000	25.36	21.26
	2000	26.96	16.62
	3000	24.40	26.53
	5000	25.04	1.33
300	300	4.17	1.58
	500	5.14	1.08
	750	24.08	3.29
	1000	27.93	4.78
	2000	28.57	2.49
	3000	30.50	1.26
	5000	26.00	1.92
500	500	8.67	3.17
	1000	33.38	8.69
	2000	34.67	7.29
	5000	34.99	6.48
	10000	31.78	4.25
800	1000	16.69	6.16
	2000	45.90	7.80
	5000	45.58	8.96
	10000	44.62	7.26
1000	1000	27.93	8.02
	5000	50.40	12.56
	10000	51.04	13.65
	15000	52.00	9.68
	20000	57.46	10.28

Table 9: Type TII-run time comparison in second of MWDS_CRO with the recent state-of-the-art algorithm.

$\|V\|$	$\|E\|$	HTS-DS	MWDS_CRO
50	50	0.32	0.13
	100	0.32	0.12
	250	0.13	0.10
	500	0.13	0.08
	750	0.13	0.08
	1000	0.32	0.07
100	100	0.64	0.24
	250	0.32	0.23
	500	0.32	0.26
	750	0.64	0.26
	1000	0.64	0.28
	2000	0.96	0.24
150	150	1.61	0.47
	250	0.96	0.32
	500	0.96	0.39
	750	1.28	0.40
	1000	1.28	0.39
	2000	1.61	0.45
	3000	2.57	0.43
200	250	1.61	0.56
	500	1.28	0.56
	750	1.28	0.47
	1000	1.61	0.51
	2000	1.93	0.62
	3000	3.21	0.61
250	250	3.53	0.79
	500	2.25	0.90
	750	2.25	0.89
	1000	2.25	0.69
	2000	3.53	1.01
	3000	4.49	1.12
	5000	7.06	0.97
300	300	4.17	1.20
	500	2.89	1.54
	750	2.89	1.36
	1000	3.21	1.78
	2000	3.53	2.00
	3000	4.82	1.82
	5000	8.99	1.50
500	500	8.99	1.06
	1000	6.42	1.89
	2000	7.70	2.88
	5000	10.91	2.68
	10000	27.29	4.07
800	1000	13.80	9.29
	2000	13.16	12.80
	5000	34.67	24.34
	10000	69.02	20.65
1000	1000	28.25	8.06
	5000	53.29	9.12
	10000	100.15	13.17
	15000	117.17	10.18
	20000	178.48	12.87

Table 10: Comparison of results with respect to best-known results.

Data set	No. of instances	CC^2 FS	FBPSO	HTS-DS	ABC-EDA	MWDS_CRO	Improvement (%)
TI-SMPI	32	24	5	27	6	30	9.38
TI-LPI	21	3	0	6	0	19	61.9
TII-SMPI	32	31	14	32	20	32	0
TII-LPI	21	8	0	16	2	21	23.81
BHOSLIB	41	—	—	33	—	39	14.63

Table 11: Comparison in average time improvement for datasets.

Data set	No. of instances	Avg T (s) in HTS-DS	Avg T (s) in the proposed method	Avg T (s) improvement	Improvement in (%)
TI-SMPI	32	10.95	0.77	10.18	92.97
TI-LPI	21	32.45	6.22	26.23	80.83
TII-SMPI	32	1.61	0.46	1.15	71.43
TII-LPI	21	33.32	6.87	26.45	79.38
BHOSLIB	41	144.62	31.42	113.2	78.27

Table 12: Type TI comparison of MWDS_CRO for different components.

$\|V\|$	$\|E\|$	Ranking based population	Without repair operator and common neighbor elimination	Without common neighbor elimination	With repair operators and common neighbor elimination
50	50	555.7	531.3	531.3	531.3
	100	427.6	370.9	370.9	370.9
	250	217.0	175.7	175.7	175.7
	500	148.8	94.9	94.9	94.9
	750	92.9	63.1	63.1	63.1
	1000	62.6	41.5	41.5	41.5
100	100	1189.4	1090.0	1061.0	1061.0
	250	765.5	618.9	618.9	618.9
	500	480.5	355.6	355.6	355.6
	750	346.1	255.8	255.8	255.8
	1000	293.7	203.6	203.6	203.6
	2000	167.1	107.4	107.4	107.4
150	150	1760.6	1580.5	1580.5	1580.5
	250	1466.3	1218.2	1218.2	1218.2
	500	983.6	744.6	744.6	744.6
	750	753.7	546.1	546.1	546.1
	1000	591.5	432.8	432.8	432.8
	2000	344.8	240.8	240.8	240.8
	3000	252.3	166.9	166.9	166.9
200	250	2234.6	1909.7	1909.7	1909.7
	500	1588.4	1308.4	1238.2	1232.8
	750	1240.9	1038.6	911.2	911.2
	1000	1022.4	912.8	723.5	723.5
	2000	578.3	412.7	412.7	412.7
	3000	429.9	344.6	312.2	292.8
250	250	3030.5	2812.8	2633.0	2633.0
	500	2303.2	2108.0	1805.9	1805.9
	750	1861.2	1421.7	1361.9	1361.9
	1000	1486.6	1242.2	1089.9	1089.9
	2000	891.4	621.6	621.6	621.6
	3000	657.7	447.9	447.9	447.9
	5000	452.2	289.5	289.5	289.5
300	300	4221.6	3456.5	3223.9	3175.4
	500	3040.2	2652.7	2556.4	2435.6
	750	2256.6	1853.8	1853.8	1853.8
	1000	2052.4	1816.2	1652.4	1494.0
	2000	1240.7	862.4	862.4	862.4
	3000	1080.7	624.1	624.1	624.1
	5000	623.7	406.1	406.1	406.1
500	500	9852.4	6121.3	5702.1	5304.7
	1000	6045.7	4087.3	3607.3	3607.3
	2000	3668.0	2176.8	2176.8	2176.8
	5000	3351.7	1624.2	1218.6	1042.3
	10000	867.2	587.2	587.2	587.2
800	1000	12232.6	8765.4	7655.0	7655.0
	2000	8668.0	5246.7	4982.1	4982.1
	5000	4612.2	2432.6	2432.6	2432.6
	10000	1890.9	1393.7	1393.7	1393.7
1000	1000	18656.8	12231.7	10574.4	10574.4
	5000	6221.3	3656.6	3656.6	3656.6
	10000	4152.6	2099.8	2099.8	2099.8
	15000	2051.3	1519.7	1519.7	1519.7
	20000	1621.5	1200.9	1200.9	1200.9

Table 13: Wilcoxon signed-rank test for TI instances.

	MWDS_CRO and CC^2 FS	MWDS_CRO and FBPSO	MWDS_CRO and HTS-DS	MWDS_CRO and ABC-EDA
W value	31.5	0	0	0
Mean difference	2139.51	1311.64	3647.31	1345.55
Sum of positive ranks	403.5	1176	153	1128
Sum of negative ranks	31.5	0	0	0
Z value	−4.0219	−6.0308	−3.6214	−5.9683
Mean (W)	217.5	588	76.5	564
Standard deviation (W)	46.25	97.5	21.12	94.5
Sample size (N)	29	48	17	47
p value	<0.00001	<0.00001	0.0003	<0.00001
Result	Significant	Significant	Significant	Significant

Table 14: Wilcoxon signed-rank test for TII instances.

	MWDS_CRO and CC^2 FS	MWDS_CRO and FBPSO	MWDS_CRO and HTS-DS	MWDS_CRO and ABC-EDA
W value	0	0	0	0
Mean difference	4057.2	1461.05	9461.65	1837.74
Sum of positive ranks	105	780	21	496
Sum of negative ranks	0	0	0	0
Z value	−3.2958	−5.4424	−2.2014	−4.8599
Mean (W)	52.5	390	—	248
Standard deviation (W)	15.93	71.66	—	51.03
Sample size (N)	14	39	6	31
p value	0.00096	<0.00001	—	<0.00001
Result	Significant	Significant	—	Significant

6. Conclusion

More research on the MWDS problem is unavoidable because it has a wide range of practical applications, including data mining, the study of social networks and influence transmission, protein interaction networks, and covering codes. Due to its NP-hardness, this problem cannot be solved in a limited time using an exact algorithm. Thus implementation of a metaheuristic algorithm can obtain the proper solution. To cope with the issue, in this experiment, CRO based model has been proposed to solve the MWDS problem where population generates based on ranking. Then the population is intensified by applying the molecule intensification procedure. Then CRO operators work on the generated population. Besides CRO operators, the coordination of three supporting operators helps to eradicate infeasibility and redundancy from the up-to-date population. The further procurement of better offspring depends on the repair operator and the common neighbor elimination procedure. To prove the supremacy of our model, we compare the results with other well-known related algorithms, and the significance test is also provided. From the significance tests and comparison results, it is clear that our proposed model outperforms the other related existing algorithms.

Data Availability

The data sets used in this research are available publicly at the following link: https://drive.Google.com/drive/folders/1bTuTVYorJ6Qy5uwNLq5XAKIPKyu84zL4?usp=sharing.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by ICT-Division, Ministry of Posts, Telecommunications and Information Technology, Bangladesh. This support is gratefully acknowledged.

References

[1] M. M. Daliri, A. Rezvanian, N. Bagherpour, and M. R. Meybodi, “Minimum positive influence dominating set and its application in influence maximization: a learning automata approach,” *Applied Intelligence*, vol. 48, no. 3, pp. 570–593, 2018.

[2] J. C. Nacher and T. Akutsu, “Minimum dominating set-based methods for analyzing biological networks,” *Methods*, vol. 102, pp. 57–63, 2016.

[3] S. Wuchty, “Controllability in protein interaction networks,” *Proceedings of the National Academy of Sciences*, vol. 111, no. 19, pp. 7156–7160, 2014.

[4] J. Yu, N. Wang, G. Wang, and D. Yu, “Connected dominating sets in wireless ad hoc and sensor networks–a comprehensive survey,” *Computer Communications*, vol. 36, no. 2, pp. 121–134, 2013.

[5] P. R. J. Ostergard, “Constructing covering codes by tabu search,” *Journal of Combinatorial Designs*, vol. 5, no. 1, pp. 71–80, 1997.

[6] Y. Wang, W. Wang, and X. Y. Li, “Efficient distributed low-cost backbone formation for wireless networks,” *IEEE*

Transactions on Parallel and Distributed Systems, vol. 17, no. 7, pp. 681–693, 2006.

[7] D. Dai and C. Yu, "A 5+ approximation algorithm for minimum weighted dominating set in unit disk graph," *Theoretical Computer Science*, vol. 410, no. 8-10, pp. 756–765, 2009.

[8] F. Zou, Y. Wang, X. H. Xu et al., "New approximations for minimum-weighted dominating sets and minimum-weighted connected dominating sets on unit disk graphs," *Theoretical Computer Science*, vol. 412, no. 3, pp. 198–208, 2011.

[9] M. El Houmaidi and M. A. Bassiouni, "k-weighted minimum dominating sets for sparse wavelength converters placement under nonuniform traffic," in *Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems*, pp. 56–61, Manhattan, NY, USA, October 2003.

[10] D. Subhadrabandhu, S. Sarkar, and F. Anjum, "Efficacy of misuse detection in ad hoc networks," in *Proceedings of the 2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, pp. 97–107, Santa Clara, CA, USA, October 2004.

[11] P. Wu, J. R. Wen, H. Liu, and W. Y. Ma, "Query selection techniques for efficient crawling of structured web sources," in *Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*, p. 47, IEEE, Atlanta, GA, USA, April 2006.

[12] N. Alon, D. Moshkovitz, and S. Safra, "Algorithmic construction of sets for k-restrictions," *ACM Transactions on Algorithms*, vol. 2, no. 2, pp. 153–177, 2006.

[13] W. Liu, Y. J. Gong, W. N. Chen, Z. Liu, H. Wang, and J. Zhang, "Coordinated charging scheduling of electric vehicles: a mixed-variable differential evolution approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 12, pp. 5094–5109, 2020.

[14] F. Zhao, R. Ma, and L. Wang, "A self-learning discrete jaya algorithm for multiobjective energy-efficient distributed no-idle flow-shop scheduling problem in heterogeneous factory system," *IEEE Transactions on Cybernetics*, vol. 52, no. 12, pp. 12675–12686, 2022.

[15] F. Zhao, X. He, and L. Wang, "A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of no-wait flow-shop problem," *IEEE Transactions on Cybernetics*, vol. 51, no. 11, pp. 5291–5303, 2021.

[16] 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," *Computers & Industrial Engineering*, vol. 153, Article ID 107082, 2021.

[17] F. Zhao, S. Di, J. Cao, and J. Tang, "A novel cooperative multi-stage hyper-heuristic for combination optimization problems," *Complex System Modeling and Simulation*, vol. 1, no. 2, pp. 91–108, 2021.

[18] S. Zhou, L. Xing, X. Zheng, N. Du, L. Wang, and Q. Zhang, "A self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times," *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1430–1442, 2021.

[19] M. R. G. Johnson, "Computers and intractability: a guide to the theory of np-completeness," *WH Freeman & Company*, New York, NY, USA, 1979.

[20] E. J. Cockayne, R. M. Dawes, and S. T. Hedetniemi, "Total domination in graphs," *Networks*, vol. 10, no. 3, pp. 211–219, 1980.

[21] M. Albuquerque and T. Vidal, "An efficient matheuristic for the minimum-weight dominating set problem," *Applied Soft Computing*, vol. 72, pp. 527–538, 2018.

[22] G. Lin and J. Guan, "A binary particle swarm optimization for the minimum weight dominating set problem," *Journal of Computer Science and Technology*, vol. 33, no. 2, pp. 305–322, 2018.

[23] R. Jovanovic, "Ant colony optimization applied to minimum weight dominating set problem," in *Proceedings of the 12th WSEAS International Conference on Automatic Control*, pp. 29–31, Modelling & Simulation, Catania, Italy, July 2010.

[24] A. Potluri and A. Singh, "Hybrid metaheuristic algorithms for minimum weight dominating set," *Applied Soft Computing*, vol. 13, no. 1, pp. 76–88, 2013.

[25] S. Bouamama and C. Blum, "A hybrid algorithmic model for the minimum weight dominating set problem," *Simulation Modelling Practice and Theory*, vol. 64, pp. 57–68, 2016.

[26] G. Lin, W. Zhu, and M. M. Ali, "An effective hybrid memetic algorithm for the minimum weight dominating set problem," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 6, pp. 892–907, 2016.

[27] Y. Wang, S. Cai, and M. Yin, "Local search for minimum weight dominating set with two-level configuration checking and frequency based scoring function," *Journal of Artificial Intelligence Research*, vol. 58, pp. 267–295, 2017.

[28] J. Q. James, A. Y. S. Lam, and V. O. K. Li, "Evolutionary artificial neural network based on chemical reaction optimization," in *Proceedings of the 2011 IEEE Congress of Evolutionary Computation (CEC)*, pp. 2083–2090, IEEE, New Orleans, LA, USA, June 2011.

[29] T. K. Truong, K. Li, and Y. Xu, "Chemical reaction optimization with greedy strategy for the 0–1 knapsack problem," *Applied Soft Computing*, vol. 13, no. 4, pp. 1774–1780, 2013.

[30] M. D. Islam, C. M. Saifullah, and M. D. Mahmud, "Chemical reaction optimization: survey on variants," *Evolutionary Intelligence*, vol. 12, pp. 1–26, 2019.

[31] B. Pan, A. Y. S. Lam, and V. O. K. Li, "Network coding optimization based on chemical reaction optimization," in *Proceedings of the 2011 IEEE Global Telecommunications Conference-GLOBECOM 2011*, pp. 1–5, IEEE, Houston, TX, USA, January 2011.

[32] R. Islam, I. H. Arif, and R. H. Shuvo, "Generalized vertex cover using chemical reaction optimization," *Applied Intelligence*, vol. 49, no. 7, pp. 2546–2566, 2019.

[33] P. Khan-Boni and B. Shahriar-Abir, "Handwritten bangla digit recognition using chemical reaction optimization," in *Proceedings of the 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–7, IEEE, Bengaluru, India, July 2018.

[34] C. M. Khaled-Saifullah and M. Rafiqul-Islam, "Chemical reaction optimization for solving shortest common supersequence problem," *Computational Biology and Chemistry*, vol. 64, pp. 82–93, 2016.

[35] A. Y. S. Lam and V. O. K. Li, "Chemical reaction optimization for cognitive radio spectrum allocation," in *Proceedings of the 2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pp. 1–5, IEEE, Miami, FL, USA, December 2010.

[36] J. Xu, A. Y. S. Lam, and V. O. K. Li, "Stock portfolio selection using chemical reaction optimization," *International Journal of Chemical and Molecular Engineering*, vol. 5, no. 5, pp. 423–428, 2011.

[37] A. Y. S. Lam and V. O. K. Li, "Chemical-reaction-inspired metaheuristic for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 381–399, 2010.

[38] J. Xu, A. Y. S. Lam, and V. O. K. Li, “Parallel chemical reaction optimization for the quadratic assignment problem,” in *World Congress in Computer Science, Computer Engineering, and Applied Computing, World 2010*, The World Congress on Engineering, London, UK, 2010.

[39] M. R. Islam, M. R. Mahmud, and R. M. Pritom, “Transportation scheduling optimization by a collaborative strategy in supply chain management with tpl using chemical reaction optimization,” *Neural Computing & Applications*, vol. 32, no. 8, pp. 3649–3674, 2019.

[40] R. Kabir and R. Islam, “Chemical reaction optimization for rna structure prediction,” *Applied Intelligence*, vol. 49, no. 2, pp. 352–375, 2019.

[41] M. R. Islam, C. M. K. Saifullah, Z. T. Asha, and R. Ahamed, “Chemical reaction optimization for solving longest common subsequence problem for multiple string,” *Soft Computing*, vol. 23, no. 14, pp. 5485–5509, 2019.

[42] M. Shams Wadud, M. R. Islam, N. Kundu, and M. Rayhanul Kabir, “Multiple sequence alignment using chemical reaction optimization algorithm,” in *Proceedings of the International Conference on Intelligent Systems Design and Applications*, pp. 1065–1074, Springer, Berlin, Germany, September 2018.

[43] A. Bhattacharjee, M. Rahad, and I. Rafiqul, “Phylogenetic tree construction using chemical reaction optimization,” in *Proceedings of the International Conference on Intelligent Systems Design and Applications*, pp. 915–924, Springer, Berlin, Germany, July 2018.

[44] M. R. Islam, R. A. Smrity, S. Chatterjee, and M. R. Mahmud, “Optimization of protein folding using chemical reaction optimization in hp cubic lattice model,” *Neural Computing & Applications*, vol. 32, no. 8, pp. 3117–3134, 2019.

[45] M. R. Islam, M. S. Islam, and N. Sakeef, “Rna secondary structure prediction with pseudoknots using chemical reaction optimization algorithm,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 18, no. 3, pp. 1195–1207, 2021.

[46] R. Bar-Yehuda and S. Moran, “On approximation problems related to the independent set and vertex cover problems,” *Discrete Applied Mathematics*, vol. 9, pp. 1–10, 1984.

[47] M. O. Ball, “Heuristics based on mathematical programming,” *Surveys in Operations Research and Management Science*, vol. 16, no. 1, pp. 21–38, 2011.

[48] C. Archetti and M. Speranza, “A survey on matheuristics for routing problems,” *EURO Journal on Computational Optimization*, vol. 2, no. 4, pp. 223–246, 2014.

[49] F. Yuan, C. Li, X. Gao, M. Yin, and Y. Wang, “A novel hybrid algorithm for minimum total dominating set problem,” *Mathematics*, vol. 7, no. 3, p. 222, 2019.

[50] S. Shetgaonkar and A. Singh, “Hybridization of artificial bee colony algorithm with estimation of distribution algorithm for minimum weight dominating set problem,” in *ICT Systems and Sustainability*, pp. 607–619, Springer, Berlin, Germany, 2021.

[51] A. Y. S. Lam and V. O. K. Li, “Chemical reaction optimization: a tutorial,” *Memetic Computing*, vol. 4, no. 1, pp. 3–17, 2012.