IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 4, 2021, accepted January 7, 2021, date of publication January 11, 2021, date of current version January 20, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3050930

Weighted Connected Vertex Cover Based Energy-Efficient Link Monitoring for Wireless Sensor Networks Towards Secure Internet of Things

ZULEYHA AKUSTA DAGDEVIREN
International Computer Institute, Ege University, 35100 Izmir, Turkey
e-mail: zuleyhaakusta@gmail.com

ABSTRACT Industry 4.0 utilizes the Internet of Things (IoT) to rise the efficiency in manufacturing and automation where wireless sensor networks (WSNs) are crucial technologies for communication layer of IoT. WSNs include hundreds of small sized sensor nodes that have the abilities of wireless transmission and environmental sensing. Wireless transmission is prone to various attacks such as data manipulation since data communication is achieved through transfer of radio packets. A countermeasure of this issue is link monitoring by deploying secure points that can physically capture and inspect radio packets. Graph theory plays a critical role to solve various problems in WSNs. Finding minimum Vertex Cover (VC) is an important NP-Hard graph theoretic problem in which the minimum set of nodes (vertices) is aimed to select in such a way that each link should be incident to at least one node from this set. VC is a significant structure for WSNs where it perfectly fits for link monitoring when nodes in VC are set as secure points (monitors). Since sensor nodes are generally battery-powered and have limited transmission range, energy-efficient multi-hop communication to the sink node is of utmost importance. In weighted connected VC (WCVC) structure, subgraph induced by monitor nodes are connected where monitors are chosen according to their weights. When weights of nodes are assigned as reciprocal of their energies, an energy-efficient virtual backbone can be formed. We propose a novel metaheuristic WCVC algorithm for link monitoring and backbone formation in WSNs modeled as undirected graphs. Our proposed algorithm integrates a genetic search with a greedy heuristic to improve WCVC solution quality and decrease the search time. To evaluate the efficiencies of greedy heuristics, we adopt three different heuristics for WCVC problem. We implement our algorithm with its counterparts and reveal that the algorithm is favorable in terms of solution quality and resource consumption.

INDEX TERMS Internet of Things, wireless sensor networks, link monitoring, vertex cover, metaheuristic algorithm.

I. INTRODUCTION

New manufacturing, automation and production process requirements brought by Industry 4.0 will boost the development of Internet of Things (IoT) which is envisioned as a network of billions of connected smart objects to increase safety, efficiency and intelligence [1]–[3]. Application, communication and physical layers are IoT architecture layers from top to bottom. Smart plants, factories and supply chain are some well-known examples located in the application layer. Mobile phones, computing terminals and data centers are some example devices belonging to the physical layer. Wireless sensor networks (WSNs) are indispensable networking technologies used in the communication layer of IoT [4].

WSNs are composed of tiny sensor motes (sensor nodes) which are embedded in the environment to sense various events and transmit the sensed data through wireless transmission making them applicable in a broad spectrum areas such as habitat monitoring, outer space exploration, target tracking, miner safety, healthcare and military surveillance [5]–[18]. Generally, motes are geographically dispersed in a sensing area to accomplish these applications and they are

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

programmed to operate in an autonomous way. A specialized sink node plays the central data collector role of the network where it achieves a gateway operation between ordinary sensor nodes and users interested in sensed data. Although, star topology and its similar variants are known as deployment alternatives to achieve data transmission through single-hop communication between the sink and ordinary nodes, they have many defects. For example, in many real world scenarios, the maximum transmission range of motes can be limited to transmit the sensed data, obstacles may be present to interfere the wireless communication, energy conservation techniques can be applied to reduce the transmission power to prolong the network lifetime. When at least one of these cases is present, engineering multi-hop communication where the sensed data is relayed through intermediate nodes, is of paramount importance.

Wireless communication is known prone to various attacks such as spoofing, jamming and eavesdropping due to its inherent properties [19]–[22]. As an example, a successful transmission between a sender and receiver node can be physically overheard by other nodes in transmission range of sender node. In this way, an adversary node can collect various packets to learn the behaviour pattern of other legitimate nodes and may inject fake packets to misinform the nodes in network. To detect and prevent these attacks, countermeasures should be taken before WSN starts to operate. Monitoring network traffic through secure points is one of the most common methods to overcome this type of attacks [23]. When every data transmission is inspected by monitor nodes, secure points in another words, adversely generated packets may be detected and precautions can be taken. Although, this countermeasure is a very effective strategy to detect and prevent this type of misbehavior, monitor nodes can be costly in terms of many parameters such as deployment time and extra hardware cost, thus minimizing the number of these nodes is of very important.

WSNs can be modeled with an undirected graph (UG) $G(V,E)$ in which V and E represent the set of nodes and communication links (edges), respectively. Vertex cover (VC) is one of the important graph theoretic structures which can be used in various domains such as race condition detection in parallel systems, camera deployment in traffic supervision, phylogenetic tree construction and finally, the link monitoring in WSNs [23], [24]. A VC set consists of nodes where for each edge (i,j) in E at least one of i and j is in VC. In another words, at least one endpoint of each edge is a node in VC set. In this manner, the nodes in VC set can be assigned as monitor nodes. For a given UG, finding the minimum cardinality VC set is an NP-Hard problem. If each node pair in VC is connected through path of only nodes in VC set, then we call this structure as connected VC (CVC). CVC provides a virtual backbone of monitor nodes where the collected data by monitor nodes can be routed to the sink node through CVC backbone. Obviously, same with VC problem, finding minimum CVC for a given graph is NP-Hard. Since sensor nodes are generally battery powered, energy conservation is very important where the transmission is the most energy consuming factor [25]. So that, maximizing the energies of CVC set is crucial to prolong the lifetime of backbone. To accomplish this issue, weighted CVC (WCVC) can be used in which weight of each node is assigned as reciprocal of its energy. The objective of minimum WCVC problem is to find a connected VC set having minimum total weight. WCVC provides both link monitoring and energy-efficient backbone formation operations for WSNs, so it is a crucial structure. In this article, to address these challenges, we propose WCVC algorithms for link monitoring in WSNs. Our contributions are listed as follows:

- As the main contribution of this article, we propose a novel hybrid genetic WCVC algorithm for link monitoring in WSNs modeled as UG. Our novel approach combines a greedy heuristic with a genetic search to decrease the weights of WCVC solutions and to reduce the time needed to search new solutions.
- We adapt three different greedy heuristics to solve WCVC problem. After extensive experiments, we find the best heuristic and use it as the selection criteria of our metaheuristic algorithm.
- We implement the proposed algorithm with its counterparts and find that our proposed algorithm outperforms its competitors in terms of WCVC weight and monitor count. We also reveal that our proposed algorithm finds optimum results in small size instances while consuming far less time than the brute force algorithm. These findings show us that our algorithm is favorable in terms of WCVC quality and resource consumption.

The rest of this article is organized as follows. In Section II, we provide a detailed survey of the related studies. Problem formulation is given in Section III. Proposed metaheuristic algorithm along with the adopted greedy heuristics are elaborated in Section IV. Section V presents extensive experimental evaluations. Conclusions are drawn in VI.

Before ending this section, finally, we summarize the notations used throughout the paper in Table 1 to clarify and ease the reading.

TABLE 1. The list of notations used in this article.

Notation	Description
G	Graph
V	Set of vertices (nodes)
E	Set of bidirectional links (edges)
n	Number of nodes
m	Number of edges
(u, v)	The edge between node u and v
$w(u)$	Weight of node u
$N(u)$	Open neighborhood of node u
$N[u]$	Closed neighborhood of node u
$N(v, color)$	Neighbors of node v in $color$

II. RELATED WORK

In this section, we investigate studies related to vertex cover and its varieties. In [26], isolation algorithm that is a heuristic

method aiming to solve minimum VC problem has been proposed. Measurements taken on widely used datasets revealed that the proposed algorithm performs well when the graph size is small. A VC algorithm has been given in [27] that is based on two-stage exchange and edge weighting methods. The first method aims to find two nodes to exchange and the second method targets to decrease edge weights. By combining these two methods, a local search approach has been proposed and tested on widely used datasets. Caskurlu *et al.* proposed a partial VC formulation considered for risk management and proved that the tackled problem is NP-hard [28]. In [29], a local search solver has been designed to enhance best-picking strategy and it has been shown that although this method has high complexity, it can be powerful to solve problems. Hong *et al.* studied VC problem with various constraints on hypergraphs and proposed a primal-dual algorithm [30]. Cheung *et al.* proposed another study targeting the same problem on hypergraphs [31]. A multi stage metaheuristic approach in which a degree-based initialization method and snowdrift game is used that aims to solve VC problem has been designed in [32]. For other similar studies, we refer the readers to [33], [34]. The algorithms mentioned so far do not provide both weighted and connected VC whereas the proposed algorithm in this article aims to construct WCVC structure.

Xu *et al.* designed a WVC solver based on a primal-dual algorithm [35]. Through performance evaluations, they showed that their proposed solver is efficient. In [36], a list-heuristic algorithm to solve WVC has been proposed. The dual formulation of WVC and its usage has been studied in [37]. Cai *et al.* designed two algorithms which improve search process to solve WVC [38]. This algorithm has been evaluated for map labeling and tested on massive graphs. In [39], k-weighted VC problem has been studied in which the weight of VC is bounded by k. Islam et. al has been investigated vertex and edge weighted VC problem in which a chemical reaction optimization approach has been proposed to tackle this problem [40]. The algorithm has been compared with other metaheuristic methods such as genetic algorithm and tabu search. In [41], authors proposed a WVC algorithm running on graphs whose maximum degrees are 3. Li *et al.* proposed an algorithm to solve WVC based on reduction rules, configuration checking and self-adaptive node removal [42]. The authors evaluated the proposed algorithm on massive graphs and real-world instances. Although the algorithms given in this paragraph produce weighted VC, they do not aim to output CVC, thus they can not be used for backbone formation.

In [43], authors designed approximation algorithms for CVC problem on 4-regular problems where this version of CVC problem is in NP-hard same with the original CVC problem. Krithika *et al.* studied the parameterized complexity cases of CVC [44]. Johnson *et al.* concerned the CVC problem for some special cases of graphs [45]. A two stage algorithm has been given in [46] where a greedy algorithm is used in construction stage and a configuration checking method is used in local search stage. In [47], CVC problem on k-regular problems for constant k values where $k \geq 4$ is studied. The algorithms listed in this paragraph construct CVC but they do not use node weights,thus they do not target to construct energy-efficient backbones.

Fan *et al.* proposed a polynomial-time approximation scheme to solve WCVC problem on UDGs such that the input graph is c-local [48]. Despite dealing with the WCVC problem, the underlying network model of this study is completely different from our paper. Besides, UDG modeled WSNs may lack modeling simple realistic problems such as the link construction between nodes when obstacles are present in the network [53]. Hence we study UG which is more realistic model in this manner. We also left WCVC studies having different constraints (such as every path consisting of k vertices has one monitor node) [49], [50] that are out of our scope.

A similar and very interesting problem is p-self-protection in WSNs [51], [52]. This problem is finding the minimum set of safe nodes for protecting other nodes in WSN. The p-self-protection problem resembles our target problem in this study (WCVC problem), but it is directly related to dominating set problem. As a simple example, the network given in Figure 1 is a 1-self-protected WSN but not a VC since edges (2,3), (2,4), (3,6), (4,5) and (5,6) are not covered by any monitor node.

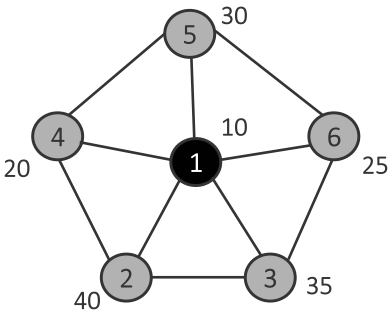


FIGURE 1. ***p*-self-protection example (node 1 is monitor).**

A table which categorizes all the reviewed works discussed in this section along their main distinctive features such as target problem, network model and algorithm type is given in Table 2.

III. PROBLEM FORMULATION

We model WSN as a node weighted UG $G(V, E, w)$ in which V represents the set of nodes, E is the set of edges and $w : V \rightarrow R^+$ is a weight function. A sample network model is given in Figure 2. In this model, each node's unique id is written inside it, each node's weight is written near it and node 0 is assigned as the sink node. Solid lines show undirected links. There is a channel between node x and node y if and only if there exists a link from node y to node x. We assume that the nodes are not mobile to preserve the network structure at least during the execution of the algorithm.

TABLE 2. Summary of the related work.

Author(s)	Target Problem	Network Model	Algorithm Type
Ugurlu [26]	Vertex Cover	Undirected Graph	Heuristic
Cai et al. [27]	Vertex Cover	Undirected Graph	Local Search
Caskurlu et al. [28]	Partial Vertex Cover	Bipartite Graph	Approximation
Ma et al. [29]	Vertex Cover	Massive Graph	Local Search
Hong and Kao [30]	Vertex Cover	Hypergraph	Approximation
Cheung et al. [31]	Vertex Cover	Hypergraph	Approximation
Wu et al. [32]	Vertex Cover	Undirected Graph	Game-based Memetic
Witt [33]	Vertex Cover	Undirected Graph	Iterated Local Search
Bazzi et al. [34]	Vertex Cover	Undirected Graph	Approximation
Xu et al. [35]	Weighted Vertex Cover	Undirected Graph	Approximation
Shimizu et al. [36]	Weighted Vertex Cover	Undirected Graph	Heuristic
Pourhassan et al. [37]	Weighted Vertex Cover	Undirected Graph	Evolutionary
Cai et al. [38]	Weighted Vertex Cover	Massive Graph	Local Search
Xu et al. [39]	k-Weighted Vertex Cover	Undirected Graph	Buss Reduction
Islam et al. [40]	Generalized Vertex Cover	Undirected Graph	Metaheuristic
Tsur [41]	Weighted Vertex Cover	Graphs with 3 max degree	Exact
Li et al. [42]	Weighted Vertex Cover	Undirected Graph	Local Search
Li et al. [43]	Connected Vertex Cover	4-regular Graph	Approximation
Krithika et al. [44]	Connected Vertex Cover	Undirected Graph	Approximation
Johnson et al. [45]	Connected Vertex Cover	(sP1+P5)-Free Graph	Exact
Zhang et al. [46]	Connected Vertex Cover	Undirected Graph	Heuristic
Li et al. [47]	Connected Vertex Cover	k-Regular Graph	Approximation
Fan et al. [48]	Weighted Connected Vertex Cover	Unit Disk Graph	Approximation
Wang et al. [49]	Weighted Connected 3-Path Vertex Cover	Unit Disk Graph	Approximation
Ran et al. [50]	Weighted Connected 3-Path Vertex Cover	Unit Disk Graph	Approximation
Wang et al. [51]	p-self-protection	Undirected Graph	Approximation
Mostafaei and Obaidat [52]	p-self-protection	Undirected Graph	Learning Automaton
This Paper	Weighted Connected Vertex Cover	Undirected Graph	Hybrid Genetic Algorithm

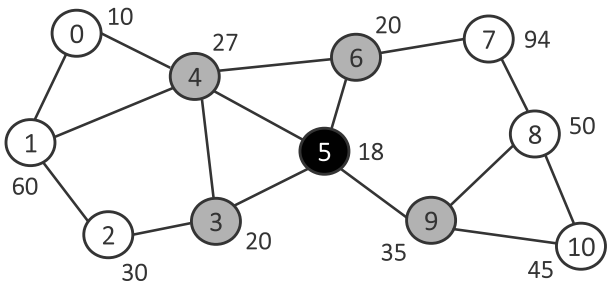


FIGURE 2. An example network.

In another words, we assume that the neighborhoods of nodes (the input communication graph) do not change when the proposed algorithm is executing. This is a fair assumption which is widely used [54] to provide consistent operation by preventing a change in the input graph during the execution of the algorithm.

Our proposed algorithm in this article is executed by the sink node. To provide central execution in the sink node, the global network graph should be constructed. Many methods can be applied to accomplish the global construction of the graph in the sink node. For example a distributed spanning tree rooted at the sink node can be constructed, then weight and neighbor list of each node can be transmitted through this spanning tree to the sink node. We assume a similar method is applied before the execution of our proposed approach.

For a given node weighted UG $G(V, E, w : V \rightarrow R^{+})$, let $V_m \in V$ is the set of monitor nodes, VC in another words, where $V_m = \{v | \forall (i,j) \in E : (i \in V_m) \vee (j \in V_m)\}$. Also let $G_m(V_m, E_m, w_m)$ be the monitor induced subgraph. Minimum WCVC problem is to find V_m such that G_m is connected and w_m is minimized.

In this article, we use $N(v)$ for neighbor set of node v where $N(v) = \{u \mid (u, v) \in E\}$. We also call $N(v)$ as the open neighborhood of node v. Closed neighborhood of node v includes open neighborhood and node v itself defined as $N[v] = \{N(v) \cup \{v\}\}$. We use BLACK, GRAY and WHITE colors to classify nodes in the graph. A BLACK node is a monitor node, a GRAY node is not a monitor node but have at least one monitor neighbor and a WHITE node is not a monitor and does not have any BLACK neighbor. To clarify colors, a sample coloring operation is shown in Figure 2. node 5 is BLACK, its neighbors (Nodes 3, 4, 6 and 9) are GRAY and the other nodes (Nodes 0, 1, 2, 7, 8 and 10) are WHITE.

IV. PROPOSED ALGORITHMS

In this section, we first define three greedy heuristics to tackle with the problem and to determine which selection strategy produces WCDS having less weight. Secondly, we give the detailed description of the proposed HGA. Lastly, we present an example application of the proposed HGA on an illustration.

A. GREEDY HEURISTICS

To deal with the WCVC problem on UG modeled WSNs, we first define greedy heuristics. The design of our heuristics are based on the ideas to solve weighted connected

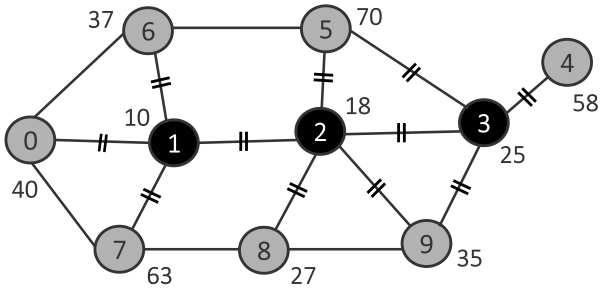


FIGURE 3. WCDS Heuristic Example.

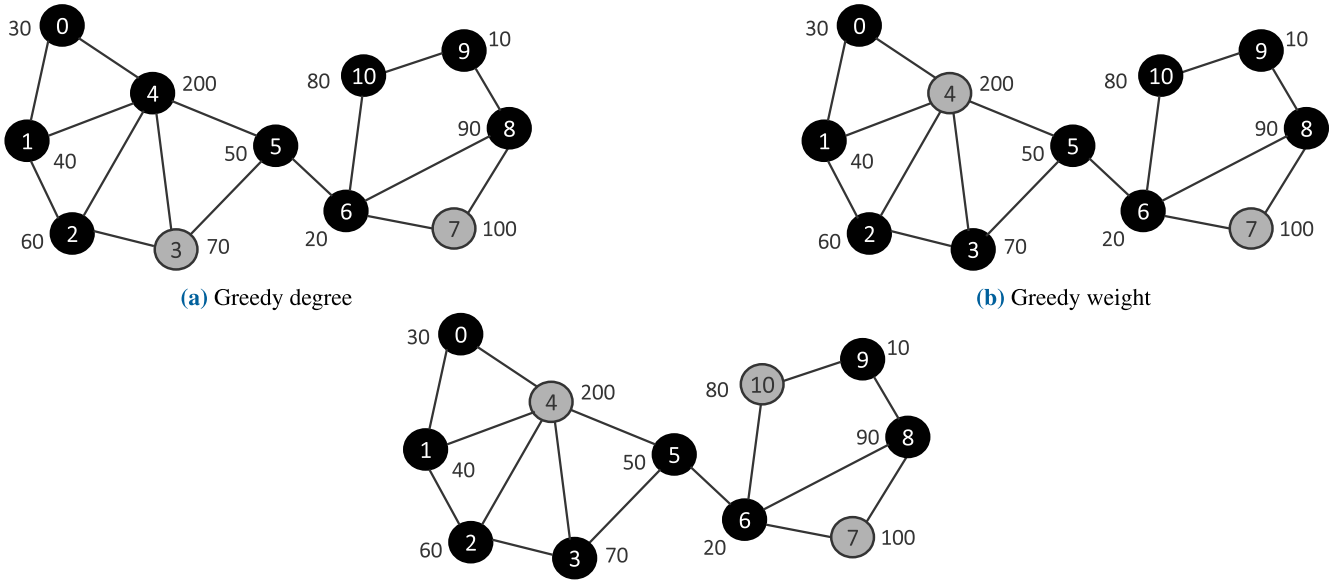
dominating set (WCDS) problem given in [53]. In each heuristic, all nodes are initially WHITE. When a node v is selected, it is colored BLACK and its WHITE neighbors are colored GRAY. When no WHITE node is left, the algorithm terminates. The heuristics differentiate each other with their node selection policy. The greedy degree (GD) heuristic chooses node having the maximum WHITE neighbor count: $\arg\max_v\{v \in V \mid N(v, WHITE)\}$. The greedy weight (GW) heuristic selects the node having the minimum weight: $\arg\min_v\{v \in V \mid w(v)\}$. The last heuristic, greedy ratio (GR), chooses the node having the minimum value calculated as: $\arg\min_v\{v \in V \mid w(v)/\sum_{u\in N(v,WHITE)} w(u)\}$.

Although the heuristics given in [53] construct WCDS, they can not guarantee to form WCVC, since two GRAY nodes can be neighbors at the end of the algorithm, so they are incapable to solve the target problem of this article. If at least one edge exists that is incident to two GRAY nodes at the end of the algorithm, then BLACK nodes do not constitute a WCVC. In another words, if there is an edge between two GRAY nodes, this edge is not covered. An example situation where the heuristics given in [53] fail to construct is depicted in Figure 3. In this figure nodes 1, 2 and 3 are selected to construct a WCDS but edges (0,6), (0,7), (5,6), (7,8) and (8,9) are not covered so WCVC can not be constructed. To overcome this problem, our proposed heuristics choose additional GRAY nodes having the smallest weight until there is no uncovered edge. In this manner, all edges will be covered by selected monitor nodes at the end of the algorithm and WCVC is constructed.

Example operations of WCVC heuristics are given in Figure 4 where execution of GD, GW and GR on sample topology are illustrated in Figures 4a, 4b and 4c, respectively. GD heuristic first chooses node 4 among all nodes (initially all nodes are WHITE) because its degree is 5 that is greater than all other nodes' degrees. After node 4 is selected, all neighbors of node 4 (nodes 0, 1, 2, 3 and 5) are colored GRAY and all edges incident to node 4 are covered. At the second step, node 5 having the maximum WHITE neighbor count among GRAY nodes is colored BLACK and its white neighbor, node 6, is colored GRAY. After that, node 6 is chosen, it is colored BLACK and its WHITE neighbors (nodes 7, 8 and 10) are colored GRAY. Following this step, node 10 is colored BLACK and node 9 is colored GRAY. From now on, there is no WHITE node left in the network, so GRAY node having the smallest weight (node 9) is chosen. Afterwards, nodes 0, 1, 2 and 8 are chosen in sequence.

GW heuristic first selects node 9 because it has the smallest weight among all nodes. It is colored BLACK and its WHITE neighbors (nodes 8 and 10) are colored GRAY. Among GRAY nodes (nodes 8 and 10), since node 10's weight is smaller than node 8, node 10 is colored BLACK and node 6 is colored GRAY. Following this, nodes 6, 5, 3, 2, 1, 0 and 8 are colored BLACK, sequentially.

When GR heuristic is applied on the example topology, node 6 is chosen at the first step, because its ratio equals to



(a) Greedy degree

(b) Greedy weight

(c) Greedy ratio

FIGURE 4. Example WCVC solutions produced after executing greedy heuristics (Initially all nodes are WHITE).

20/(20+50+80+90+100)=0.06 that is the minimum among others. In this manner, node 6 is colored BLACK and its WHITE neighbors (nodes 5, 7, 8 and 10) are colored GRAY. At the second step, node 5 which has the smallest ratio among other GRAY nodes is chosen and its WHITE neighbor (node 4) is colored GRAY. Following this, nodes 3, 2, 1, 0, 8 and 9 are chosen sequentially. Among greedy heuristics implemented in this example, GR has the best performance. To measure and evaluate the effectiveness of the heuristics in a broad and accurate manner, we implement these heuristics on a dataset including various graphs having different node counts and degrees in Section V. From extensive evaluations, we reveal that GR generally produces WCVCs having less weight than the other heuristics. Hence, we use GR as the greedy heuristic in our proposed HGA described in the following section.

B. DESCRIPTION OF THE PROPOSED ALGORITHM

We propose a hybrid steady state genetic algorithm for minimum WCVC problem. The proposed algorithm combines genetic approach and greedy heuristic to provide a feasible solution.

The detailed description of proposed algorithm is given in Algorithm 1. The algorithm starts by detecting cut vertices whose removal breaks the input topology into one or more components (Line 2). These critical nodes will definitely be in WCVC solution at the end of the algorithm to connect graph components. So that, we define them at the beginning of the algorithm. For cut vertex detection procedure, we use Tarjan's depth-first search based algorithm by taking node weighted graph G as the input [55]. After detecting cut vertices, we generate initial population having *size* members (Line 3). The first member of the topology is the solution produced by GR heuristic. The rest of the chromosome members are generated randomly. We represent the chromosomes with a bit vector of size equals to n (node count in graph). An example chromosome structure is given in Figure 5 with a bit vector of the length n where each value of 1 indicates monitors. Following that, the main loop of the algorithm that will be iterated for input I_{max} times begins (Line 4). Afterwards, a new

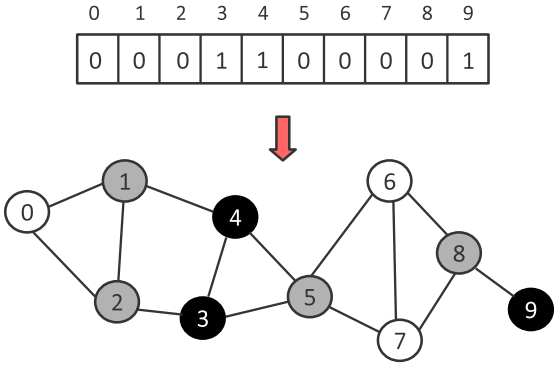


FIGURE 5. Chromosome representation.

Algorithm 1 Proposed HGA

input : pr_c – probability of crossover
pr_s – probability of selection
pr_m – probability of mutation
pr_i – probability of minimization
pr_e – probability of repair
size – population size
I_{max} – maximum iteration
$G(V, E, w)$ – node weighted graph
output: *weight* – weight of the best member

```
1  begin
2      C := DetectCutVertices(G)
3      P := GeneratePopulation(size)
4      while I_max > 0 do
5          pr_t :=GenerateFloatNumber(0, 1)
6          if pr_t < pr_c then
7              parent_1 := SelectParent(pr_s)
8              parent_2 := SelectParent(pr_s)
9              child := Crossover(parent_1,
               parent_2)
10             child := Mutate(child, pr_m)
11         end
12         else
13             child := GenerateChromosome()
14         end
15         child := RepairChromosome(pr_e,
           child)
16         child := MinimizeChromosome(C,
           pr_i, child)
17         if child ∉ P then
18             Add(child)
19             RemoveChromosome(P [size-1])
20         end
21         I_max := I_max-1
22     end
23     return Weight(P[0])
24 end
```

chromosome is generated either by genetic operations or by generating randomly (Lines 5-14) with respect to probability of crossover input (pr_c). If the genetic operations are selected (if control in Line 6 is true), a new child is constructed by first selecting two parents by applying binary tournament with respect to pr_s (Lines 7-8). A child chromosome is generated by applying a fitness based crossover operation on selected parents (Line 9) and this chromosome is mutated with respect to probability of mutation (pr_m). Fitness value of a chromosome equals to reciprocal of total monitor weight. If random chromosome generation is selected (else control in Line 12 is true), a random chromosome is generated. After child chromosome is constructed, repair and minimize operations given in Algorithms 2 and 4 are applied sequentially (Lines 15 and 16). These operations will be described in detail

Algorithm 2 RepairChromosome Algorithm

input : $G(V, E, w)$ – node weighted graph
pr_e – probability of repair
chr – chromosome
output: chr – repaired chromosome

```
1  begin
2      if CheckWCVC(G, chr) = true then
3          return chr
4      end
5      while CheckWCVC(G, chr) = true do
6          monitors
           :=GetMonitorVertices(chr)
7          pr_t :=GenerateFloatNumber(0, 1)
8          if pr_t > pr_e then
9              v := ChooseRandomVertex(G \
               monitors)
10         end
11         else
12             v := GetBestVertex(G \ monitors)
13         end
14         chr := SetNodeToMonitor(G, chr, v)
15     end
16     return chr
17 end
```

Algorithm 3 SetNodeToMonitor Algorithm

input : $G(V, E, w)$ – node weighted graph
chr – chromosome
v – vertex id
output: chr – chromosome

```
1  begin
2      chr [v] := 1
3      Color(v) := BLACK
4      Color(N(v, WHITE)) := GRAY
5      Cover(UncoveredEdges(v))
6      return chr
7  end
```

Algorithm 4 MinimizeChromosome Algorithm

input : $G(V, E, w)$ – node weighted graph
C – cut vertices
pr_i – probability of minimization
chr – chromosome
output: chr – minimized chromosome

```
1  begin
2      R :=FindRedundant(G, C, chr)
3      while SizeOf(R) ≠ 0 do
4          pr_t :=GenerateFloatNumber(0, 1)
5          if pr_t < pr_i then
6              v := GetWorstVertex(R)
7          end
8          else
9              v := GetRandomVertex(R)
10         end
11         Color(v) := GRAY
12         chr [v] := 0
13         UncoverEdges(v)
14         foreach u ∈ (N(v, WHITE) \ monitors) do
15             SetNodeToMonitor(u)
16         end
17         R := FindRedundant(G, C, chr)
18     end
19     return chr
20 end
```

in following paragraphs. When newly generated individual reaches its final form, we check the population whether it includes this chromosome (Line 17). If the new individual does not exist in the population, we add it to the population (Line 18) and remove the worst (having the lowest fitness value) chromosome member of the population. I_{max} is decremented (Line 21) and described operations (Lines between 4-22) are again executed until I_{max} equals to 0. The weight of the best member of the population is returned at the end of the algorithm (Line 23).

Repair operation given in Algorithm 2 starts by checking the chromosome that whether monitor nodes cover all edges to constitute a WCVC (Line 2). If this control is true then chromosome is immediately returned without any extra operation (Line 3). If the monitor nodes in input chromosome do not constitute a WCVC, monitor node selection procedures are applied (Lines 5-15). First, monitor nodes are extracted from the input chromosome (Line 6). Afterwards, a new monitor node is added whether by random choosing from non monitor nodes (Lines 8-10) or selecting the non monitor node having the lowest ratio by applying GR heuristic (Lines 11-13). Following monitor node selection, it is added to chromosome and its status is changed in graph (Line 14). As given in Algorithm 3, node's corresponding bit in the chromosome is set to 1, its color is set to BLACK, its WHITE neighbors' color is set to GRAY and its uncovered edges are turned to covered status. Repair operation in Algorithm 2 ends when monitor nodes construct a WCVC. At the end of this algorithm, the chromosome is returned (Line 23).

Minimize algorithm given in Algorithm 4 begins by finding redundant monitor nodes (Line 2). This operation will be described in detail in following paragraph. After redundant nodes are found, they are removed from the chromosome (Lines 3-18). A redundant monitor node is selected from the set of redundant nodes (R) either by choosing the worst member (Lines 5-7) or by random selection (Lines 8-10) with respect to minimization probability (pr_i). Following redundant monitor node selection, its color is set to GRAY, its corresponding bit is set to 0, its edges are uncovered

(Lines 11-13). Also its WHITE neighbors are assigned as monitors to the uncovered edges of the redundant monitor by Algorithm 3 (Lines 14-16). At the last step of the main loop, redundant nodes are again identified (Line 17). When the loop terminates, the minimized chromosome is returned (Line 19).

Algorithm 5 FindRedundant Algorithm

input : $G(V, E, w)$ – node weighted graph
C – cut vertices
chr – chromosome
output: R – set of redundant vertices

```
1  begin
2      BC :=DetectBlackCutVertices(G,chr)
3      monitors :=GetMonitorVertices(chr)
4      foreach v ∈ monitors do
5          if Weight(N(v, GRAY)) < Weight(v)
           ∧ v ∉ BC ∧ v ∉ C then
6              R := R ∪ v
7          end
8      end
9      return R
10 end
```

To minimize the chromosome, redundant nodes should be identified as aforementioned in previous paragraph. Finding the set of redundant monitors algorithm is given in Algorithm 5. First, cut vertices in monitor nodes induced subgraph are identified (Line 2). In another words, cut vertices are identified on the subgraph only consisting of monitor nodes (BLACK nodes) with their incident edges. We call this special set of cut vertices as black cut vertices (BC). Following this, monitors are extracted from the input chromosome (Line 3). A monitor node v is added to the set of redundant vertices (R), if node v's weight is greater than the total weight of its non monitor neighbors' (neighbors having GRAY color) and node v is not in both sets of BC and C (Lines 5 and 6). At the end of the algorithm, the list of redundant nodes are returned (Line 9).

The computational complexity of the Algorithm 5 equals to O(n+m)=O(m) (considering the network is connected) that is equal to the complexity of detecting black cut vertices algorithm in which depth-first search (DFS) based approach is used. The computational complexity of the Algorithm 3 is dominated by Lines 4 and 5 which are equal to O(n). Algorithm 4 takes O(n)×O(m)=O(nm) time considering the loop between Lines 3-18 iterates for O(n) and Lines 14-17 take O(m) time. The computational complexity of Algorithm 2 is O(m^2) since Lines 5-15 iterate for O(m) times and Line 12 executes for O(m). Generating the initial population in Line 3 of Algorithm 1 finishes in O($size \times nm$) times since $size$ number of chromosomes are generated and repaired. The loop between Lines 4-22 in Algorithm 1 executes for I_{max} times. Repairing each individual in Line 15 finishes in O($I_{max} \times nm$). The uniqueness control in Line 17 of Algorithm 1 takes O($I_{max} \times size \times n$) times. So the total computational complexity of the proposed algorithm is O((I_{max} +$size$)mn+I_{max} $size$ n). In following section, we will illustrate the operation of the proposed algorithm on a sample topology.

C. AN EXAMPLE OPERATION

In this section, we present an example operation of the proposed HGA on a topology which has 10 nodes 14 edges as given in Figure 6.

In the first step, we first generate two random chromosomes as [0110000000] and [0000001110]. Representations of the generated chromosomes on the graph are given in Figures 6a and 6b. In the first graph shown in Figure 6a, nodes 1 and 2 are monitors and 5 out of 14 edges are covered. Similarly, nodes 6, 7 and 8 are monitors in Figure 6b and 6 out of 14 edges are covered. These chromosomes are parents used in crossover operation to generate a new child given in Figure 6c. Since we use a fitness based probabilistic crossover, the outputs of this operation may vary. We accomplish our crossover operation by copying each parent's 1 bit to the child in this example. Following crossover, a mutation operation is applied on the child chromosome where node 9 changes its color from WHITE to BLACK as given in Figure 6d. After the child is constructed and the mutation is applied, no WHITE node is left in the graph. However edges (3,4), (3,5) and (4,5) are uncovered. So, we repair the topology by firstly selecting node 4 due to its lower ratio than the other candidates (nodes 3 and 5). After node 4 is colored BLACK, only edges (3,5) is left uncovered. At the second step of the repair operation, node 5 is colored BLACK to cover the last uncovered edge. When the repair operation is finished, 8 out of 10 nodes constitute a WCVC in Figure 6e. After than, we make a minimization operation on the given graph. Since the edges of BLACK nodes 6 and 9 are covered by others, they are not cut vertices and also not black cut vertices (nodes 5 and 8 are cut vertices, nodes 2, 4, 5 and 8 are black cut vertices), these nodes are redundant. So they are colored GRAY and removed from the set of monitors. Finally, monitors are assigned as nodes 2, 4, 5, 7, and 8 in Figure 6f.

V. PERFORMANCE EVALUATIONS

We implement the proposed algorithm in Java to test its performance against varying parameters. To compare the proposed HGA, three greedy algorithms with a brute force (BF) algorithm is implemented. BF algorithm checks all possible power set of nodes (2^n) and outputs the WCVC solution having the smallest weight. The algorithms are tested on dataset used in [53], [56]. This dataset is randomly generated to test graph theoretic problems on WSNs [53]. It is composed of the communication graphs having undirected edges and weighted vertices where the weight of each vertex is set as reciprocal of its energy. The dataset is divided into small, moderate (medium) and large scale WSN topologies regarding node counts in the communication graphs.

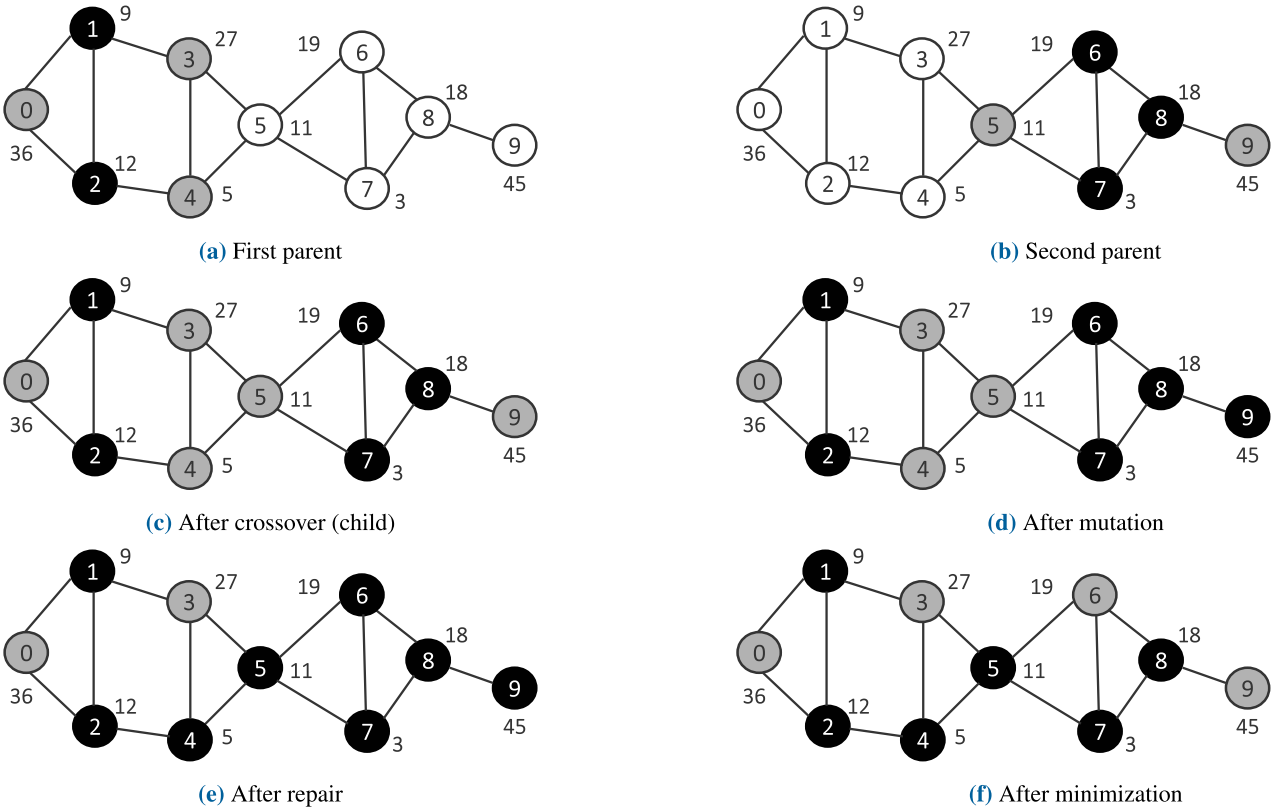


FIGURE 6. An example operation.

Node counts are 10, 15, 20 and 25 in small scale; 50, 100, 150 and 200 in moderate scale; 250, 500, 750 and 1000 in large scale WSN topologies. Besides, edge counts are varied to construct WSN topologies with different connectivity properties. Edges counts are defined as $e \times n$ for each topology where n is node count and $e \in \{2, 4, 6, 8\}$. Since BF is an exponential time algorithm, its execution time is unacceptably long for medium and large scale topologies. In this manner, we did not execute BF algorithm in these topologies.

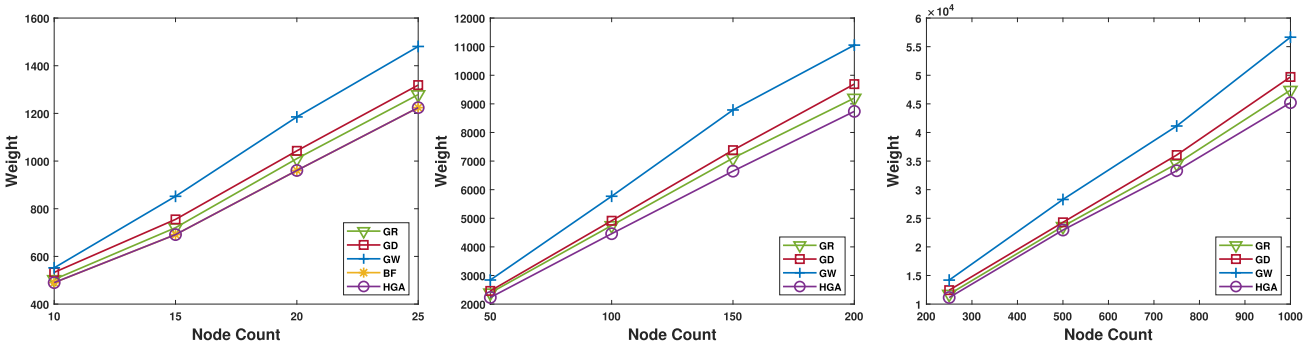
We set the parameters of the proposed HGA according to the extensive experimental evaluations given in recent studies [53], [56], [57]. In this manner, the population size ($size$), the maximum iteration count (I_{max}), crossover probability (pr_c), selection probability (pr_s), mutation probability (pr_i), repair probability (pr_e) and minimization probability (pr_i) are given as 100, 200, 0.9, 0.9, 0.8, 0.005, 0.7 and 0.6, respectively. Simulation related items such as implemented algorithms, parameters and dataset properties are summarized in Table 3.

As aforementioned, minimizing the total weight of the WCVC is of utmost importance to construct energy-efficient monitoring infrastructure. In this manner, we measure the total weight of the WCVCs produced by the implemented algorithms against varying node count and average connectivity. Total weight values of the algorithms in small, medium and large scale topologies are given

TABLE 3. Simulation parameters.

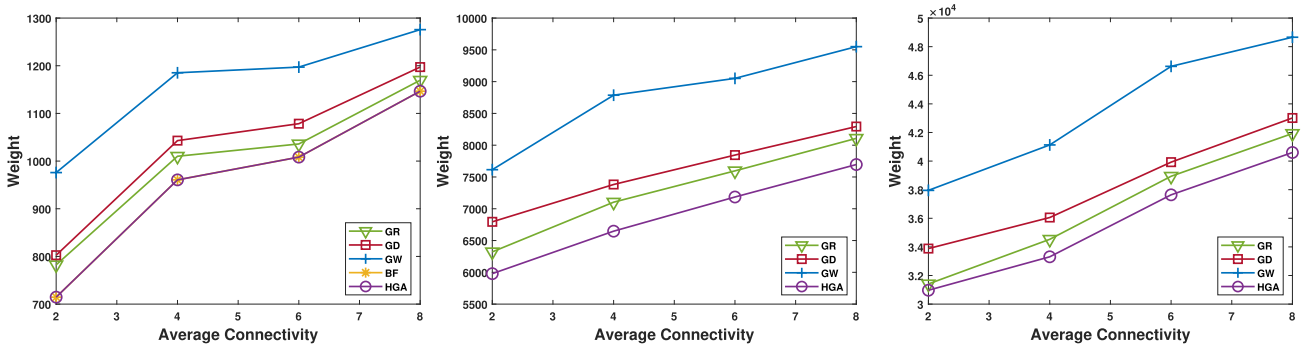
Parameter	Values
Implemented Algorithms	HGA, GR, GD, GW, BF
Node Count	{10, 15, 20, 25}, {50, 100, 150, 200}, {250, 500, 750, 1000}
Average Connectivity	2, 4, 6, 8
Population size ($size$)	100
Maximum Iteration (I_{max})	200
Crossover Probability (pr_c)	0.9
Selection Probability (pr_s)	0.8
Mutation Probability (pr_m)	0.005
Repair Probability (pr_e)	0.7
Minimization Probability (pr_i)	0.6

in Figures 7a, 7b and 7c, respectively. As node count increases, the weights of WCVCs produced by the algorithms increase linearly. GW has the worst performance among all algorithms regardless of the topology type. Although GD performs much better than GW, it has the second worst performance. GR produces the best WCVCs among implemented greedy approaches running on graphs having various node counts. For small scale topologies, the performances of the proposed HGA and BF are the same. This means that our proposed algorithm finds the optimum solution for given small scale topologies. Moreover, its time consumption is far more better than BF as we will discuss in following paragraphs. For medium and large scale topologies, again the proposed HGA outperforms the other algorithms in terms of



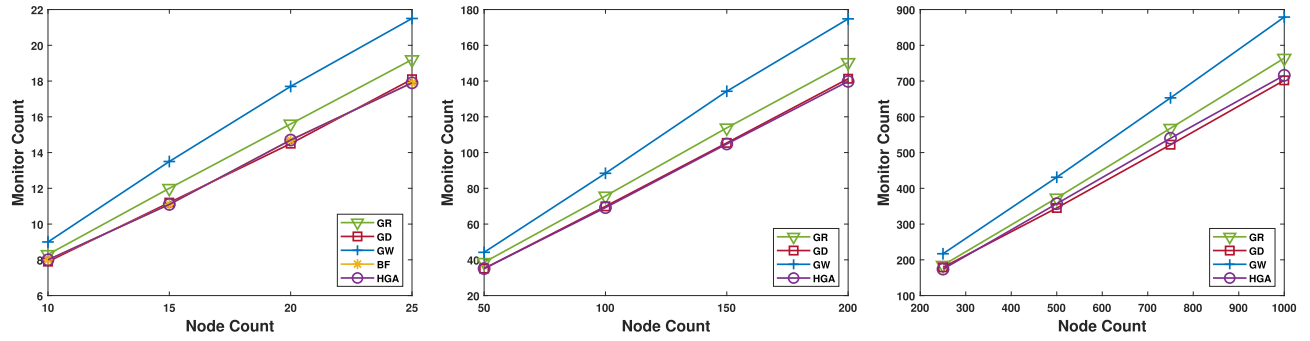
(a) Small scale topologies (b) Medium scale topologies (c) Large scale topologies

FIGURE 7. Total weight values against node count.



(a) Small scale topologies (b) Medium scale topologies (c) Large scale topologies

FIGURE 8. Total weight values against connectivity.

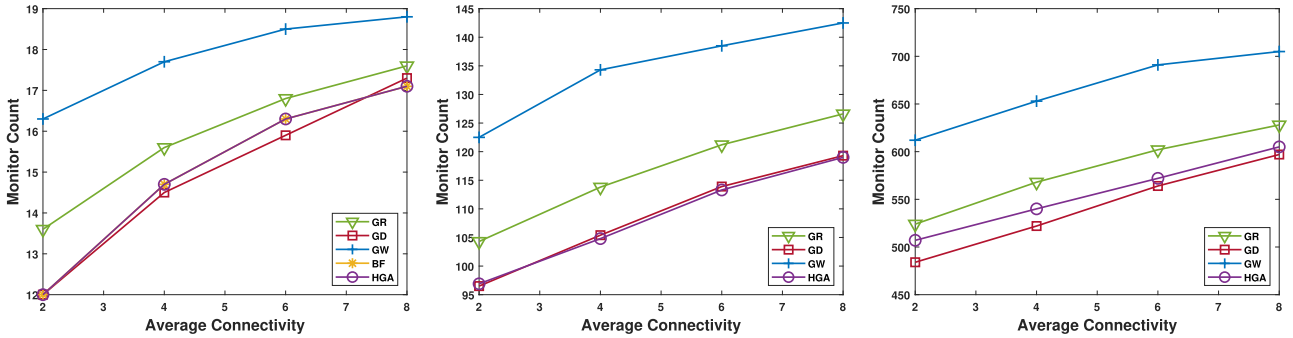


(a) Small scale topologies (b) Medium scale topologies (c) Large scale topologies

FIGURE 9. Monitor count values against node count.

WCVC weight. Total weight values of the algorithms against average network connectivity are given in Figures 8a, 8b and 8c. As network density increases, the weights of WCVCs produced by the algorithms generally increase. The reason of this increase is when the total edge count rises, more monitor nodes may be needed to cover the newly added edges. Similar with the measurements given in previous figures, the proposed HGA has the best performance among the other algorithms. For small scale graphs, again HGA and BF produce same results. The performance order of the remaining algorithms is GR, GD and GW. For medium size and large size instances, this performance order does not change and our algorithm outperforms other algorithms in terms of WCVC weight.

Although minimizing weight is the first objective of WCVC problem, solving the problem with less monitor count is also significant. With regarding this, monitor counts of the algorithms with respect to varying network parameters such as average connectivity and total node count are measured. Figures 9a, 9b and 9c show the monitor count values against the node count for small, medium and large scale topologies, respectively. Monitor counts produced by the



(a) Small scale topologies (b) Medium scale topologies (c) Large scale topologies

FIGURE 10. Monitor count values against node connectivity.

algorithms increase linearly with respect to increasing node count values. GW has the worst performance similar to the previous measurements so far. GD performs better than GR against increasing average connectivity values since GD aims to assign monitor nodes according to their degrees which produces less monitors. BF algorithm guarantees to find optimum WCVC in terms of weight, but this does not mean that BF algorithm minimizes the monitor count values. Although, for small scale graph instances, the performances of GD, BF and the proposed HGA are very close. Similarly, GD and the proposed HGA perform nearly same for the medium scale topologies. GD has the best results and the proposed HGA performs better than the other remaining algorithms in large scale instances. Monitor counts of the algorithms against average connectivity values are given in Figures 10a, 10b and 10c. As aforementioned, when network connectivity rises, counts of monitor nodes generally increase to cover newly added edges. Same with the previous results, GW has the worst and GR has the second worst performance among implemented algorithms for all datasets. For small scale instances when average connectivity value equals to 2, the proposed HGA and GD perform nearly same. For small scale instances, GD has the best results when average connectivity value equals to 4 and 6, but our algorithm outperforms GD when average connectivity value equals to 8. Our proposed HGA again produces better results than GD on medium scale instances. GD algorithm has the best measurements on large size graphs, on the other side the gap between our proposed HGA and GD algorithm decreases for increasing average connectivity. In other words, the increase rate of GD is greater than that of HGA as graph density increases.

Time consumption of the algorithms on the sink node can be an important metric if WSN executes a time critical operation such as a military surveillance application. Considering this fact, we measure the wallclock time of the algorithms against node count and average connectivity. Since GD, GR and GW execute to find an only single solution without improving it, their wallclock times are generally very small compared to population based solutions [53]. So that we compare BF and our proposed HGA against network size and density. For node count equals to 10, average connectivity values 6 and 8 are not applicable, since a graph having 10 nodes can have maximum 45 edges. Same situation is valid for node count and edge count equal to 15 and 8, respectively. For all node counts except 10, our proposed HGA is better than the BF algorithm. Moreover, for all connectivity values, when node count increases, wallclock times of our proposed HGA grow much more slower than the BF algorithm.

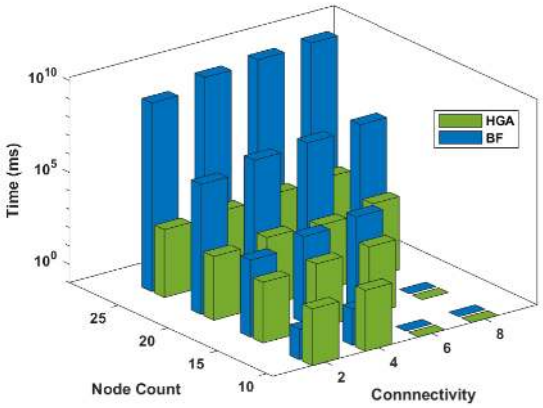


FIGURE 11. Time values of algorithms.

VI. CONCLUSION

In this article, we propose a metaheuristic to solve minimum WCVC problem to monitor links and form a virtual backbone for WSNs which are vital technologies located at the communication layer of IoT. We adopt three heuristics, namely GD, GW and GR to evaluate the effectiveness of different monitor selection strategies. Our proposed HGA is a population based metaheuristic that uses genetic search and GR heuristic to increase solution quality. Our proposed algorithm prevents the removal of cut vertices to decrease the execution time.

Measurements obtained from extensive experiments reveal that the proposed HGA outperforms other greedy algorithms in terms of total WCVC weights against varying node count and average connectivity values. Besides, although the main objective of our target problem is to minimize the weight of

WCVC to provide energy efficiency, the results show that the monitor counts produced by our algorithm is very promising for most of the cases. Moreover, the execution time of our algorithm is far more better than BF algorithm at the same producing optimum solution for small size instances against varying network sizes and densities. These findings show us that our algorithm is favorable in terms of resource efficient WCVC construction for WSNs.

In future, we plan to design distributed WCVC algorithms for providing fully autonomous execution of sensor nodes. In this algorithm each node should decide based on its energy and its neighbors' states. We also plan to model WSNs with graph convolutional networks to implement deep learning approaches. The capacitated version of the problem is open in which each monitor listens at most k links where k is a predefined parameter. This parameter will be based on the energies of the monitor nodes to provide energy-efficient link monitoring. Another potential research topic is self-stabilizing WCVC construction in which the network stabilizes in finite number of executions when transient faults are present.

REFERENCES

[1] C. Altun, B. Tavli, and H. Yanikomeroglu, "Liberalization of digital twins of IoT-enabled home appliances via blockchains and absolute ownership rights," *IEEE Commun. Mag.*, vol. 57, no. 12, pp. 65–71, Dec. 2019.

[2] M. Ayaz, M. Ammad-Uddin, Z. Sharif, A. Mansour, and E.-H.-M. Aggoune, "Internet-of-Things (IoT)-based smart agriculture: Toward making the fields talk," *IEEE Access*, vol. 7, pp. 129551–129583, 2019.

[3] M. A. Akka and R. Sokullu, "An IoT-based greenhouse monitoring system with Micaz motes," *Procedia Comput. Sci.*, vol. 113, pp. 603–608, Jan. 2017.

[4] J.-W. Lin, P. R. Chelliah, M.-C. Hsu, and J.-X. Hou, "Efficient fault-tolerant routing in IoT wireless sensor networks based on bipartite-flow graph modeling," *IEEE Access*, vol. 7, pp. 14022–14034, 2019.

[5] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Comput. Netw.*, vol. 38, no. 4, pp. 393–422, 2002.

[6] M. Tubaishat and S. Madria, "Sensor networks: An overview," *IEEE Potentials*, vol. 22, no. 2, pp. 20–23, Aug. 2003.

[7] R. Niu and P. K. Varshney, "Performance analysis of distributed detection in a random sensor field," *IEEE Trans. Signal Process.*, vol. 56, no. 1, pp. 339–349, Jan. 2008.

[8] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Comput. Netw.*, vol. 52, no. 12, pp. 2292–2330, Aug. 2008.

[9] V. Potdar, A. Sharif, and E. Chang, "Wireless sensor networks: A survey," in *Proc. Int. Conf. Adv. Inf. Netw. Appl. Workshops*, 2009, pp. 636–641.

[10] A. Hadjidj, M. Souil, A. Bouabdallah, Y. Challal, and H. Owen, "Wireless sensor networks for rehabilitation applications: Challenges and opportunities," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 1–15, Jan. 2013.

[11] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, "Wireless sensor networks: A survey on recent developments and potential synergies," *J. Supercomput.*, vol. 68, no. 1, pp. 1–48, Apr. 2014.

[12] B. Rashid and M. H. Rehmani, "Applications of wireless sensor networks for urban areas: A survey," *J. Netw. Comput. Appl.*, vol. 60, pp. 192–219, Jan. 2016.

[13] D. Ciuonzo, P. S. Rossi, and P. Willett, "Generalized rao test for decentralized detection of an uncooperative target," *IEEE Signal Process. Lett.*, vol. 24, no. 5, pp. 678–682, May 2017.

[14] S. R. J. Ramson and D. J. Moni, "Applications of wireless sensor networks—A survey," in *Proc. Int. Conf. Innov. Electr., Electron., Instrum. Media Technol. (ICEEIMT)*, 2017, pp. 325–329.

[15] R. Rucco, A. Sorriso, M. Liparoti, G. Ferraioli, P. Sorrentino, M. Ambrosanio, and F. Baselice, "Type and location of wearable sensors for monitoring falls during static and dynamic tasks in healthy elderly: A review," *Sensors*, vol. 18, no. 5, p. 1613, May 2018.

[16] O. Dagdeviren, V. K. Akram, and B. Tavli, "Design and evaluation of algorithms for energy efficient and complete determination of critical nodes for wireless sensor network reliability," *IEEE Trans. Rel.*, vol. 68, no. 1, pp. 280–290, Mar. 2019.

[17] R. E. Mohamed, A. I. Saleh, M. Abdelrazzak, and A. S. Samra, "Survey on wireless sensor network applications and energy efficient routing protocols," *Wireless Pers. Commun.*, vol. 101, no. 2, pp. 1019–1055, Jul. 2018.

[18] X. Cheng, D. Ciuonzo, and P. S. Rossi, "Multibit decentralized detection through fusing smart and dumb sensors based on rao test," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 2, pp. 1391–1405, Apr. 2020.

[19] W. Xu, K. Ma, W. Trappe, and Y. Zhang, "Jamming sensor networks: Attack and defense strategies," *IEEE Netw.*, vol. 20, no. 3, pp. 41–47, May 2006.

[20] K. Bicakci and B. Tavli, "Denial-of-service attacks and countermeasures in IEEE 802.11 wireless networks," *Comput. Standards Interfaces*, vol. 31, no. 5, pp. 931–941, Sep. 2009.

[21] D. Ciuonzo, A. Aubry, and V. Carotenuto, "Rician MIMO channel- and jamming-aware decision fusion," *IEEE Trans. Signal Process.*, vol. 65, no. 15, pp. 3866–3880, Aug. 2017.

[22] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on IoT security: Application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82721–82743, 2019.

[23] V. Kavalci, A. Ural, and O. Dagdeviren, "Distributed vertex cover algorithms for wireless sensor networks," *Int. J. Comput. Netw. Commun.*, vol. 6, no. 1, pp. 95–110, Jan. 2014.

[24] Y. Yigit, C. U. Ileri, and O. Dagdeviren, "Fault tolerance performance of self-stabilizing independent set algorithms on a covering-based problem: The case of link monitoring in WSNs," in *Proc. 5th Int. Conf. Electr. Electron. Eng. (ICEEE)*, May 2018, pp. 423–427.

[25] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*. Hoboken, NJ, USA: Wiley, 2005.

[26] O. Ugurlu, "New heuristic algorithm for unweighted minimum vertex cover," in *Proc. 4th Int. Conf. Problems Cybern. Inform. (PCI)*, 2012, pp. 1–4, doi: 10.1109/ICPCI.2012.6486444.

[27] S. Cai, K. Su, C. Luo, and A. Sattar, "NuMVC: An efficient local search algorithm for minimum vertex cover," *J. Artif. Intell. Res.*, vol. 46, pp. 687–716, Apr. 2013.

[28] B. Caskurlu, V. Mkrtchyan, O. Parekh, and K. Subramani, "On partial vertex cover and budgeted maximum coverage problems in bipartite graphs," in *Proc. IFIP TCS*, in Lecture Notes in Computer Science, vol. 8705, J. Díaz, I. Lanese, and D. Sangiorgi, Eds. Berlin, Germany: Springer, 2014, pp. 13–26.

[29] Z. Ma, Y. Fan, K. Su, C. Li, and A. Sattar, "Local search with noisy strategy for minimum vertex cover in massive graphs," in *Proc. 14th Pacific Rim Int. Conf. Trends Artif. Intell. (PRICAI)*. Cham, Switzerland: Springer, 2016, pp. 283–294.

[30] E. Hong and M.-J. Kao, "Approximation algorithm for vertex cover with multiple covering constraints," in *Proc. 29th Int. Symp. Algorithms Comput. (ISAAC)*, in Leibniz International Proceedings in Informatics, vol. 123, W.-L. Hsu, D.-T. Lee, and C.-S. Liao, Eds. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, pp. 43:1–43:11.

[31] C. W. Cheung, X. M. Goemans, and C.-W. S. Wong, "Improved algorithms for vertex cover with hard capacities on multigraphs and hypergraphs," in *Proc. SODA*, 2014, pp. 1714–1726.

[32] J. Wu, X. Shen, and K. Jiao, "Game-based memetic algorithm to the vertex cover of networks," *IEEE Trans. Cybern.*, vol. 49, no. 3, pp. 974–988, Mar. 2019.

[33] C. Witt, "Analysis of an iterated local search algorithm for vertex cover in sparse random graphs," *Theor. Comput. Sci.*, vol. 425, pp. 117–125, Mar. 2012.

[34] A. Bazzi, S. Fiorini, S. Pokutta, and O. Svensson, "No small linear program approximates vertex cover within a factor 2-e," in *Proc. IEEE 56th Annu. Symp. Found. Comput. Sci.*, Oct. 2015, pp. 1123–1142.

[35] H. Xu, T. K. S. Kumar, and S. Koenig, "A new solver for the minimum weighted vertex cover problem," in *Proc. 13th Int. Conf. Integr. Artif. Intell. Oper. Res. Techn. Constraint Program. (CPAIOR)*, 2016, pp. 392–405.

[36] S. Shimizu, K. Yamaguchi, T. Saitoh, and S. Masuda, "A fast heuristic for the minimum weight vertex cover problem," in *Proc. IEEE/ACIS 15th Int. Conf. Comput. Inf. Sci. (ICIS)*, Jun. 2016, pp. 1–5.

[37] M. Pourhassan, T. Friedrich, and F. Neumann, "On the use of the dual formulation for minimum weighted vertex cover in evolutionary algorithms," in *Proc. 14th ACM/SIGEVO Conf. Found. Genet. Algorithms (FOGA)*. New York, NY, USA: Association for Computing Machinery, 2017, pp. 37–44.

[38] S. Cai, W. Hou, J. Lin, and Y. Li, "Improving local search for minimum weight vertex cover by dynamic strategies," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 1412–1418.
[39] H. Xu, X.-Z. Wu, C. Cheng, S. Koenig, and T. K. Satish Kumar, "The buss reduction for the k-weighted vertex cover problem," in *Proc. ISAIM*, 2018, pp. 1–29.
[40] M. R. Islam, I. H. Arif, and R. H. Shuvo, "Generalized vertex cover using chemical reaction optimization," *Int. J. Speech Technol.*, vol. 49, no. 7, pp. 2546–2566, Jul. 2019.
[41] D. Tsur, "Weighted vertex cover on graphs with maximum degree 3," *CoRR*, vol. abs/1810.12982, 2018. [Online]. Available: http://arxiv.org/abs/1810.12982
[42] R. Li, S. Hu, S. Cai, J. Gao, Y. Wang, and M. Yin, "NuMWVC: A novel local search for minimum weighted vertex cover problem," *J. Oper. Res. Soc.*, vol. 71, pp. 1–12, Jun. 2019.
[43] Y. Li, Z. Yang, and W. Wang, "Complexity and algorithms for the connected vertex cover problem in 4-regular graphs," *Appl. Math. Comput.*, vol. 301, pp. 107–114, May 2017.
[44] R. Krithika, D. Majumdar, and V. Raman, "Revisiting connected vertex cover: FPT algorithms and lossy kernels," *Theory Comput. Syst.*, vol. 62, no. 8, pp. 1690–1714, Jan. 2018, doi: 10.1007/s00224-017-9837-y.
[45] M. Johnson, G. Paesani, and D. Paulusma, "Connected vertex cover for (sP1+P5)-free graphs," in *Proc. 44th Int. Workshop Graph-Theoretic Concepts Comput. Sci. (WG)*, in Lecture Notes in Computer Science, vol. 11159, A. Brandstädt, E. Köhler, and K. Meer, Eds. Cham, Switzerland: Springer, Sep. 2018, pp. 279–291. [Online]. Available: http://dro.dur.ac.uk/25710/
[46] Y. Zhang, J. Wu, L. Zhang, P. Zhao, J. Zhou, and M. Yin, "An efficient heuristic algorithm for solving connected vertex cover problem," *Math. Problems Eng.*, vol. 2018, pp. 1–10, Sep. 2018.
[47] Y. Li, W. Wang, and Z. Yang, "The connected vertex cover problem in k-regular graphs," *J. Combinat. Optim.*, vol. 38, no. 2, pp. 635–645, Aug. 2019.
[48] L. Fan, Z. Zhang, and W. Wang, "PTAS for minimum weighted connected vertex cover problem with c-local condition in unit disk graphs," *J. Combinat. Optim.*, vol. 22, no. 4, pp. 663–673, Nov. 2011.
[49] L. Wang, X. Zhang, Z. Zhang, and H. Broersma, "A PTAS for the minimum weight connected vertex cover p3 problem on unit disk graphs," *Theor. Comput. Sci.*, vol. 571, pp. 58–66, Mar. 2015.
[50] Y. Ran, Z. Zhang, X. Huang, X. Li, and D.-Z. Du, "Approximation algorithms for minimum weight connected 3-path vertex cover," *Appl. Math. Comput.*, vol. 347, pp. 723–733, Apr. 2019.
[51] Y. Wang, X.-Y. Li, and Q. Zhang, "Efficient algorithms for p-self-protection problem in static wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 10, pp. 1426–1438, Oct. 2008.
[52] H. Mostafaei and M. S. Obaidat, "Learning automaton-based self-protection algorithm for wireless sensor networks," *IET Netw.*, vol. 7, no. 5, pp. 353–361, Sep. 2018.
[53] Z. A. Dagdeviren, D. Aydin, and M. Cinsdikici, "Two population-based optimization algorithms for minimum weight connected dominating set problem," *Appl. Soft Comput.*, vol. 59, pp. 644–658, Oct. 2017.
[54] M. M. Zanjireh and H. Larijani, "A survey on centralised and distributed clustering routing algorithms for WSNs," in *Proc. IEEE 81st Veh. Technol. Conf. (VTC Spring)*, May 2015, pp. 1–6.
[55] J. Hopcroft and R. Tarjan, "Algorithm 447: Efficient algorithms for graph manipulation," *Commun. ACM*, vol. 16, no. 6, pp. 372–378, Jun. 1973.
[56] H. Dahmri and S. Bouamama, "Improved NSGA-II for minimum weight minimum connected dominating set problem," in *Proc. Int. Symp. Modelling Implement. Complex Syst.*, Sep. 2020, pp. 248–261.
[57] A. Potluri and A. Singh, "Hybrid Metaheuristic algorithms for minimum weight dominating set," *Appl. Soft Comput.*, vol. 13, no. 1, pp. 76–88, Jan. 2013.

ZULEYHA AKUSTA DAGDEVIREN received the B.Sc. degree in computer engineering from the Izmir Institute of Technology, in 2010, and the M.Sc. and Ph.D. degrees in computer science from the International Computer Institute, Ege University, Izmir, Turkey, in 2012 and 2017, respectively. Her research interests include artificial intelligence, graph theory, and medical image processing. Her current research interest includes the design and implementation of metaheuristic algorithms to solve emerging problems in recent communication networks.