Journal of Network and Systems Management (2019) 27:647–687
https://doi.org/10.1007/s10922-018-9480-1

Two Meta-Heuristics Designed to Solve the Minimum Connected Dominating Set Problem for Wireless Networks Design and Management

Abdel-Rahman Hedar[1,2] · **Rashad Ismail**[3,4] · **Gamal A. El-Sayed**[1,5] · **Khalid M. Jamil Khayyat**[6]

Received: 29 July 2017 / Revised: 23 October 2018 / Accepted: 8 November 2018 /
Published online: 21 November 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract
Wireless ad hoc and sensor networks play an important role in providing flexible deployment and mobile connectivity for next generation network. Since there is no fixed physical backbone infrastructure, some of the nodes are selected to form a virtual backbone. Efficient algorithms for identifying the Minimum Connected Dominating Set (MCDS) have many practical applications in wireless sensor networks deployment and management. We propose two algorithms in this paper for solving the MCDS problem. The first algorithm called Memetic Algorithm for the MCDS problem, or MA-MCDS shortly. This is a new hybrid algorithm based on genetic algorithm in addition to local search strategies for the MCDS problem. In order to achieve fast performance, MA-MCDS algorithm uses local search and intensification procedures in addition to genetic operations. In the second algorithm, simulated annealing is used to enhance a stochastic local search with the ability to of run away from local solutions. In addition, we present a new objective function that effectively measure the quality of the solutions of our proposed algorithms. Both algorithms are tested using different benchmark test graph sets available in the literature, and shows good results in terms of solution quality.

Keywords Minimum connected dominating set · Memetic algorithm · Simulated annealing · Stochastic local search · Wireless network design

1 Introduction

Minimum Dominating Sets (MDS) are minimum subsets of the nodes whose neighborhoods cover the whole graph. Computing MDS is a classical graph theory problem that has been covered in the literature. Minimum Connected Dominating Set

✉ Abdel-Rahman Hedar
ahahmed@uqu.edu.sa; hedar@aun.edu.eg

Extended author information available on the last page of the article

المنارة للاستشارات

(MCDS) problem is a subordinate of the original MDS problem that inspired scientists and researchers in field of wireless networks. In addition, the MCDS concept is useful in wireless sensor network, especially large scale ones, in order to find the minimum-sized connected virtual backbone in those networks. Therefore, identifying important nodes which compose the skeleton of the virtual backbone network is crucial in wireless sensor networks management [1–9]. Moreover, there are several application domains for the MCDS problem. For examples, it can be used as a tool in fault management of wireless sensor networks [10], in clustering mobile ad hoc networks [1], and in general network management [2, 9, 11]. The main challenge in such applications that their original problems, MDS and MCDS, are NP-complete [12, 13].

Using meta-heuristics as artificial intelligence tools have attracted researchers in the area of optimization recently. They cad be derived by simulating natural processes or by triggering intelligent-learned routines [14]. Meta-heuristics use different techniques to avoid local minima issue. These techniques can be classified into three categories: point-to-point techniques, population-based techniques, and hybrid techniques. Simulated annealing and tabu search are examples of the first category, while genetic algorithms and scatter search are examples of the second one. Memetic algorithms are example of the third category which combine techniques from the first two categories [15].

Simulated Annealing (SA) is a meta-heuristic based on point-to-point technique that has been used effectively in combinatorial problems. Its capability to get away from local maxima entrapment represents one of the most attractive characteristics of SA. It does so by moving down-hill using a probabilistic procedure particularly in the early stages of the search. Hence, SA has been used widely in different problems [16–18]. Kirkpatrick et al. introduce a form of SA that could be used to solve hard optimization problems [19]. It is derived from neighborhood search at which trial solutions are generated incrementally in the neighborhood of the current solution. Replacing the current solution by one of recently generated trial solutions is decided by SA according to a probability depending on the difference between their objective function values. In theory, a convergence to an optimal solution needs infinite number of iterations controlled by a cooling schedule procedure [16, 20]. Practically, a proper cooling schedule is important to behave as the asymptotic convergence of the SA [21].

Genetic Algorithm (GA) is a population-based techniques has been used to solve hard combinatorial problems. It has been derived from the biological evolution at which previous solutions are used to generate successively better solutions. GA has been extended by using local search algorithm for each solution among generations, this modification to GA has been called Memetic Algorithm (MA) by Moscato in 1989 [22]. Pastorino proved that MA is able to improve convergence time, and consequently MA is more attractive than GA [23]. MA utilizes both global and local search by using GA to perform exploration while the local search method performs exploitation. This combination of global and local search inspired many researchers in global optimization field.

We propose two new algorithms for solving the MCDS problem. We emphasize the application of our two algorithm in wireless network. Specifically, the proposed

المنارة للاستشارات

algorithms deduce a skeleton for a virtual backbone network through which a complex wireless sensor network can be controlled and managed. These new algorithms presented at this paper are two new meta-heuristics called Memetic Algorithm for the MCDS problem (MA-MCDS), and Simulated Annealing for the MCDS problem (SA-MCDS). In each algorithm, we use an on/off variable representation of different solutions while searching for the MCDS, then a new objective function will be used by each algorithm to measure the quality of the solution. That objective function takes into consideration the size of the domination in addition to the connectivity at the graph and its cardinality for each solution. In addition, MA-MCDS uses both intensification search methods and GA search methodology. Local Search, Filtering Search and Elite Dominating Sets Inspiration are the intensification search methods used by MA-MCDS [24]. In order to achieve better results, solution connecting is used as well as a new mutation based on the best solution found so far and called BS-Mutation. The role of BS-mutation is to add some nodes gradually from the best obtained solutions or remove some in order to get a better comprehensive coverage and connectivity. MA-MCDS invokes a solution connecting procedure by connecting the disconnected nodes in the obtained solution.

On the other heuristic SA-MCDS, the stochastic local search (SLS) method is implemented as a first step to search for local solutions near to the given solutions. The next step is to improve the SLS method by invoking the annealing acceptance which makes it possible to escape from local solutions. Also, Node-Reduction, Node-Addition and Node-Swapping are used to improve iterated solutions.

We compare the results of our proposed meta-heuristics against some standard methods from the literature. We used several instances of the MCDS problem to test the performance and the effectiveness of our methods. The experimental results show that our proposed methods generally outperform the ones tested against. Moreover, SA-MCDS shows a superior performance in obtaining better solutions within cheaper computational costs.

This paper is organized as follows. In the next section, we briefly give an overview to the MCDS problem. A description of the related research works on the MCDS problem is presented in Sect. 3. Section 4 presents the design of a new objective function for the MCDS problem. In Sects. 5 and 6 , we highlight the main components of both the MA-MCDS and SA-MCDS methods and present their formal algorithms. The details of the proposed methods implementation and their initial and control parameters setting are illustrated in Sect. 7. The numerical experiments with the proposed methods and their comparisons against some benchmark methods are presented in Sect. 8. Finally, the conclusion makes up Sect. 9.

2 Minimum Connected Dominating Set Problem

This section introduces the MCDS problem and explains the system model. Let $G = (V, E)$ be a simple undirected graph where $V(G)$ is a set of nodes of the graph, and $E(G)$ is a set of edges connecting the nodes of the graph. There is a path P of alternating consecutive sequence of nodes and edges that connects nodes $u, v \in V(G)$ provided that no node is repeated in the path from u to v. The length of the path P is calculated

المنارة للاستشارات

by the number of edges in that path. Nodes u and v are said to be connected if there is a path between u and v. Hence, the condition for a graph G to be connected is that every pair of nodes is connected. In a disconnected graph G, a connected component is a subgraph that is connected and cannot be part of any bigger subgraph. A graph is said to be itself connected if it has exactly one connected component which is the whole graph.

A dominating set in a graph is a subset of nodes $D \subseteq V$ such that for all $u \in V - D$ there exists a $v \in D$ for which $\{u, v\} \in E$. Consequently, each node is either a member of the dominating set or an adjacent to a node that is a member to the dominating set. Thus, this node u is said to be covered. The MDS problem is to find a dominating set in a graph with minimum cardinality. The cardinality of an MDS in a graph G is called the domination number of G and written as $\gamma(G)$.

The MCDS is another important class of domination problems that has several applications in networks. In a connected graph G, there can be a connected dominating set that is a subgraph of G. Among all possible connected dominating sets of G, the one with the minimum cardinality is called the MCDS. Hence, the MCDS problem is to find in a graph G a MCDS. The cardinality of a MCDS of graph G is called the connected domination number of G and is written as $\gamma_c(G)$.

As it is known from the literature finding a MCDS for a given graph G is not an easy task because it is an NP-complete combinatorial problem [12, 13]. Therefore, the MCDS problem is a hard combinatorial problem and cannot be solved exactly in a polynomial time. Consequently, finding an efficient solution for the MCDS problem is always one of the major areas of research in the graph theory. Recently, the MCDS problem gain more attention as it is promising in connected facility locations and has many applications in wireless networks [2, 3, 5, 6, 25–27]. It has been studied intensively in computer science and operational research [28–31].

3 Related Work

In this section, we review the state-of-the art of theory and applications of constructing the MCDS problem. The Connected Dominating Set (CDS) in a network corresponding graph can form a skeleton of a virtual backbone of this network. Actually, the domination property of this dominating set ensures that every node is either in the set or adjacent to (some node in) the set. Therefore, the network connectivity property guarantees that any two nodes can message each other via a series of adjacent nodes in the set. There are many algorithms in the literature to identify important nodes (virtual backbone nodes) form a connected dominating set of the wireless ad hoc networks [1–9, 29, 32, 33]. Beside those methods, there are several approximate and heuristic methods that attempts to solve the considered problem in generic ways for its general formulations.

3.1 Virtual Backbone of Wireless Networks

Wireless networks such as sensor network and ad-hoc network consist of several wireless nodes. In general, Wireless sensor networks are deployed in many fields,

المنارة للاستشارات

for example, biological, medical, military, environment monitoring and protection, traffic and crowd management. its physical characteristics resulted in some limitations like limited storage capacity, processing speed, communication bandwidth, short transmission range and typically powered by batteries [34]. Recharging these batteries is not easy because sensors are usually deployed in awkward locations [35]. The absence of physical backbone network makes it difficult to control such networks. Hence, virtual backbone network can be a feasible solution. Basically, virtual backbone is a set of nodes that looks like a skeleton connecting the entire network together. A message can be sent from any regular node to a destination node by passing a message to a neighboring node that is a member of connected set. The major advantage of the virtual backbone is network routing and management as it limits the search space to the set of backbone nodes.

Messages exchange among sensor nodes depend on the architecture of the constructed virtual backbone. Virtual backbone construction can be achieved by computing a connected minimum dominating set for sensor network nodes. Among important features of a connected minimum dominating set are maintenance reduction, and improving routing time [1–3, 9].

A smaller virtual backbone consume less energy, and performs the routing more efficiently. Hence, finding the MCDS is among important performance factor in wireless network routing [2, 36, 37]. Data aggregation is another example for the important benefit of computing MCDS in efficient data transmission in wireless sensor network [26, 38]. In data aggregation, each node delay its data transmission for certain period of time to merge any data received from its neighbor within this time window with its delayed data. The drawback of data aggregation is data delayed delivery, this issue can be practically mitigated using minimum connected dominating set with a minimum average backbone path that work faster for data aggregation.

Yu et al. [8, 9] proposed a mapping algorithm for the structural controllability problem on a communication network. Their algorithm finds crucial nodes in sophisticated communication networks by identifying crucial links in the network according to the number of its close subordinates.

The multi-hop connected clustering problem for a given homogeneous wireless network has been simulated into computing a minimum d-hop CDS problem by Gao et al. [1]. They developed a distributed approximation method named Connected Sparse Clustering Scheme. The first step of their algorithm is dominator selection, then inserting connector, and finally eliminating redundancy.

A greedy approximation algorithm for computing a MCDS in multi-hop wireless networks with disparate communications ranges is presented in [39] with good approximation ratio compared to previous work. In [40], Mohanty et al. proposed a distributed three-phase greedy approximation method. At this algorithm nodes store one-hop neighborhood information to compute the next dominators. The CDS size was reduced by the demotion of some existing dominators after finding CDS. Kui et al. presented an energy-balanced connected dominating set distributed scheme that extends the network lifetime by constructing an energy-balanced connected dominating set for data collection [41]. Mohanty et al. [3] presented a centralized degree-based greedy approximation algorithm for constructing a connected dominating set in the wireless networks. CDS is constructed by selecting pseudo-dominating

المنارة للاستشارات

set (PDS), then using an improved Steiner tree construction technique to connect PDS nodes, and finally removing redundancy in dominators within CDS. As this is a centralized algorithm, it does not scale well with the large number of wireless sensor networks. Consequently, they [2] developed a distributed version of the algorithm to solve the scalability and reliability problems. However they paid the cost of achieving scalability and reliability in terms of accuracy, as their centralized algorithm is more accurate than the distributed one.

Kim et al. developed algorithms for constructing an energy-efficient CDS with limited diameter for a wireless network [38]. They presented two centralized algorithms with constant performance ratios for CDS size and CDS diameter as well. Moreover, they developed a distributed version of one of them. All three algorithms are based on building a tree search algorithm, then finding a Maximum Independent SET (MIS) and finally connecting MIS nodes to form a CDS [38]. In their first algorithm, a CDS with a small diameter is formed by constructing a BFS tree, and then by connecting the root r to all the MIS nodes in tree search level by level. Consequently, the maximum number of nodes between an MIS node at level i and its nearest MIS nodes at level $i+1$ and level $i+2$ has been determined. Next they computed an MIS of G from which they finally get a CDS of G. In their second algorithm, first an MIS is obtained, then spanning tree construction connects the MIS nodes. Moreover, their second algorithm uses root node's hop-distance information to each node. Hence, a node in the level $i+1$ is connected to another node in level i by adding at most one node. Consequently, the First algorithm has a larger performance ratio for size and a smaller one for diameter than the second one. Their final algorithm is the distributed version of the second one. In addition, there are several energy-efficient algorithms have been investigated in the literature. In [42], the authors propose a distributed algorithm to find a network dominating set using capability function which tries to utilize memory, processing power, battery power, mobility ratio and computing load. Moreover, a polynomial time algorithm which can recursively computing minimum weighted dominating sets has been proposed in [43]. That algorithm respects latency and energy consumption constraints.

3.2 Theoretical Studies

Design approximate algorithms for the MCDS problem in various graphs have attracted many theoretical researchers, see [29] and references therein.

Guha and Khuller [44] proposed two polynomial-time greedy and centralized algorithms to solve the MCDS problem for a general undirected graph. Ruan et al. [45] present a new one-step greedy approximation with logarithmic performance ratio of the maximum degree in the input graph.

Wu et al. [46] designed approximation algorithms with an improved performance ratio to solve the MCDS and maximal independent sets for unit disk graphs. In [47], an approximation algorithm is developed with an approximation ratio which is a fraction of the size compared to that of the approximation algorithm at worst over the size of the optimal solution.

المنارة للاستشارات

A generalized MCDS problem called k-hop connected dominating set is considered in [28, 48]. In this generalized problem, a node v is dominated by another node u if the distance between v and u is at most k. Coelho et al. [28] present an approximation algorithm for this problem for weighted and unweighted graphs.

3.3 Heuristic Studies

Developing heuristics to construct MCDS has been the focus of many researchers on graph theory and artificial intelligence for many years.

Misra et al. proposed a new heuristic named collaborative cover [26]. The heuristic assumes a connected graph dominating number is at least two, and subset of independent dominator defines optimal substructure. A partial Steiner tree is developed during the construction of the independent dominators. Steiner nodes in the formation of Steiner tree for the independent set of G is computed in post-processing step. It has been shown that collaborative cover heuristics outperform degree-based heuristics in computing independent set and Steiner tree.

Guha and Kuller [44] propose two centralized greedy heuristic algorithms for connected dominating set formation. The one node with maximum degree will become dominating node (dominator) at each step. Their first algorithm builds up the connected dominating set at one node, then restricts the searching space for the next dominator(s) to the current uncovered nodes. The connected dominating set expands until there is no uncovered nodes. In the second algorithm, all the possible dominating nodes are determined in the first stage, then intermediate nodes are selected to create a connected dominating set in the second stage. The implementations of both algorithms were provided by Das et al. [49], and they mention the maintenance of the connected dominating set if nodes have mobility. Cheng et al. [32], proposed a greedy algorithm for MCDS in unit-disk graphs. There algorithm is based on an MIS but the computed connected dominating set may not contain all the nodes in the MIS.

The algorithm proposed by Wu and Li computes a connected dominating set and then removes some redundant nodes from the CDS using two rules [50]. In phase one, each node is marked true (dominator) if it has two unconnected neighbors. Based on the first rule, a marked node can unmark itself if its neighbor set is covered by another neighboring marked node. Based on the second rule, a marked node can unmark itself if its neighborhood is covered by two other neighboring directly connected marked nodes. the combination of both rules reduce the cardinality of connected dominating set efficiently. In [7], Wan et al. gave the performance ratio of this algorithm and correct the time complexity. Raghavan et al. [4] presented a CDS algorithm called marking process and two backbone node-reducing rules and provided some experimental results compared with the algorithms in [51].

Several meta-heuristics algorithms have been developed to deals with the considered problem. Li et al. design a GRASP for connected dominating set problems [25]. Ant colony optimization algorithms for the minimum connected dominating set problem have been proposed in [52]. In [53], the authors proposed two population-based methods using hybrid GA and greedy search for the minimum weighted

المنارة للاستشارات

connected dominating set problem. Another GA-based method has been presented in [54] that utilizes the MCDS to save energy and optimize the load balanced in wireless networks.

4 Solution Representation and Evaluation

In this section, a new objective function f is presented in order to evaluate the solution quality for the MCDS. First, we describe how to represent solutions in the proposed methods.

4.1 Solution Representation

In the MA-MCDS and SA-MCDS methods, solutions are represented in binary forms. Specifically, a trial solution x represents a subset of nodes V_x ($\subseteq V$), and is coded as a 0-1 vector with dimension equal to $|V|$, where $|V|$ is the number of nodes in the graph. Then, each component x_i of x can be defined as

$$x_i = \begin{cases} 1, & \text{if } V_i \in V_x, \\ 0, & \text{otherwise,} \end{cases}$$

for all $i = 1, \ldots, |V|$.

4.2 Solution Evaluation

During the search process, generated solutions in both methods are evaluated using a specified objective function f to determine their quality. This objective function considers the number of nodes covered by a solution, and their connectivity and cardinality. The objective function can be formally defined as:

$$f(x) = \omega_1 \frac{n_x}{|V|} + \omega_2 \frac{|C_x|}{\gamma_x} + \omega_3 \frac{|V| - \gamma_x}{|V|}, \tag{1}$$

where n_x is the number of nodes covered by solution x, $|C_x|$ is the number of nodes contained in the largest maximal connected component C_x of x, and γ_x is the number of nodes contained in x. Moreover, three weights ω_1, ω_2, and ω_3 ($0 \leq \omega_1, \omega_2, \omega_3 \leq 1$, and $\omega_1 + \omega_2 + \omega_3 = 1$) are used to trade-off between the objective function components.

The objective function has three terms which can be classified as follow:

- *Coverage* The first part, $f_{Cov}(x) = n_x/|V|$, reflects the size of domination on G by x. If x represents a dominating set, then this part is equal to 1.
- *Connectivity* The second part, $f_{Con}(x) = |C_x|/\gamma_x$, reflects the connectivity between the nodes in x. If x represents a connected set, then this part is equal to 1.

- *Cardinality* The third part $f_{Card}(x) = (|V| - \gamma_x)/|V|$ distinguishes between solutions that have the same values of the first and second parts based on the number of nodes contained in each of them.

From the previously mentioned definitions of those three parts, the objective function can be reformulated as

$$f(x) = \omega_1 f_{Cov}(x) + \omega_2 f_{Con}(x) + \omega_3 f_{Card}(x). \tag{2}$$

Therefore, the considered problem is reformulated as the following maximization problem

$$\max f(x), \tag{3}$$

where x is a binary variable of size $|V|$. In addition, the considered problem can be treated as a multi-objective optimization problem. Equations (2) and (3) convert it to a single-objective optimization problem using the weighted sum method [55]. The main challenge of such problem reformulation is how to control and tune the objective weights. During designing and implementing the proposed methods, special attention has been given first to achieve the graph coverage and solution connectivity, then trying to reduce the solution cardinality.

For wireless and ad hoc networks fault tolerance, routing and deployment issues, this objective function can be extended by adding more objectives. For example, we can add new weighted terms that reflects the energy saving by reducing the solution diameters, and the fault tolerance by reducing the node degrees in solutions.

5 Memetic Algorithm for the MCDS Problem

In this section, a memetic-based method called MA-MCDS is designed to solve the considered problem. Like other genetic-based algorithms, MA-MCDS starts with a well-distributed random population of solutions or individuals. Then, the previously defined objective function is repeatedly called in order to estimate the fitness of the initial population individuals and to rank them. At each generation of the MA-MCDS method, an intermediate population of parents is selected from the current population individuals based on their fitness. Then, three genetic operators; crossover, mutation and survival selection, are applied in order to reproduce the next generation population. The MA-MCDS method invokes three more improvement operators; local search, filtering and solution connecting, in order to improve and refine the generated solutions in each generation. Finally, a final intensification mechanism is applied in order to enhance the best obtained solutions before terminating the search process. In the following subsections, we describe components and operators of the MA-MCDS method before formally presenting the algorithm.

5.1 Genetic Operations

For the main genetic operations, the MA-MCDS uses linear ranking selection [56], and the standard one-point crossover and uniform mutation [57]. Beside the

المنارة للاستشارات

standard mutation operation, MA-MCDS invokes another special type of mutation which is called the best solution (BS) mutation [58]. MA-MCDS uses *BS-mutation* to refine the best solution x^{best} by adding nodes gradually to x^{best} in order to achieve the required coverage f_{Cov} and connectivity f_{Con} in Eq. (2). Moreover, *BS-mutation* also detects and removes redundant nodes contained in x^{best} to eventually obtain a small size f_{Card} in Eq. (2). Using this type of mutation, a new mutated child can be computed through the following procedure.

Procedure 1 $BS - mutation(x^{best})$

1. Set $x^{new} = x^{best}$.
2. If $f_{Cov}(x^{best}) < 1$, go to Step 3. Otherwise, go to Step 5.
3. Randomly select a component x_i^{new} with value 0, and set it to 1.
4. If $f_{Cov}(x^{new}) > f_{Cov}(x^{best})$, then set $x^{best} = x^{new}$, and go to Step 7.
5. Randomly select a component x_i^{new} with value 1, and set it be 0.
6. If $f_{Cov}(x^{new}) > f_{Cov}(x^{best})$, then set $x^{best} = x^{new}$.
7. If $f_{Con}(x^{best}) < 1$, use Procedure 4 to increase the connectivity of x^{best}.
8. Update x^{best}, and return.

5.2 Intensification Schemes

The main features of intelligent search methods are their abilities to perform wide exploration and deep exploitation mechanism. Even if such exploration and exploitation mechanisms are well-defined, it is still challenging to apply them in appropriate time to avoid premature convergence and unnecessary search generations. MA-MCDS invokes four intensification mechanisms, *Local Search* [24, 58], *Filtering Search* [24, 58], *Solution Connecting* [58] and *Elite Inspiration* [58] in order to achieve a faster and a better performance.

5.2.1 Local Search

Local Search is an intensification mechanism that adds or deletes some nodes in order to improve a given solution x, and this process is repeated n_l times. The details of this mechanism are formally stated in Procedure 2.

Procedure 2 $LocalSearch(x)$

1. Repeat the following steps n_l times.
2. Set $\tilde{x} = x$.
3. If $f_{Cov}(\tilde{x}) = 1$, randomly select a component $\tilde{x}_i$ with value 1. The selection probability of a component is inversely proportional to the degree of the corresponding node. Set $\tilde{x}_i = 0$, and go to Step 5.

4. If $f_{Cov}(\tilde{x}) < 1$, randomly select a component $\tilde{x}_i$ with value 0. The selection probability of a component is proportional to the degree of the corresponding node. Set $\tilde{x}_i = 1$.
5. If $f(\tilde{x}) > f(x)$, set $x = \tilde{x}$, and return.

5.2.2 Filtering Search

Filtering Search is another intensification search mechanism. Filtering Search aims to refine the best solution x^{best} found so far, if exists. If x^{best} represents a dominating set, then Filtering Search filters x^{best} by eliminating some of unnecessary nodes contained in it. Therefore, this mechanism tries to reduce the cardinality of the solution represented by x^{best} without losing its coverage property. The formal description of the *Filtering Search* mechanism is given in the following procedure.

Procedure 3 *Filtering*(x^{best})

1. If $f_{Cov}(x^{best}) < 1$, return.
2. Compute the set $\Sigma = \{\sigma_1, \ldots, \sigma_{|\Sigma|}\}$ of all positions of a value one in x^{best}.
3. Set $x^{trial} = x^{best}$.
4. Repeat the following Steps 5-6 for $j = 1, \ldots, |\Sigma|$.
5. Set $x^{trial}_{\sigma_j} = 0$, and compute $f(x^{trial})$.
6. Update x^{best} to be equal to x^{trial} if $f(x^{trial}) > f(x^{best})$.

It is worthwhile to mention that the selection of the first removed node in Procedure 3 may influence the selection of the subsequent nodes to be removed. Therefore, set Σ of all positions of a value one in x^{best} is randomly ordered to give more varieties in nodes removal. As this procedure is called several times within the main designed meta-heuristics, this gives a high possibility to have different updates of x^{best}.

5.2.3 Solution Connecting

Solution connecting is another search mechanism which aims to increase the connectivity between nodes in a solution. Specifically, *Solution Connecting* mechanism tries to compose a new connected set using a minimal number of nodes existing in the best solution x^{best} if it is not connected. First, *Solution Connecting* finds a component C which is the largest maximal connected component in x^{best}. Then, all nodes remaining in other components in x^{best} are denoted as set C'. *Solution Connecting* tries to find the shortest path to connect any two nodes ζ and ζ', where $\zeta \in C$ and $\zeta \in C'$. Hence, all intermediate nodes along the shortest path ρ from ζ to ζ' are added to x^{best}. The formal description of the *Solution Connecting* mechanism is stated in Procedure 4.

المنارة للاستشارات

Procedure 4 *Solution Connecting*(x^{best})

1. Set C equal to the set of nodes involved in the largest maximal connected component in x^{best}, and set C' equal to set of the other nodes involved in other components in x^{best}.
2. If $|C'| = 0$, return.
3. Set $x^{trial} = x^{best}$.
4. Randomly select a node $\zeta \in C$, and $\zeta' \in C'$.
5. Find the shortest path $\rho(\zeta, \zeta')$ from ζ to ζ', and add all intermediate nodes in $\rho(\zeta, \zeta')$ to x^{trial}.
6. Update x^{best} to be equal to x^{trial} if $f(x^{trial}) > f(x^{best})$.

It is worthwhile to mention that achieving solution connectivity starting from the largest maximal connected component is faster than achieving it by connecting smaller size components. The latter connecting mechanism may have better graph coverage, however the objective function considers the coverage and connectivity as two different terms. Moreover, the invoked search processes try to keep and/or increase the solution coverage using different mechanisms. Another implementation issue of Procedure 4 is to what extend we need to derive some rule to select which node pairs should be evaluated first in Step 4. Actually, this procedure is called several times within the main designed meta-heuristics. Therefore, the best solution x^{best} which usually survives over iterations has many chances to be updated using different choices of connecting pairs.

5.2.4 Elite Inspiration

As a final intensification mechanism, the MA-MCDS method invokes a search procedure called *Elite Inspiration*. In order to find a MCDS, the best n_{DS} solutions found so fare are saved in a set called *Dominating Set* (*DS*). A new trial solution x^{Core} is initialized as the intersection of the n_{Core} best solutions saved in *DS*, where n_{Core} is a pre-specified number. If the cardinality of the solution represented by x^{Core} is less than that in x^{best} by at least two, then the zero position in x^{Core} which is related to the node with a maximum degree is updated to be one. This step is repeated until the number of nodes involved in x^{Core} becomes less than that in x^{best} by one, or a better MCDS is found.

Procedure 5 $[x^{Core}] = EliteInspiration(DS, n_{Core})$

1. If *DS* is empty, then return.
2. Set n_F equal to the number of nodes involved in x^{best}, and set x^{Core} equal to the intersection of the n_{Core} solutions in *DS*.
3. If $\sum_{i=1}^{|V|} x_i^{Core} < n_F - 1$, then go to Step 4. Otherwise, return.
4. If $f_{Cov}(x^{Core}) = 1$ and $f_{Con}(x^{Core}) = 1$, then return.
5. Update the zero position in x^{Core} which gives the highest fitness, and go to Step 3.

المنارة للاستشارات

5.3 MA-MCDS Algorithm

The main structure of MA-MCDS is shown in Fig. 1. MA-MCDS starts with an initial population of μ chromosomes. Each individual in the population represents a trial solution to the MCDS problem. In order to evaluate and rank chromosomes in a population, a fitness function based on the objective function f (see Eq. (1) and Eq.(2)) is implemented. Three operators must be specified to construct the complete structure of the GA procedure; selection, crossover and mutation operators beside the four intensification schemes. MA-MCDS applies *Local Search*, Procedure 2, to improve the trial solutions. Then, in each generation the population is updated through the genetic operators. Specifically, good individuals are selected based on the linear ranking selection [56] in order to be used by other operators: crossover and mutation. MA-MCDS invokes the standard one-point crossover and uniform mutation [57], as well as *Local Search* Procedure to update the current population. Whenever a new better solution x^{best} is found, MA-MCDS invokes *Filtering Search* in order to improve it. Moreover, MA-MCDS applies Procedure 4 to interconnect the disconnected nodes in x^{best} to increase the connectivity between nodes in x^{best}.

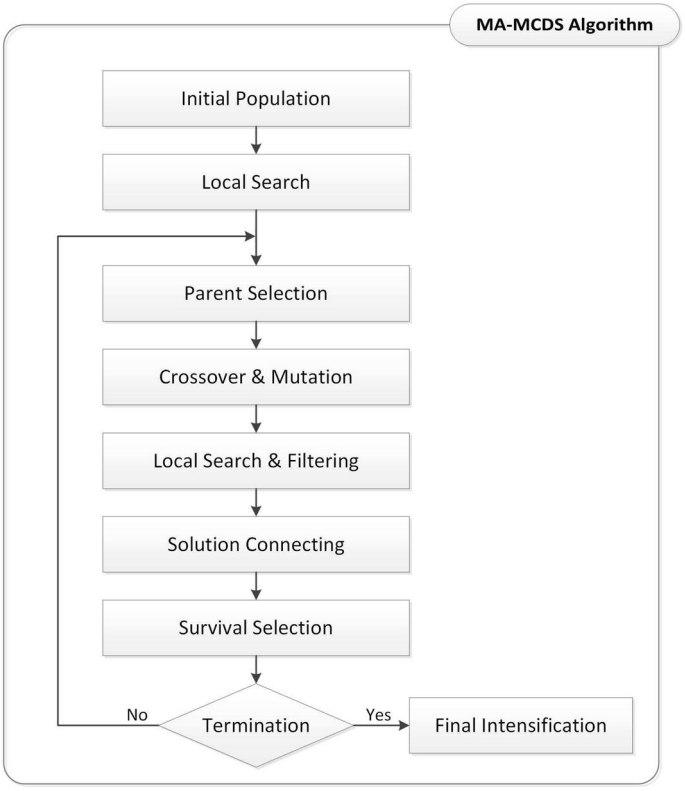

Fig. 1 MA-MCDS flowchart

المنارة للاستشارات

The search may be terminated if number of generations exceeds $g_{\max}$. Finally, the Procedure 5 is applied as a final intensification mechanism. The detailed structure of MA-MCDS is presented in the following formal algorithm.

Algorithm 6 *MA-MCDS*

1. Initialization.	Set values of μ, $g_{\max}$, n_{Core}, n_l. Set the crossover and mutation probabilities $p_c \in (0,1)$ and $p_m \in (0,1)$, respectively. Set DS to be an empty set. Generate an initial population P_0 of size μ.
2. Fitness Evaluation & Local Search.	Evaluate the fitness function of all individuals in P_0 by using the Eq. (1), and then apply *Local Search* (Procedure 2) to improve the trial solutions in P_0. Set the generation counter $t = 0$.
3. Parent Selection.	Select an intermediate population P'_t from the current population P_t using the linear ranking selection.
4. Crossover.	Apply the standard one-point crossover to chromosomes in P'_t, and update P'_t.
5. Mutation.	Apply the standard uniform mutation to chromosomes in P'_t.
6. Fitness Evaluation.	Evaluate the fitness function of all generated children in the updated P'_t.
7. BS-Mutation.	Apply the BS-mutation described in Procedure 1 on the best solution in P'_t, and update it.
8. Survival Selection.	Set $P_{t+1} = P'_t$. If the best solution in P_{t+1} is worse than the best solution in P_t, then replace the worst solution in P_{t+1} by the best solution in P'_t.
9. Local Search.	Apply *Local Search* (Procedure rm 2) starting from each individual in P_{t+1} in order to improve them, update DS and x^{best}.
10. Filtering Search.	If x^{best} represents a dominating set, then apply *Filtering Search* (Procedure 3) in order to reduce its cardinality, and update DS and x^{best}.
11. Connecting.	Apply Procedure 4 on x^{best} to increase the connectivity between nodes in x^{best}, update DS and x^{best}.

12. Stopping Condition. If termination condition is satisfied, then go to Step 13. Otherwise, set $t = t + 1$, and go to Step 3.

13. Final Intensification. Apply Procedure 5 to obtain x^{Core}. Update DS by x^{Core} if a better solution is found, and terminate.

6 Simulated Annealing for the MCDS Problem

In this section, we present the SA-MCDS method to solve the MCDS problem addressed in this research. First, we introduce a simple search procedure called the Stochastic Local Search (SLS) [59] which updates an input solution by adding, deleting and/or swapping the nodes contained in this solution in a stochastic way. Then, the SA-MCDS method globalizes the SLS procedure using the annealing acceptance and cooling schedule concepts. In the following, we give a description of the SLS procedure, then we show how it can be enhanced by the simulated annealing methodology.

6.1 Stochastic Local Search

In this section, we use the SLS method [59] for the MCDS problem. The idea of the SLS is to improve a solution by altering its nodes. First, we try to improve the solution quality by adding or deleting some nodes. If no improvement could be achieved, we exchange some of the solution nodes with other nodes. Specifically, if the current solution x represents a connected dominating set, then SLS tries to improve it by reducing its cardinality. This can be achieve by removing a node with a small degree from the node set represented by x. On the other hand, if x dose not represent a dominating set, then SLS tries to increase the solution coverage by adding a new node with a high degree. Another possibility to improve x is to exchange one node with small degree contained in x with another one with a high degree from the nodes that are not contained in x. The processes of removing, adding and replacing nodes are done in a probabilistic manner. The formal description of the SLS is shown in Procedure 7.

Procedure 7 *Stochastic Local Search*(x)

1. Set $\grave{x} = x$.
2. If $f_{Cov}(\grave{x}) = 1$, then go to Step 3. Otherwise, go to Step 4.
3. Select a component $\grave{x}_i$ with value 1 randomly with a probability inversely proportional to the degree of its corresponding node. Set $\grave{x}_i = 0$, and go to Step 6.
4. Select a component $\grave{x}_i$ with value 0 randomly with a probability proportional to the degree of its corresponding node. Set $\grave{x}_i = 1$.

المنارة للاستشارات

5. If $f_{Cov}(\grave{x}) < f_{Cov}(x)$, then select a component $\grave{x}_j$ with value 1 as in Step 3 and a component $\grave{x}_k$ with value 0 as in Step 4, and swap their values, i.e., $\grave{x}_j = 0$, $\grave{x}_k = 1$.
6. If $f(\grave{x}) \leq f(x)$, then stop. Otherwise, set $x = \grave{x}$, and go to Step 1.

6.2 SA-MCDS Algorithm

In this section, the details of the second proposed method which is called the SA-MCDS method are explained. As a point-to-point method, SA-MCDS starts with an initial solution which can be randomly chosen from the search space. The SA-MCDS search process tries to select trial solutions in the neighborhood of the current solution by adding a random displacements to the latter. Then, the objective function defined by Eq. (1) is used to evaluate the quality of both solutions. A move can be certainly accepted if the new solution has a better objective value than that of the current one. Otherwise, the move is accepted with a probability depending on the difference between the objective function values. Thereby, a worse trial solution can be accepted with the probability $p = \exp(\frac{\Delta f}{T})$; where Δf is the amount of the decrease in the objective value caused by the downhill move and T is a control parameter called the "annealing temperature". This parameter T is used to control the acceptance of inferior trial solutions. At the start of a search the temperature is initialized to be $T_{\max}$ which should be high enough to allow almost unrestricted movement around the search space. The temperature $T_{\max}$ is gradually reduced during the search constraining the acceptance of inferior trial solutions. This temperature reduction process is called the "cooling schedule," and it continues until T reaches the lower limit temperature $T_{\min}$.

SA-MCDS follows the above-mentioned framework of SA as shown in Fig. 2 with some modifications. Therefore, an initial solution x^0 is randomly selected in the search space. In each iteration k, a trial solution y is generated in the neighborhood of the current iterate solution x^k. This generation process is done through the following cases.

- If x^k represents a dominating set (i.e., $f_{Cov}(x^k) = 1$), then y is generated in the manner that reduces the cardinality of x^k as in Step 3 of SLS Procedure 7. This generation process is called "*Node-Reduction*".
- If x^k does not represent a dominating set (i.e., $f_{Cov}(x^k) < 1$), then y is generated in the manner that increases the number of nodes covered by x^k as in Step 4 of SLS Procedure 7. This generation process is called "*Node-Addition*".
- If the *Node-Addition* process fails to improve $f(x^k)$, then another process called "*Node-Swapping*" is applied as in Step 5 of SLS Procedure 7.

In these generation processes, the annealing acceptance mechanism is applied in order to accept or reject the generated trial solution y. These steps are repeated M times, where M is the epoch length [20]. At the end of each epoch, Procedure

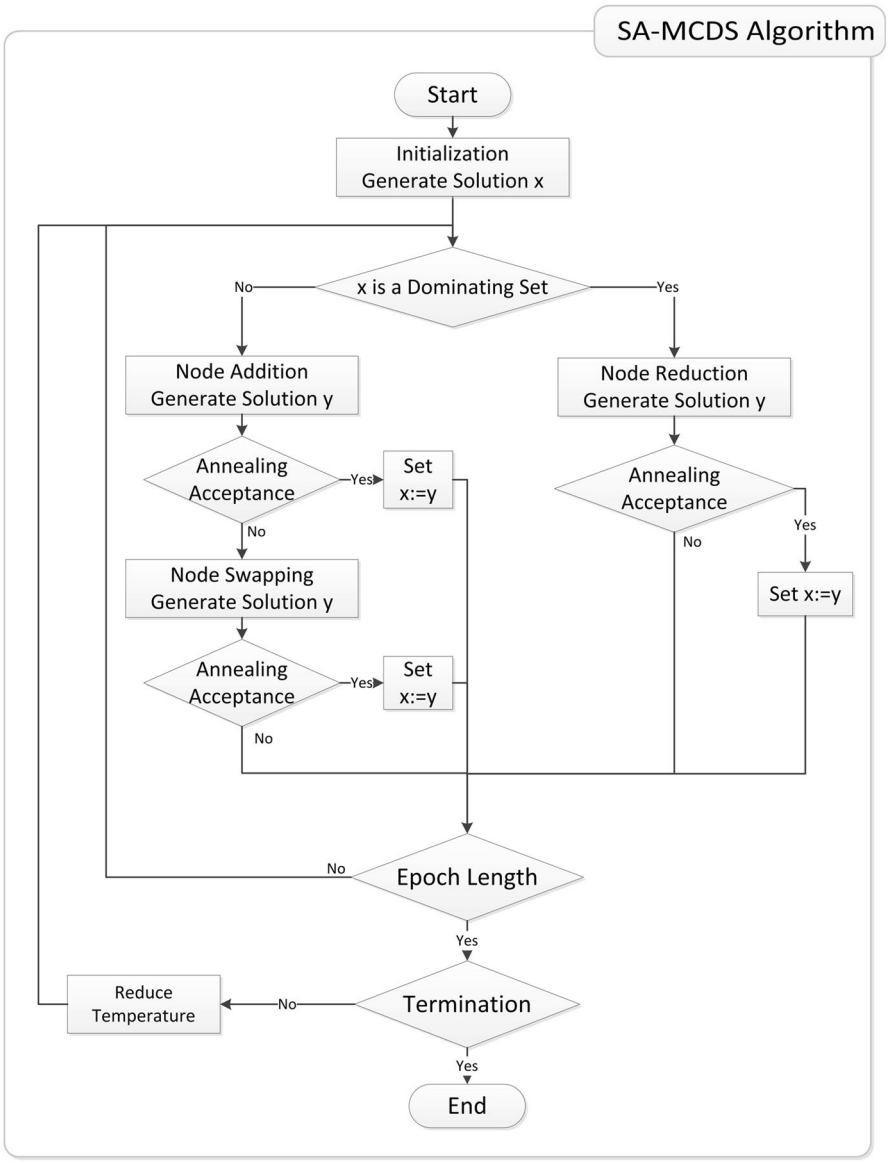

Fig. 2 SA-MCDS flowchart

4 is recalled in order to increase the connectivity of the final solution obtained in this epoch. The formal description of SA-MCDS is stated in the following algorithm.

المنارة للاستشارات

Algorithm 8 *SA-MCDS*

1. Initialization. Choose the cooling schedule parameters: initial temperature $T_{\max}$, final temperature $T_{\min}$, cooling ratio $\lambda \in (0, 1)$, and the epoch length M. Set $T = T_{\max}$, and generate an initial solution x^0. Set $x^{best} = x^0$, and $k = 0$.
2. Evaluation. Evaluate $f(x^k)$ using Eq. (1).
3. Main Loop. Repeat the following Steps (3.1–3.5) M times.

3.1. Set $y = x^k$. If $f_{Cov}(x^k) = 1$, then go to Step 3.2. Otherwise, go to Step 3.3.
3.2. Node-Reduction.

3.2.1. Randomly select a component y_i with value 1 as in Step 3 of SLS Procedure 7, set $y_i = 0$.
3.2.2. If $f(y) > f(x^k)$, set $x^{k+1} = y$, update x^{best}, set $k = k + 1$, and go to Step 4.
3.2.3. The trial solution y is accepted with probability $p = \exp(\Delta f/T)$, where $\Delta f = f(y) - f(x^k)$.
3.2.4. If y is accepted, then set $x^{k+1} = y$. Otherwise, set $x^{k+1} = x^k$.
3.2.5. Set $k = k + 1$, and go to Step 4.

3.3. Node-Addition.

3.3.1. Randomly select a component y_i with value 0 as in Step 4 of SLS Procedure 7, set $y_i = 1$.
3.3.2. If $f(y) > f(x^k)$, set $x^{k+1} = y$, update x^{best}.
3.3.3. Set $k = k + 1$, and go to Step 4.

3.4. Node-Swapping.

3.4.1. Set $z = x^k$, and randomly swap a component z_i with value 1 with a component z_j with value 0 as in Step 5 of SLS Procedure 7, i.e., $z_i = 0$, $z_j = 1$.
3.4.2. If $f(z) > f(x^k)$, then set $x^{k+1} = z$, update x^{best}, set $k = k + 1$, and go to Step 4.
3.4.3. The trial solution z is accepted with probability $p = \exp(\Delta f/T)$, where $\Delta f = f(z) - f(x^k)$.
3.4.4. If z is accepted, then set $x^{k+1} = z$. Otherwise, set $x^{k+1} = x^k$.
3.4.5. Set $k = k + 1$.

3.5. Epoch Length Condition. If the epoch length M is attained, then go to Step 4. Otherwise, apply Solution Connecting (Procedure 4) to improve x^k, update x^{best}, and go to Step 3.1.

4. Stopping Condition. If $T > T_{\min}$, then set $T = \lambda T$, and go to Step 3. Otherwise, terminate.

7 Experimental Setup

In this section, we evaluate both MA-MCDS and SA-MCDS algorithms. To compare the performance of our algorithms against several reference algorithms, we used several test graph from the literature [4, 25, 52, 60].

المنارة للاستشارات

We implemented our proposed MA-MCDS and SA-MCDS algorithms in MATLAB. Each MATLAB code was run 20 times with different initial solutions and results of 20 runs for each algorithm were averaged. We present the experimental test graphs and parameter values setting of our algorithms before results discussion.

7.1 Test Problems

Two different groups of test problems for the MCDS problem were performed. In the first group, test graphs have examples with 400 and 800 nodes and we named this group by "Test Graphs". The other group of graphs are related to network design graphs with 10–400 nodes and is denoted by "Network Graphs". The number of instance graphs that we generated are 28 and 60 ones for the test and network graphs, respectively.

7.1.1 Test Graphs

We adopted a referenced test graph construction procedure explained in [60]. To add a connectivity property on the dominating nodes, some modifications has been done to the original procedure. The modified procedure generates graphs with specific connected domination numbers and densities. The graph density p is equal to the number of edges in the graph divided by the maximum number of edges in a graph with the same number of nodes. This maximum number of edges is equal to $n(n-1)/2$ for an n nodes' undirected graph. We used a generation procedure similar to that presented in [60] to produce test graphs $G^n_{p,d}$, where d is the connected domination number. V is the set of nodes, and $|V| = n$. For each graph, we generated a number of problem instances according to density parameter (p) and connected domination number parameter (d) as shown in the 3-rd column and the 4-th column in Table 1. Initially, V is partition into d nonempty subsets $V_1, V_2, \ldots, V_d$. Nodes $x_i, y_i \in V_i$ for each $i = 1, \ldots, d$, (x_i and y_i do not need to be distinct) are chosen. Then we have the set $X = \{x_1, x_2, \ldots, x_d\}$, add edges joining each disconnected node x_i in X to another node in X in order to connect X. Finally, additional edges are added to get the required density, at the same time making sure that y_i is not connected to any node not in V_i, for $i = 1, \ldots, d$. This constraint guarantees that the connected

Table 1 Test problems

Test graphs	No. of nodes	Density (p)	Connected domination no. (d)	No. of Problem instances
$G^{400}_{0.1,d}$	400	0.1	8, 11, 14, 18, 23	5
$G^{400}_{0.3,d}$	400	0.3	3, 5, 8, 11, 14	5
$G^{400}_{0.5,d}$	400	0.5	3, 5, 8, 11	4
$G^{800}_{0.1,d}$	800	0.1	8, 11, 14, 18, 22, 26	6
$G^{800}_{0.3,d}$	800	0.3	3, 5, 9, 13	4
$G^{800}_{0.5,d}$	800	0.5	3, 6, 9, 12	4

المنارة للاستشارات

domination number of the graph is equal to d. Problem instances for each graph is shown in the last column in Table 1. To be more specific, we obtained 28 problem instances form graphs $G^{400}_{p,d}$ and $G^{800}_{p,d}$.

7.1.2 Network Graphs

We used graph generation method available in the literature to generate random graphs, see [4]. Graph nodes N are randomly deployed to a fixed area of 100×100. Next, the mobile nodes' transmission range r is set to three different values: 25, 50, 75. First we set the transmitter range r to 25, and we increase the number of mobile nodes N from 20 to 100 with a step of 20. Then we set r to 50, and increase N from 10 to 80 with a step of 10. Finally, r is set to 75, and N is increased from 10 to 60 with a step of 10. As the density of generated graphs is directly proportional to the mobile nodes' transmission range, we can control the density of generated graphs. thus, we can conclude that there is a link between two nodes if the distance between them is less than r.

Another group of network graphs is given as ad hoc network clustering instances which are described in Table 2, see [52]. In which, 8 different network instances are used; all of them occupy the same area with different number of nodes N for each instance starting from 80 up to 400 nodes. Also, we run experiments for each network size with the shown transmission ranges r. Thus, we covered different sizes of networks in our experiments.

7.2 Parameters Setting

We set the initial values of control parameters according to our numerical experiments or according to known common settings in the literature. Parameters' tuning process final values could have a positive impact on the two proposed algorithms' efficiency. Consequently, parameters' tuning process is discussed in this section.

Table 2 Ad hoc network clustering instances [52]

Network	Area ($L \times L$)	No. of nodes (N)	Range (r)
Net^1	400×400	80	60–120
Net^2	600×600	100	80–120
Net^3	700×700	200	70–120
Net^4	1000×1000	200	100–160
Net^5	1500×1500	250	130–160
Net^6	2000×2000	300	200–230
Net^7	2500×2500	350	200–230
Net^8	3000×3000	400	210–240

المنارة للاستشارات

7.2.1 MA-MCDS Parameters

Table 3 shows all parameters used in MA-MCDS associated with their values. Parameters' values are set according to our numerical experiments or according to known common settings in the literature. MA-MCDS parameters are categorized into four groups:

- The Population Parameter: μ is the population size.
- The GA operator Parameters: p_c and p_m are crossover probability and mutation probability, respectively.
- The Intensification Parameters: n_l is the number of nodes to apply *Local Search*, n_{DS} is the maximum number of the best solutions used to update *DS*, and n_{Core} is the certain number of the n_{DS} best solutions used to compute x^{Core}.
- the Termination Parameter: $g_{\max}$ is the maximum number of generations.

We used different values of these parameters to test MA-MCDS performance. Initially, the population size μ was set to 40. The preliminary numerical experiments showed that this setting was enough to obtain the best solution during search process in the most of runs. The value of crossover probability p_c is set to 0.8 and the value of mutation probability p_m is set to 0.05 improve the initial population. The numbers n_l, n_{DS} and n_{Core} used in the *Local Search* and the best connected dominating sets are set equal to 2, 10 and 3, respectively, which help MA-MCDS to improve the best connected dominating set found so far. The preliminary numerical experiments showed that these settings were reasonable to filter the elite dominating set found so far. Finally, the maximum number $g_{\max}$ of generations is set to be equal to 100. The preliminary numerical experiments showed that this setting was enough to avoid premature termination.

7.2.2 SA-MCDS Parameters

Parameters set up of SA-MCDS algorithm will be discussed in this section to wrap-up the algorithm description stated in Sect. 6. Parameters' values are set either

Table 3 The MA-MCDS parameters

Parameter	Definition	Value
μ	Population size	40
p_c	Crossover probability	0.8
p_m	Mutation probability	0.05
n_l	Number of iterations in *Local Search*	2
n_{DS}	Max number of the best solutions used to update *DS*	10
n_{Core}	The number of the best solutions used to compute x^{Core}	4
$g_{\max}$	Max number of generations	100

المنارة للاستشارات

according to our numerical experiments or according to known common settings in the literature. SA-MCDS parameters and their definitions are presented in Table 4.

SA-MCDS parameters are divided into two categories:

- *Cooling schedule* A large value is assigned to the initial temperature T_{max} to let the initial probability of accepting transition close to 1. in addition to the initial solution x^0, another solution y is generated in a neighborhood of x^0 to calculate T_{max} as shown in [17, 21]. Therefore, T_{max} is given by:

$$T_{max} = -\frac{1}{\ln(0.9)}|f(y) - f(x^0)|.$$

  The temperature is reduced with the cooling ratio λ which usually is assigned a value from the interval [0.9, 0.99]. We assigned 0.95 to λ as recommended in [21]. The epoch length M is set to 15. The epoch length M represents number of trials allowed at each temperature. Increasing its value has no significant improvement on the quality of the obtained solutions, on the other hand decreasing its value has an impact on solutions' quality.
- *Termination criterion* The termination criterion of SA-MCDS algorithm is intended to reflect the progress of this algorithm. So, it is terminated when the cooling schedule is completed. The cooling schedule is terminated when the temperature decreases to a predefined minimum temperature T_{min}. As an observation, we noticed that setting T_{min} equal to $\min(10^{-7}, 10^{-10}T_{max})$ could give a complete cooling schedule. Hence, the acceptance probability at the end is almost zero.

7.2.3 Objective Function Weights

The performance of the objective function with different values of the weights ω_1, ω_2 and ω_3 has been studied. The following settings are tested in order to choose the best setting of the objective function weights.

- Weights setting 1: $\omega_1 = 0.2, \omega_2 = 0.4, \omega_3 = 0.4$.
- Weights setting 2: $\omega_1 = 0.4, \omega_2 = 0.2, \omega_3 = 0.4$.
- Weights setting 3: $\omega_1 = 0.4, \omega_2 = 0.4, \omega_3 = 0.2$.
- Weights setting 4: $\omega_1 = 0.3, \omega_2 = 0.3, \omega_3 = 0.4$.
- Weights setting 5: $\omega_1 = 0.3, \omega_2 = 0.4, \omega_3 = 0.3$.
- Weights setting 6: $\omega_1 = 0.4, \omega_2 = 0.3, \omega_3 = 0.3$.

Table 4 The SA-MCDS parameters

Parameter	Definition	Value
M	Epoch length	15
λ	Cooling ratio	0.95
T_{max}	Initial temperature	$T_{max} = -\frac{1}{\ln(0.9)}\|f(y) - f(x^0)\|$
T_{min}	Final temperature	$\min(10^{-7}, 10^{-10}T_{max})$

المنارة للاستشارات

Different measures have been presented in Fig. 3 in order to analyze the performance of the objective function using 8 different network graphs. These measures are:

- The number of generations or iterations in which the graph coverage was achieved.
- The number of generations or iterations in which the connectivity of the best solution was achieved.
- The approximate domination numbers obtained by the proposed algorithms using the different weight settings.

Figure 3 shows that changes of the parameters values weight are not much sensitive on achieving graph coverage. Actually, both methods can obtain solutions which cover the whole networks within few generations or iterations using any of the considered weight setting. Moreover, achieving the connectivity of the best solutions was occurred later than achieving the graph coverage. All weight settings have close

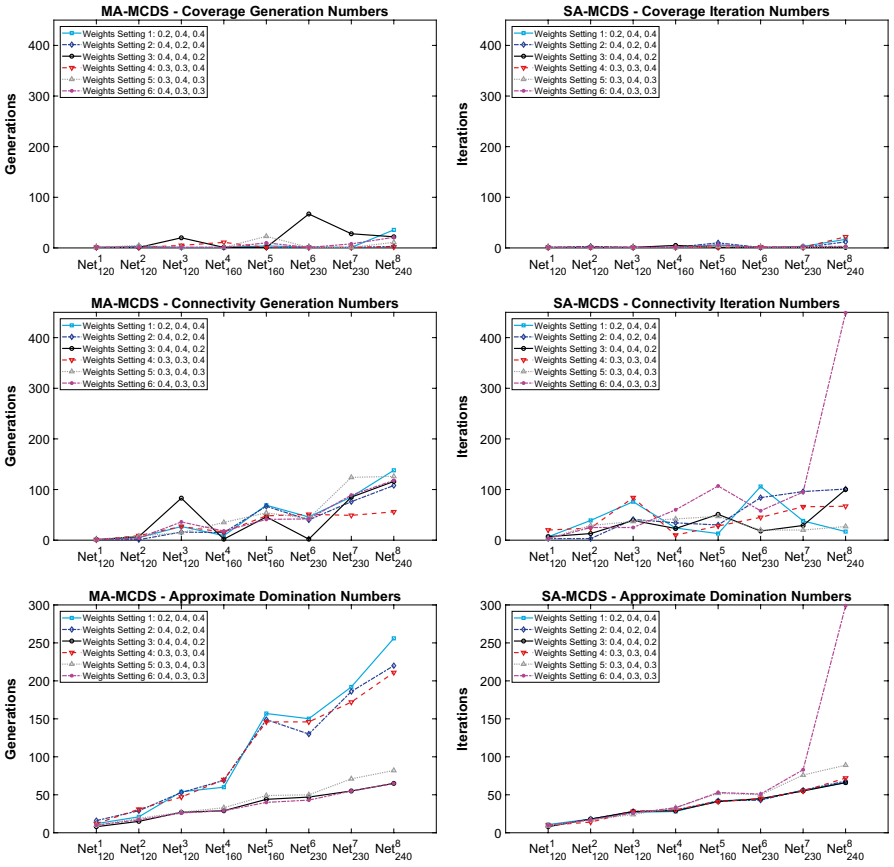

Fig. 3 The performance of the objective function with different weight values

performance regarding the achievement of the connectivity. However, the weight setting 6 gives some late connectivity results with the SA-MCDS method. Finally, the weight settings 3 and 6 could help the GA-MCDS method to obtain the best approximate domination numbers, the weight settings 3 and 4 did the same with the SA-MCDS method. Therefore, the weight setting 3 is selected for the both methods.

7.3 Procedural Analysis

The proposed MA-MCDS method contains three search procedures which are not related to the main genetic procedures. These procedures are the Filtering, Local Search and Final Intensification. In this section, we discuss the need for invoking such procedures in order to enhance the main proposed method. Figure 4 shows how these procedures affect the performance of the MA-MCDS. For two experimental graphs, three independent runs were carried out. The first run is a complete MA-MCDS method, while the second and the third ones are for MA-MCDS method without Filtering and Local Search, respectively. It is clear that Filtering and Local Search is essential for achieving better performance of MA-MCDS. Moreover, the Final Intensification could improve the obtained solutions as shown from the final beak of the MA-MCDS curve for the test graph on the right hand side of Fig. 4.

8 Numerical Results

In this section, we investigate the performance of two algorithms that we introduced in Sects. 5 and 6. We have four comparisons results, the first comparison is between MA-MCDS and the standard GA using graphs $G^{400}_{p,d}$, and the comparison results of this are shown in Table 5. The second performance comparison is between SA-MCDS and SLS using graphs $G^{400}_{p,d}$, and the results of this comparison are reported in Table 6. Then, we compare the results of MA-MCDS with those of SA-MCDS, and the results of this comparison are reported in Tables 7

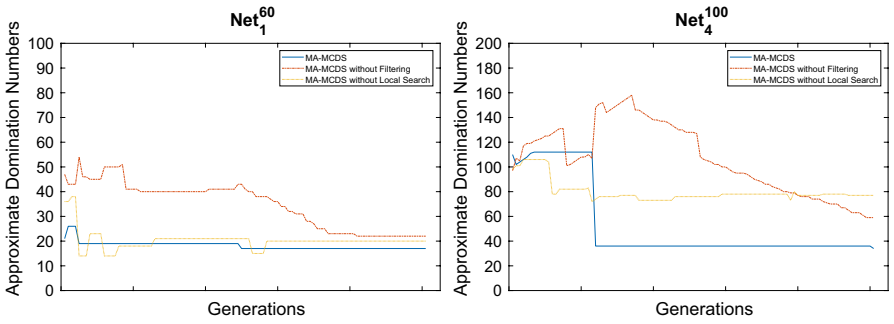

Fig. 4 The performance of the MA-MCDS procedures

Table 5 Results of running MA-MCDS and GA on $G_{p,d}^{400}$

n	p	Opt.	GA		MA-MCDS	
			Avg.	Hits(%)	Avg.	Hits(%)
400	0.1	8	137.90	0	8.05	95
400	0.1	11	151.25	0	11	100
400	0.1	14	181.15	0	14	100
400	0.1	18	204.55	0	18	100
400	0.1	23	199.30	0	23	100
400	0.3	3	52.85	0	4.25	80
400	0.3	5	98.45	0	5	100
400	0.3	8	103.95	0	8	100
400	0.3	11	150.40	0	11.5	90
400	0.3	14	195.90	0	21.05	55
400	0.5	3	26.25	0	3	100
400	0.5	5	94.75	0	5.9	90
400	0.5	8	116.05	0	8	100
400	0.5	11	163.50	0	11	100

Table 6 Results of running SA-MCDS and SLS on $G_{p,d}^{400}$

n	p	Opt.	SLS		SA-MCDS	
			Avg.	Hits(%)	Avg.	Hits(%)
400	0.1	8	13.35	65	8.35	90
400	0.1	11	14.95	45	11	100
400	0.1	14	21.15	50	14	100
400	0.1	18	21.80	50	18	100
400	0.1	23	26.75	60	23.25	95
400	0.3	3	7.75	50	3	100
400	0.3	5	6.75	55	5	100
400	0.3	8	11.20	40	8	100
400	0.3	11	19.50	50	11	100
400	0.3	14	16.75	60	14	100
400	0.5	3	4.85	45	3	100
400	0.5	5	7.10	55	5	100
400	0.5	8	11.45	45	8	100
400	0.5	11	11.95	75	11	100

and 8 . In the final comparison, we compare the results of MA-MCDS and SA-MCDS against the results of other benchmark methods presented in [4, 25, 52].

We used different quantities in making comparisons to measure the performance of each algorithm. These quantities are computed as follows.

المنارة للاستشارات

- Minimum number (Min.) This measure gives the minimum number of nodes in the best solution found in the all independent runs, this represents an approximate connected domination number of the given graph.
- Average number (Avg.) This measure gives the average number of nodes in the best solutions found in the independent runs.
- Percentage (Hits). This measure gives the percentage of the number of times the optimal solution found throughout the independent runs.
- Average number (*f*-Evals. Avg.) This measure gives the average number of objective function evaluations over the independent runs.
- Standard deviation (*f*-Evals. Std.) This measure indicates the standard deviation of the objective function evaluations over the independent runs.

We use the Wilcoxon rank-sum test [61, 62] to check the if there are statistical difference in results of our two proposed algorithms MA-MCDS, SA-MCDS and other methods. The level of significance used in the tests is 0.05. It is a pairwise test that detect significant differences in the behavior of two methods. We describe the test computations below.

We define d_i as the difference between performance scores of the two algorithms on ith out of N different results. Differences are ranked depending on their absolute values. In case of a tie average, ranks are assigned. R^+ is defined as the sum of ranks for the functions on which the first algorithm performs better than the second; and R^- is the sum of ranks for the opposite. Ranks of $d_i = 0$ are split evenly among the sums; if there is an odd number of them, one is ignored:

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i),$$

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i).$$

8.1 Performance Comparison of MA-MCDS and GA for Graphs $G^{400}_{p,d}$

The results of this comparison are reported in Table 5 and summarized in Fig. 5. All different algorithms have the same number of runs for each graph—we ran them 20 times. We considered three different density values: 0.1, 0.3, and 0.5. The connected domination number of the graph is indicated in the "Opt." column. The results show that MA-MCDS outperforms standard GA for all instances of the MCDS problem in terms average values of the obtained solutions. In addition, MA-MCDS could obtain larger percentages (Hits) of hitting the optimal solutions for all instances. GA could not outperform MA-MCDS for any instance of the MCDS problem. According to statistical results reported in Table 9, MA-MCDS demonstrates a superior performance against GA for all instances.

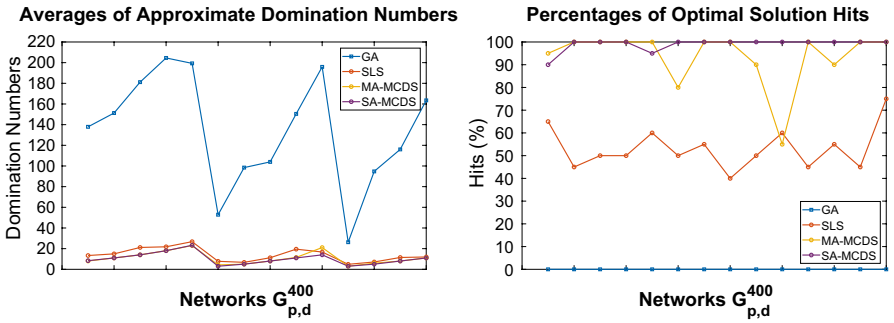

Fig. 5 Results of running GA, SLS, MA-MCDS and SA-MCDS and on $G^{400}_{p,d}$

8.2 Performance Comparison of SLS and SA-MCDS for Graphs $G^{400}_{p,d}$

In this comparison, we compared SA-MCDS with SLS method presented in [59], and results of this comparison are reported in Table 6 and summarized in Fig. 5. All methods have the same number of runs for each graph, which is 20 times. The comparison between the two methods shows that SA-MCDS outperforms SLS for all instances of the MCDS problem. More precisely, the average results of SA-MCDS for all instances are still better than those of SLS. Moreover, SA-MCDS gives considerably larger percentages of hits for all instances. This means that the annealing acceptance and the cooling schedule could help SA-MCDS to reach better solutions and escape from local solutions. According to statistical results reported in Table 9, SA-MCDS demonstrates a superior performance against SLS for all instances in terms of the success rates of finding the optimal solutions. Moreover, SA-MCDS could outperform SLS in terms of the average of the obtained solutions.

8.3 Performance Comparison of MA-MCDS and SA-MCDS

For performance evaluation of our proposed algorithms, we need to measure the size of the MCDS produced by the MA-MCDS algorithm and compare it with the one produced by the SA-MCDS algorithm in addition to the computational cost for each of them. We reported the comparison results in Tables 7 and 8, for graphs $G^{400}_{p,d}$ and $G^{800}_{p,d}$, respectively. Figures 5 and 6 summarize the approximate domination numbers and number of hits obtained by the two methods shown in Tables 7 and 8, respectively. Figure 7 also shows processing time and function evaluations of each algorithm using some test networks. This figure shows the averages of processing time and function evaluations for graphs $G^{400}_{p,d}$ and $G^{800}_{p,d}$ taken over 20 independent runs and the error bars represent the standard deviation. The two methods have similar performance regarding the processing time and number of function evaluations. It is clear from the results in Tables 7 and 8 that SA-MCDS performs better than MA-MCDS to obtain the optimum (d). Accordingly, It is

المنارة للاستشارات

Table 7 Results of running MA-MCDS and SA-MCDS on $G^{400}_{p,d}$

n	p	Opt.	MA-MCDS				SA-MCDS			
			Avg.	Hits(%)	f-Evals.		Avg.	Hits(%)	f-Evals.	
					Avg.	Std.			Avg.	Std.
400	0.1	8	8.05	95	11565.3	74.6	8.35	90	9751.1	239.8
400	0.1	11	11	100	11645.0	61.3	11	100	9495.6	33.4
400	0.1	14	14	100	11669.2	56.5	14	100	9470.7	93.1
400	0.1	18	18	100	11697.5	62.7	18	100	9478.7	75.3
400	0.1	23	23	100	11726.7	68.6	23.25	95	9433.8	51.1
400	0.3	3	4.25	80	11679.5	73.2	3	100	9712.4	134.1
400	0.3	5	5	100	11623.3	55.3	5	100	9686.9	139.5
400	0.3	8	8	100	11632.6	79.4	8	100	9775.3	126.2
400	0.3	11	11.5	90	11335.5	42.3	11	100	9443.2	68.6
400	0.3	14	21.05	55	11402.9	43.7	14	100	9477.7	112.0
400	0.5	3	3	100	11384.7	34.2	3	100	9662.5	127.4
400	0.5	5	5.9	90	11604.7	72.6	5	100	9730.0	208.3
400	0.5	8	8	100	11557.2	75.3	8	100	9585.4	109.1
400	0.5	11	11	100	11556.9	78.6	11	100	9482.9	40.5

Table 8 Results of running MA-MCDS and SA-MCDS on $G^{800}_{p,d}$

n	p	Opt.	MA-MCDS				SA-MCDS			
			Avg.	Hits(%)	f-Evals.		Avg.	Hits(%)	f-Evals.	
					Avg.	Std.			Avg.	Std.
800	0.1	8	8.05	95	11747.0	60.6	8	100	4853.2	79.9
800	0.1	11	11.05	95	11706.1	60.5	11	100	4975.1	72.4
800	0.1	14	14	100	11786.1	79.8	14	100	4858.6	43.9
800	0.1	18	18	100	11736.1	76.3	18	100	4990.5	77.6
800	0.1	22	22	100	11921.0	204.7	22	100	4828.2	38.8
800	0.1	26	26	100	11891.8	120.5	26	100	4820.2	39.3
800	0.3	3	4.7	70	11710.9	93.8	3	100	5086.3	123.0
800	0.3	5	7.7	50	12013.4	401.1	5	100	5154.4	143.4
800	0.3	9	9.9	85	11725.6	56.6	9	100	5205.5	238.4
800	0.3	13	13	100	11685.5	66.2	13	100	4889.6	13.9
800	0.5	3	6.05	40	11786.5	183.1	3	100	5047.6	174.8
800	0.5	6	7.65	55	11858.4	255.3	6	100	5015.5	140.2
800	0.5	9	12.95	35	11969.0	303.5	9	100	5117.8	198.2
800	0.5	12	12	100	11704.3	48.1	12	100	4994.6	46.6

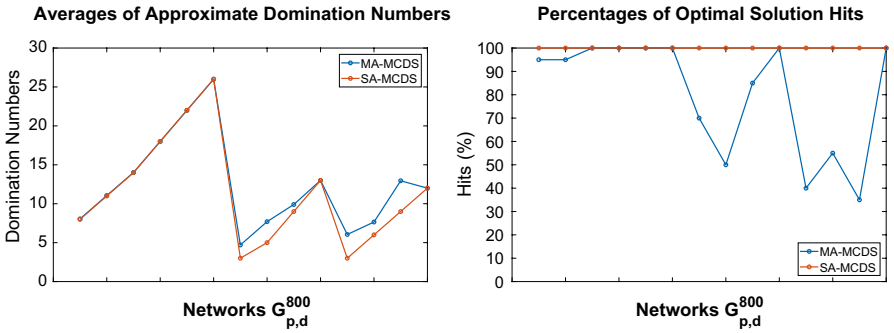

Fig. 6 Results of running MA-MCDS and SA-MCDS on $G^{800}_{p,d}$

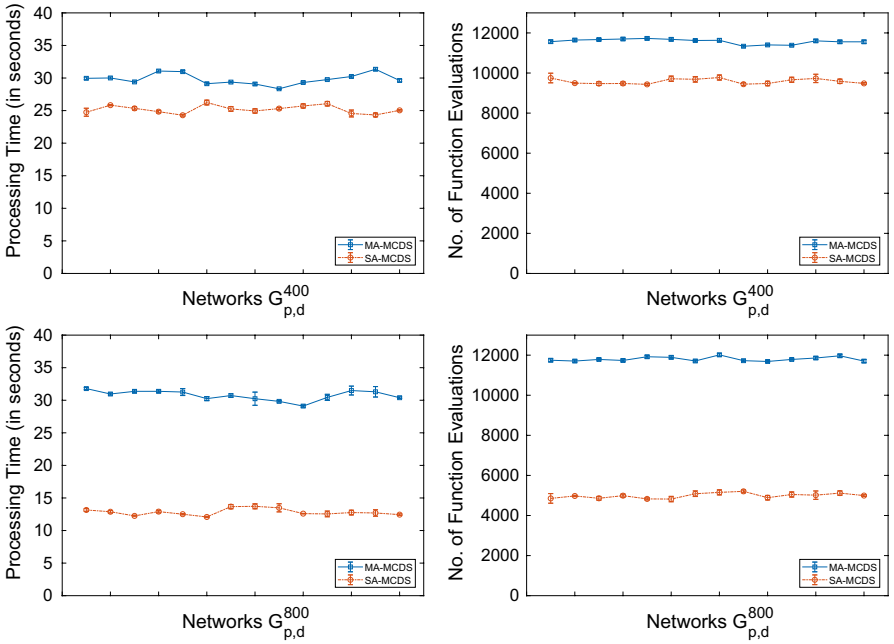

Fig. 7 The processing time and numbers of function evaluations for running MA-MCDS and SA-MCDS on $G^{400}_{p,d}$ and $G^{800}_{p,d}$

clear from the difference in average results that SA-MCDS has a better performance for most problem instances. In addition, SA-MCDS show a noticeably higher percentages of hits for many problem instances. Moreover, Fig. 7 shows that SA-MCDS could find the solutions faster than MA-MCDS. Results in Tables 7 and 8 and Fig. 7 show the difference between MA-MCDS and

المنارة للاستشارات

Table 9 Rank-sum test for comparison results in Tables 5, 6, 7 and 8

Comparing Criteria	Compared Methods		R^-	R^+	p-value	Best Method
Avg.	GA	MA-MCDS	0	105	7.425×10^{-6}	MA-MCDS
	SLS	SA-MCDS	0	105	0.1349	–
	MA-MCDS	SA-MCDS	70.5	335.5	0.6223	–
Hits(%)	GA	MA-MCDS	105	0	1.2125×10^{-6}	MA-MCDS
	SLS	SA-MCDS	105	0	2.215×10^{-6}	SA-MCDS
	MA-MCDS	SA-MCDS	337	69	1.1381×10^{-4}	SA-MCDS
f-Evals.	MA-MCDS	SA-MCDS	0	406	1.4041×10^{-10}	SA-MCDS

SA-MCDS methods in terms of the computational costs of the objective function and processing time. These results show that SA-MCDS could find the solutions faster than MA-MCDS.

According to the significance test by rank-sum test, there is no significant difference at level 0.05 between the two proposed methods in terms of the solution averages as shown in Table 9. However, SA-MCDS shows a better performance compared to MA-MCDS in terms of the f-Evals. and hits percentages. By conventional criteria, this difference in f-Evals. and percentage hits is considered to be extremely statistically significant.

Some of classical network metrics have been considered in order to analyze the performance of the proposed methods. These metrics are the diameter, average node degree, approximate domination numbers and convergence time. Figure 8 shows the average values of these metrics calculated from the obtained solutions by the proposed algorithms over 10 independent runs on Net^1, Net^2, $\cdots$, Net^8. Two types of diameters are calculated; the unweighted one in which all edges are equal in cost, and the weighted one in which the edge weight is the Euclidean distance between the nodes connected by this edge. The diameter values of the connected dominating sets obtained by the proposed methods are close to each other. However, those diameter values are slightly greater than the diameter values of the whole graph when the graph size is increased. In conclusion, the obtained connected dominating sets are located in the middle of the network graphs. For the node degree metrics, the minimum, average and maximum values of the obtained connected dominating sets are represented in Fig. 8, as well as those values of the whole network graphs. Fore each graph, those node degree values of the connected dominating sets and the whole graph are close to each other. However, those diameter values are slightly lower than the diameter values of the whole graph when the graph size is increased. This may support the quality of the obtained solutions in terms of the well-distribution of the dominating nodes over the graph. Figure 8 also shows the close performance of the two proposed methods in calculating the approximate domination numbers. Finally, the processing time of the SA-MCDS method is slightly smaller than that of the MA-MCDS method for the large size networks as reported in Fig. 8. This could be expected since the SA-MCDS method is a single-solution search method while the MA-MCDS method is a population-based search method.

المنارة للاستشارات

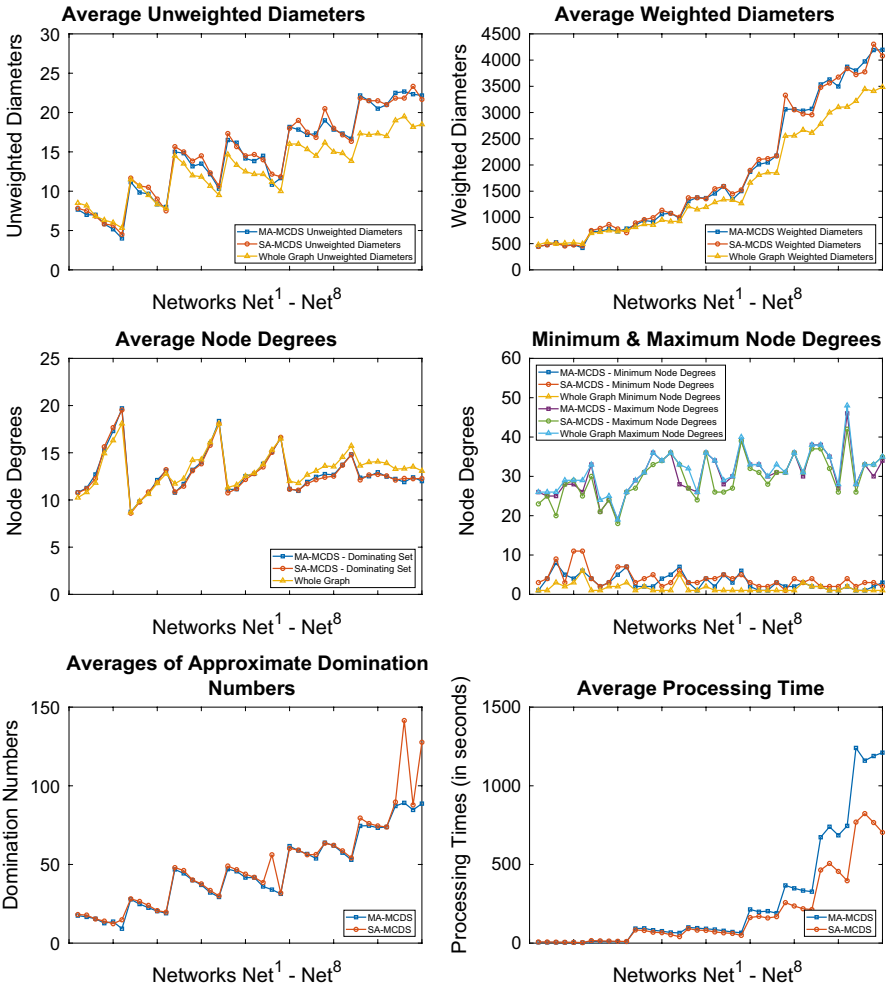

Fig. 8 Other metrics comparisons between MA-MCDS and SA-MCDS on graphs $Net^1 - Net^8$

8.4 Determining Backbone Nodes in a Static Wireless Sensors Network

In this section, we study the performance comparison of the two proposed algorithms MA-MCDS and SA-MCDS to identify backbone nodes in a static wireless sensors network. In addition, our two proposed algorithms are compared with six benchmark methods presented in [4, 25, 52] through two main experiments. In these experiments, the compared data of the benchmark methods are taken from their original references. The main reasons for doing two different experiments is that the compared performance measures and availability of the compared data are different between the compared benchmark methods. In the fist experiment, we measure the size of the connected dominating set produced by MA-MCDS

المنارة للاستشارات

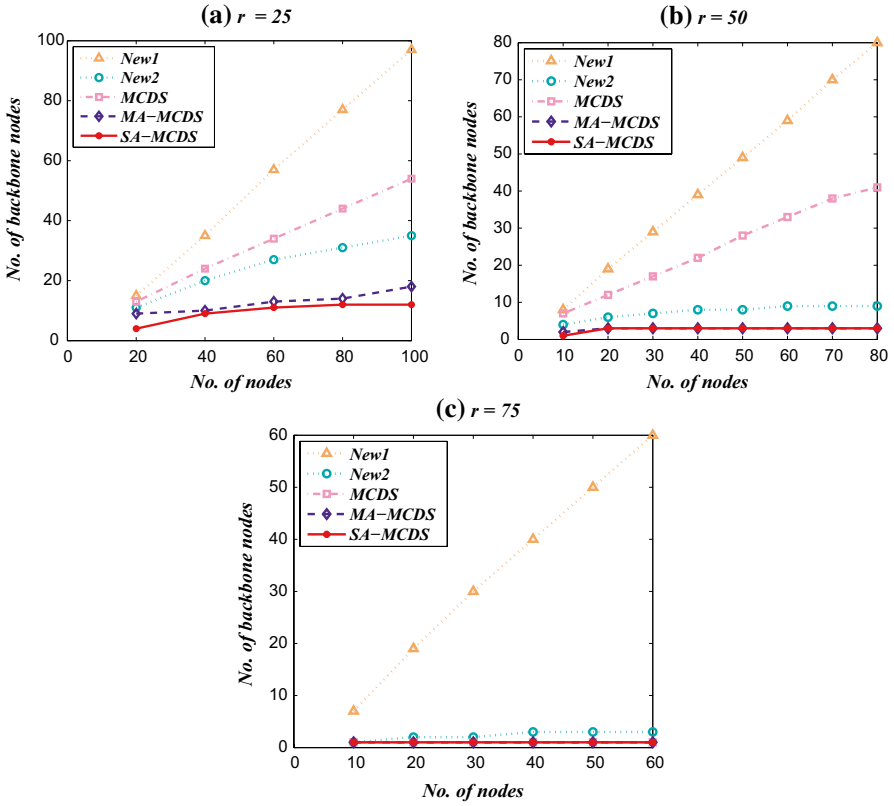

Fig. 9 Average number of backbone nodes relative to the number of nodes

and SA-MCDS, and compare them with those produced by the three methods; New1 and New2 [4] and MCDS [51]. Figure 9 shows comparison results that has been done using the following parameters.

- N: The number of nodes in the network.
- D: The number of backbone nodes (the size of the connected dominating set) in the network.
- r: The radius of nodes' transmission range.

The authors of [4] gave several experimental results comparing New1, New2 and MCDS methods in terms of size of backbone nodes. Experimental results in [4] show that New2 is better than both New1 and MCDS algorithms at relatively small values for r, for example, $r = 25, 50$. On the other hand, MCDS method outperforms New1 and New2 methods when the r value is large, for example, $r = 75$. Therefore, we paid more attention to New2 and MCDS methods.

In our experiments, we assigned three different values: 25,50,75 to the radius of the network nodes' transmission range r. For each value of r, we change the

number of nodes N from 10 to 100. For each N, MA-MCDS and SA-MCDS have the same number of runs for each graph, which is 20 times, while other two methods have 1000 times [4]. The experimental results of each algorithm are averaged. We compare all methods in terms of number of backbone nodes generated with each them. Thus, a better result is obtained when the backbone is of minimum cardinality.

Results of this comparison are demonstrated in Fig. 9 to show number of backbone nodes against number of nodes in the network for different values of radius r. It is clear that the performance of our proposed MA-MCDS and SA-MCDS methods outperform other methods, and particularly the SA-MCDS method outperforms all methods.

By Inspecting Fig. 9c we see that, when the transmission radius r is large (relative to the area), the MA-MCDS and SA-MCDS algorithms perform closely to each other. On the other hand, When the transmission radius become small, as shown in Fig. 9a, b, our two methods perform better than other methods. In Fig. 9a, we can clearly see that there is a gap between SA-MCDS and the other methods.

Additionally, experimental results shown in Figs. 10 and 11 indicate that the two proposed methods could obtain MCDS as a virtual backbone of a static wireless sensors network. It is clear that SA-MCDS gives a better backbone when $r = 25, N = 40$, however, the two methods produce the same results when $r = 50, N = 20$.

The final comparison experiment is to test our results with network instances described in Table 2. Table 10 shows the comparison results obtained by the proposed methods against the following methods:

- GRASP for connected dominating set problems [25].
- Ant colony optimization algorithms for the minimum connected dominating set problem: ACO and ACO+PCS [52].

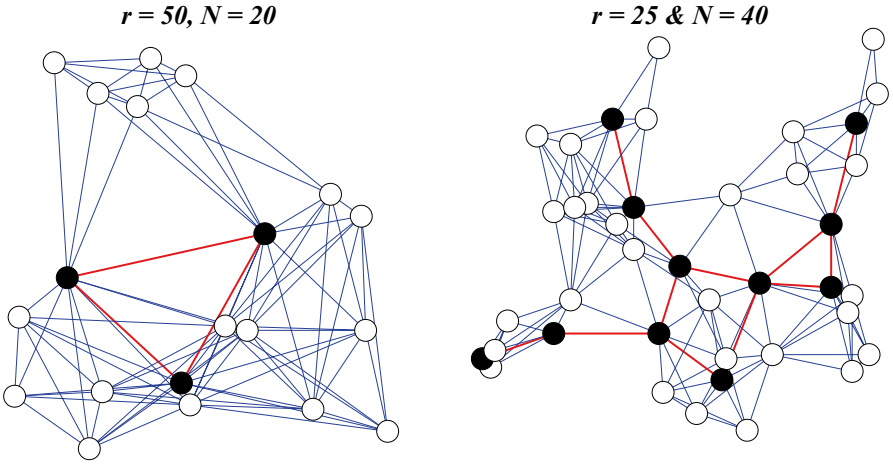

Fig. 10 Examples of obtaining backbone using MA-MCDS

المنارة للاستشارات

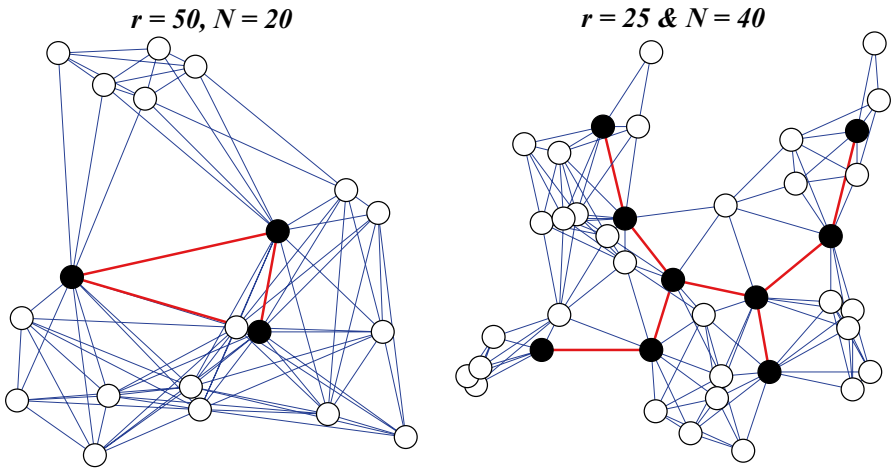

Fig. 11 Examples of obtaining backbone using SA-MCDS

Table 10 presents the minimum and average values of the MCDS obtained by the proposed methods and the above-mentioned methods. According to the significance test by rank-sum test shown in Table 11, there is no significant difference at level 0.05 between the results of the compared methods shown in Table 10. However, by careful investigation of the results shown in Table 10, we can generally conclude that our two proposed methods perform the best for network instances with large number of wireless nodes. Figure 12 shows the comparison results of different methods in big-size networks. Although our proposed methods performance is almost the same as other methods, other methods are slightly better in big ranges of small size networks. Consequently, we can claim that our two methods are more practical than others in wireless networks design and management. Specifically in wireless sensor networks, typically we have a large number of sensors in unit area. Conserving energy/battery consumption is of essence here, so finding small number of wireless sensors who are responsible for communications and controlling other wireless sensor nodes is crucial. Hence, our methods are more suitable for wireless sensor networks design and management.

A node density in a network can be defined as the density distribution of other nodes around a each node. The following relation can estimate the average number of nodes within the range of each node in a graph which may be used as an approximation of the node degree.

$$\nu = \frac{\pi r^2}{L^2/N}, \tag{4}$$

where r, L and N are the range, area length and number of nodes, respectively, as shown in Table 2. In Eq. 4, the numerator represents the range area of each nodes, while the denominator uniformly estimates area per node. Finding a MCDS in low

Table 10 Comparison results for network clustering

Network	ACO		ACO+PCS		GRASP		MA-MCDS		SA-MCDS	
ID	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Net^1_{60}	20	21.6	19	21.2	19	19.8	17	18.0	**13**	18.0
Net^1_{70}	16	17.0	15	16.2	**14**	15.1	15	15.3	15	15.4
Net^1_{80}	12	14.0	12	13.1	12	12.0	**11**	11.9	12	13.2
Net^1_{90}	11	11.8	11	11.6	**10**	10.6	11	11.7	11	11.9
Net^1_{100}	**8**	9.0	**8**	8.9	**8**	8.2	10	11.2	10	10.8
Net^1_{110}	8	8.5	8	8.5	**7**	7.8	8	8.4	9	13.4
Net^1_{120}	7	7.5	7	7.2	**6**	6.1	8	8.9	8	8.9
Net^2_{80}	23	24.7	22	23.6	22	22.9	**14**	15.5	15	16.8
Net^2_{90}	22	23.8	21	23.6	20	20.7	**18**	19.4	19	21.7
Net^2_{100}	17	20.0	17	19.0	**17**	17.9	18	19.4	**17**	19.7
Net^2_{110}	**15**	17.2	**15**	16.8	**15**	15.9	18	20.9	20	20.9
Net^2_{120}	15	16.2	14	15.5	**13**	13.8	14	14.6	**13**	14.4
Net^3_{70}	46	50.7	46	49.6	45	46.5	**36**	37.3	37	38.9
Net^3_{80}	41	43.7	41	43.9	**35**	37.5	**35**	37.4	37	39.5
Net^3_{90}	34	36.0	33	35.7	**30**	30.9	34	34.9	35	35.9
Net^3_{100}	28	30.8	23	31.0	**25**	25.8	29	31.4	30	33.4
Net^3_{110}	23	27.4	22	26.4	**22**	22.7	28	30.0	28	29.4
Net^3_{120}	21	23.6	21	23.4	**18**	19.1	27	27.8	25	27.8
Net^4_{100}	46	50.7	46	49.6	45	46.5	**33**	34.4	34	35.2
Net^4_{110}	43	44.9	42	44.8	**37**	39.5	39	40.6	38	40.3
Net^4_{120}	37	39.9	37	39.8	**34**	35.4	**34**	34.8	35	35.9
Net^4_{130}	32	34.7	32	34.9	**29**	30.5	35	35.8	36	37.8
Net^4_{140}	30	31.3	29	31.3	**25**	34.3	33	33.6	34	35.3
Net^4_{150}	28	29.6	26	28.8	**23**	24.3	29	30.8	30	33.8
Net^4_{160}	24	26.6	25	26.5	**22**	22.3	30	30.8	28	30.9
Net^5_{130}	60	64.5	60	64.3	57	58.6	**43**	48.6	46	48.2
Net^5_{140}	53	57.2	52	57.0	50	52.3	**47**	50.5	49	49.8
Net^5_{150}	51	54.9	51	54.4	46	48.5	43	44.2	**35**	45.9
Net^5_{160}	47	50.5	45	49.8	43	43.7	**41**	41.6	**41**	42.8
Net^6_{200}	55	58.6	52	58.8	**49**	50.4	59	61.5	56	63.6
Net^6_{210}	51	53.5	50	52.8	45	46.1	**43**	46.8	45	47.9
Net^6_{220}	47	48.9	45	48.4	**40**	42.1	48	49.2	47	49.1
Net^6_{230}	44	47.5	44	46.9	**39**	39.8	43	44.9	45	46.8
Net^7_{200}	79	82.0	79	81.5	73	75.4	**54**	56.2	57	58.5
Net^7_{210}	75	79.1	74	78.2	67	70.0	**64**	69.2	66	67.9
Net^7_{220}	68	72.6	69	73.8	62	67.0	**61**	64.4	62	64.5
Net^7_{230}	66	69.2	66	68.9	59	60.9	**55**	57.7	**55**	55.6
Net^8_{210}	99	101.6	98	104.0	90	94.7	**70**	75.0	**70**	75.5
Net^8_{220}	88	95.4	91	97.6	82	87.9	**73**	77.3	74	79.8

المنارة للاستشارات

Table 10 (continued)

Network ID	ACO		ACO+PCS		GRASP		MA-MCDS		SA-MCDS	
	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Net^8_{230}	86	91.4	86	90.3	78	81.6	**69**	70.8	71	72.1
Net^8_{240}	82	85.8	80	84.1	74	76.1	**65**	68.0	67	72.3
Overall Avg.	40.44	43.27	39.85	42.97	36.76	38.57	35.66	37.58	35.98	38.52

Table 11 Rank-sum test for comparison results in Table 10

Compared methods		R^-	R^+	p value	Best method
MA-MCDS	ACO	665	196	0.5221	–
MA-MCDS	ACO+PCS	622.5	235.5	0.6002	–
MA-MCDS	GRASP	467	394	0.9482	–
SA-MCDS	ACO	653.5	207.5	0.5683	–
SA-MCDS	ACO+PCS	609.5	251.5	0.6561	–
SA-MCDS	GRASP	443.5	417.5	0.9889	–
MA-MCDS	SA-MCDS	591	270	0.8783	–

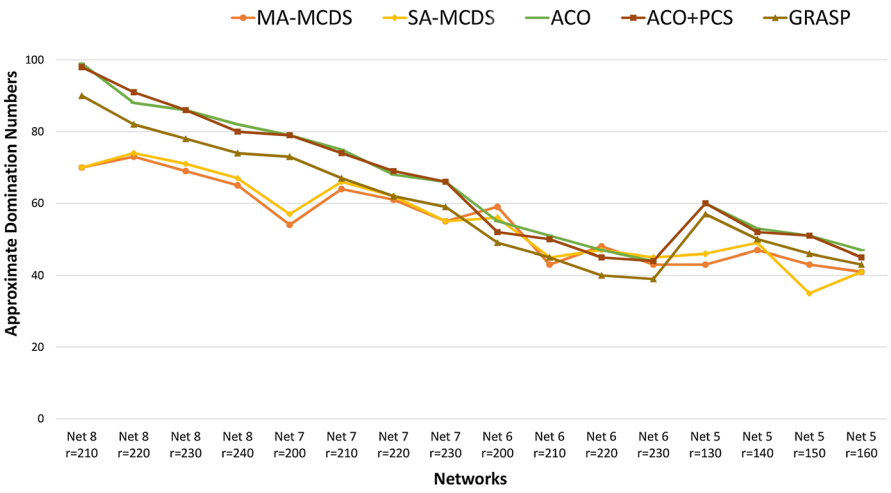

Fig. 12 Compared results for big-size networks

density networks is harder than the high density ones since the average number of dominated nodes by a MCDS member is decreased.

Figure 13 shows that the proposed methods could obtained better results than the other compared methods. Specifically, both of MA-MCDS and SA-MCDS could obtain better results in 21 out of 23 graphs with low density of node distributions.

المنارة للاستشارات

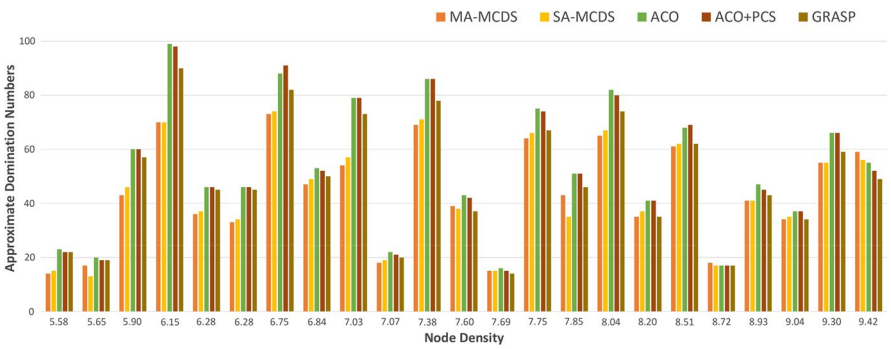

Fig. 13 Compared results for low-density networks

9 Conclusion

In this paper, we investigated the minimum connected dominating set problem. We proposed two new algorithms to solve the MCDS problem: the first algorithm is called Memetic Algorithm for solving MCDS problem (MA-MCDS), and the second algorithm is called Simulated Annealing for solving MCDS problem (SA-MCDS). Also, we presented a new objective function to be used by both algorithms to achieve a better performance. In addition, we tested the proposed algorithms and applied them to identify the network virtual backbone structure in static wireless sensors networks by reformulating them to the minimum connected dominating set problem. Our experimental results on different standard benchmark test graphs show the efficiency of the proposed algorithms especially SA-MCDS. Test results demonstrated that both MA-MCDS and SA-MCDS are very efficient in terms of computational costs and solution quality to compute and identify MCDS.

The proposed methods can be extended by enhancing the filtering and connecting procedures as future works. Specifically, the filtering procedure can be enhanced by studying the relations between nodes to be removed. In addition, the connecting procedure can be enhanced by studying relations between solution connected components.

Acknowledgements This work was funded by the Scientific Research Deanship at Umm Al-Qura University, the Kingdom of Saudi Arabia, Award Project Number (43508016). The authors would like to express their sincere gratitude to the anonymous referees and the editors for their useful and valuable comments and suggestions to improve the quality of the paper.

References

1. Gao, X., Zhu, X., Li, J., Fan, W., Chen, G., Ding-Zhu, D., Tang, S.: A novel approximation for multi-hop connected clustering problem in wireless networks. IEEE/ACM Trans. Netw. **25**(4), 2223–2234 (2017)
2. Mohanty, J.P., Mandal, C., Reade, C.: Distributed construction of minimum connected dominating set in wireless sensor network using two-hop information. Comput. Netw. **123**, 137–152 (2017)

المنارة للاستشارات

3. Mohanty, J.P., Mandal, C., Reade, C., Das, A.: Construction of minimum connected dominating set in wireless sensor networks using pseudo dominating set. Ad Hoc Netw. **42**, 61–73 (2016)
4. Narasimha Raghavan, V., Arvind Ranganath, R., Bharath, N., Khan, M.F.: Simple and efficient backbone algorithm for calculating connected dominating set in wireless adhoc networks. Int. J. Electron. Circuits Syst. **1**(3), 3–18 (2007)
5. Thai, M.T., Tiwari, R., Du, D.-Z.: On construction of virtual backbone in wireless ad hoc networks with unidirectional links. IEEE Trans. Mobile Comput. **7**(9), 1098–1109 (2008)
6. Torkestani, J.A., Meybodi, M.R.: An intelligent backbone formation algorithm for wireless ad hoc networks based on distributed learning automata. Comput. Netw. **54**(5), 826–843 (2010)
7. Wan, P.-J., Alzoubi, K.M., Frieder, O.: Distributed construction of connected dominating set in wireless ad hoc networks. In: Proceedings INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, pp. 1597–1604. IEEE (2002)
8. Yu, F., Xia, X., Li, W., Tao, J., Ma, L., Cai, Z.-Q.: Critical node identification for complex network based on a novel minimum connected dominating set. Soft Comput. **21**, 1–9 (2016)
9. Fahong, Y., Xia, X., Li, W., Tao, J., Ma, L., Cai, Z.: Critical node identification for complex network based on a novel minimum connected dominating set. Soft Comput. **21**(19), 5621–5629 (2017)
10. Paradis, L., Han, Q.: A survey of fault management in wireless sensor networks. J. Netw. Syst. Manag. **15**(2), 171–190 (2007)
11. Ferreira, C., Guardalben, L., Gomes, T., Sargento, S., Salvador, P., Robalo, D., Velez, F.J.: Supporting unified distributed management and autonomic decisions: design, implementation and deployment. J. Netw. Syst. Manag. **25**(2), 416–456 (2017)
12. Haynes, T.W., Hedetniemi, S., Slater, P.: Fundamentals of Domination in Graphs. CRC Press, Boca Raton (1998)
13. Michael, R.G., David, S.J.: Computers and Intractability: A Guide to the Theory of np-Completeness. WH Freeman & Co., San Francisco (1979)
14. Glover, F., Kochenberger, G.A.: Handbook of Metaheuristics. Springer, Berlin (2003)
15. Merz, P.: Advanced fitness landscape analysis and the performance of memetic algorithms. Evolut. Comput. **12**(3), 303–325 (2004)
16. Aarts, E., Korst, J.: Selected topics in simulated annealing. In: Ribeiro, C.C., Hansen, P. (eds.) Essays and Surveys in Metaheuristics, pp. 1–37. Springer, Berlin (2002)
17. Hedar, A.-R., Fukushima, M.: Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization. Optim. Methods Softw. **19**(3–4), 291–308 (2004)
18. Hedar, A.-R., Fukushima, M.: Derivative-free filter simulated annealing method for constrained continuous global optimization. J. Glob. Optim. **35**(4), 521–549 (2006)
19. Kirkpatrick, S., Vecchi, M.P., et al.: Optimization by simmulated annealing. Science **220**(4598), 671–680 (1983)
20. Aarts, E., Korst, J., Michiels, W.: Simulated annealing. In: Burke, E.K. (ed.) Search Methodologies, pp. 187–210. Springer, Berlin (2005)
21. Hedar, A.-R., Fukushima, M.: Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization. Optim. Methods Softw. **17**(5), 891–912 (2002)
22. Moscato, P., et al.: On genetic crossover operators for relative order preservation. C3P Report, pp. 778 (1989)
23. Pastorino, M., Caorsi, S., Massa, A., Randazzo, A.: Reconstruction algorithms for electromagnetic imaging. IEEE Trans. Instrum. Meas. **53**(3), 692–699 (2004)
24. Hedar, A.-R., Ismail, R.: Hybrid genetic algorithm for minimum dominating set problem. In: Computational Science and Its Applications–ICCSA 2010, pp. 457–467. Springer, Berlin (2010)
25. Li, R., Hu, S., Gao, J., Zhou, Y., Wang, Y., Yin, M.: Grasp for connected dominating set problems. Neural Comput. Appl. **28**, 1–9 (2016)
26. Misra, R., Mandal, C.: Minimum connected dominating set using a collaborative cover heuristic for ad hoc sensor networks. IEEE Trans. Parallel Distrib. Syst. **21**(3), 292–302 (2010)
27. Yadav, A.K., Yadav, R.S., Singh, R., Singh, A.K.: Connected dominating set for wireless ad hoc networks: a survey. Int. J. Eng. Syst. Modell. Simul. **7**(1), 22–34 (2014)
28. Coelho, R.S., Moura, P.F.S., Wakabayashi, Y.: The k-hop connected dominating set problem: hardness and polyhedra. Electron. Notes Discrete Math. **50**, 59–64 (2015)
29. Hongjie, D., Ding, L., Weili, W., Kim, D., Pardalos, P.M., Willson, J.: Connected dominating set in wireless networks. In: Pardalos, P.M., Du, D.-Z., Graham, R.L. (eds.) Handbook of Combinatorial Optimization, pp. 783–833. Springer, Berlin (2013)

المنارة للاستشارات

30. Gupta, A., Kumar, A., Roughgarden, T.: Simpler and better approximation algorithms for network design. In: Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing, pp. 365–372 (2003)
31. Swamy, C., Kumar, A.: Primal-dual algorithms for connected facility location problems. Algorithmica **40**(4), 245–269 (2004)
32. Cheng, X., Ding, M., Chen, D.: An approximation algorithm for connected dominating set in ad hoc networks. In: Proceedings of International Workshop on Theoretical Aspects of Wireless Ad Hoc, Sensor, and Peer-to-Peer Networks (TAWN) (2004)
33. Jie, W., Dai, F.: A generic distributed broadcast scheme in ad hoc wireless networks. IEEE Trans. Comput. **53**(10), 1343–1354 (2004)
34. Rawat, P., Singh, K.D., Chaouchi, H., Bonnin, J.M.: Wireless sensor networks: a survey on recent developments and potential synergies. J. Supercomput. **68**(1), 1–48 (2014)
35. Hongwei, D., Ye, Q., Weili, W., Lee, W., Li, D., Dingzhu, D., Howard, S.: Constant approximation for virtual backbone construction with guaranteed routing cost in wireless sensor networks. In: 2011 Proceedings IEEE INFOCOM, pp. 1737–1744. IEEE (2011)
36. Ren, S., Yi, P., Lin, Z., Guo, C., Wu, Y.: Constructing minimum connected dominating sets with constant update time in wireless ad-hoc sensor networks. In: 2014 IEEE 17th International Conference on Computational Science and Engineering (CSE), pp. 1570–1576. IEEE (2014)
37. Ren, S., Yi, P., Hong, D., Wu, Y., Zhu, T.: Distributed construction of connected dominating sets optimized by minimum-weight spanning tree in wireless ad-hoc sensor networks. In: 2014 IEEE 17th International Conference on Computational Science and Engineering (CSE), pp. 901–908. IEEE (2014)
38. Kim, D., Yiwei, W., Li, Y., Zou, F., Ding-Zhu, D.: Constructing minimum connected dominating sets with bounded diameters in wireless networks. IEEE Trans. Parallel Distrib. Syst. **20**(2), 147–157 (2009)
39. Wang, L., Wan, P.-J., Yao, F.: Minimum cds in multihop wireless networks with disparate communication ranges. IEEE Trans. Mobile Comput. **1**(3), 162–168 (2013)
40. Mohanty, J.P., Mandal, C.: A distributed greedy algorithm for construction of minimum connected dominating set in wireless sensor network. In: Applications and Innovations in Mobile Computing (AIMoC), 2014, pp. 104–110. IEEE (2014)
41. Kui, X., Wang, J., Zhang, S.: A data gathering algorithm based on energy-balanced connected dominating sets in wireless sensor networks. In: 2013 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1139–1144. IEEE (2013)
42. Malatras, A., Hadjiantonis, A.M., Pavlou, G.: Exploiting context-awareness for the autonomic management of mobile ad hoc networks. J. Netw. Syst. Manag. **15**(1), 29–55 (2007)
43. Aoun, B., Boutaba, R.: Clustering in wsn with latency and energy consumption constraints. J. Netw. Syst. Manag. **14**(3), 415–439 (2006)
44. Guha, S., Khuller, S.: Approximation algorithms for connected dominating sets. Algorithmica **20**(4), 374–387 (1998)
45. Ruan, L., Hongwei, D., Jia, X., Weili, W., Li, Y., Ko, K.-I.: A greedy approximation for minimum connected dominating sets. Theor. Comput. Sci. **329**(1–3), 325–330 (2004)
46. Weili, W., Hongwei, D., Jia, X., Li, Y., Huang, S.C.-H.: Minimum connected dominating sets and maximal independent sets in unit disk graphs. Theor. Comput. Sci. **352**(1–3), 1–7 (2006)
47. Kamei, S., Kakugawa, H.: A self-stabilizing 6-approximation for the minimum connected dominating set with safe convergence in unit disk graphs. Theor. Comput. Sci. **428**, 80–90 (2012)
48. Dai, F., Jie, W.: On constructing k-connected k-dominating set in wireless ad hoc and sensor networks. J. Parallel Distrib. Comput. **66**(7), 947–958 (2006)
49. Das, B., Sivakumar, R., Bharghavan, V.: Routing in ad hoc networks using a spine. In: Proceedings Sixth International Conference on Computer Communications and Networks, 1997, pp. 34–39. IEEE (1997)
50. Wu, J., Li, H.: On calculating connected dominating set forefficient routing in ad hoc wireless networks. In Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, pp. 7–14. ACM (1999)
51. Das, B., Sivakumar, R., Bharghavan, V.: Routing in ad-hoc network using a virtual backbone. Proc. ACM SIGCOMM **97**, 1–20 (1997)
52. Jovanovic, R., Tuba, M.: Ant colony optimization algorithm with pheromone correction strategy for the minimum connected dominating set problem. Comput. Sci. Inf. Syst. **10**(1), 133–149 (2013)

المنارة للاستشارات

53. Dagdeviren, Z.A., Aydin, D., Cinsdikici, M.: Two population-based optimization algorithms for minimum weight connected dominating set problem. Appl. Soft Comput. **59**, 644–658 (2017)
54. Kumar, G., Rai, M.K.: An energy efficient and optimized load balanced localization method using cds with one-hop neighbourhood and genetic algorithm in wsns. J. Netw. Comput. Appl. **78**, 73–82 (2017)
55. Deb, K.: Multi-objective Optimization Using Evolutionary Algorithms, vol. 16. Wiley, Hoboken (2001)
56. Baker, J.E.: Adaptive selection methods for genetic algorithms. In Proceedings of an International Conference on Genetic Algorithms and Their Applications, pp. 101–111. ACM, Hillsdale (1985)
57. Herrera, F., Lozano, M., Verdegay, J.L.: Tackling real-coded genetic algorithms: operators and tools for behavioural analysis. Artif. Intell. Rev. **12**(4), 265–319 (1998)
58. Hedar, A.-R., Ismail, R., El Sayed, G.A., Khayyat, : K.M.J.: Two meta-heuristics for the minimum connected dominating set problem with an application in wireless networks. In 2015 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence (ACIT-CSI), pp. 355–362. IEEE (2015)
59. Hedar, A.-R., Ismail, R.: Simulated annealing with stochastic local search for minimum dominating set problem. Int. J. Mach. Learn. Cybern. **3**(2), 97–109 (2012)
60. Sanchis, L.A.: Experimental analysis of heuristic algorithms for the dominating set problem. Algorithmica **33**(1), 3–18 (2002)
61. García, S., Fernández, A., Luengo, J., Herrera, F.: A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. Soft Comput. **13**(10), 959–977 (2009)
62. Sheskin, David J.: Handbook of Parametric and Nonparametric Statistical Procedures. CRC Press, Boca Raton (2003)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Abdel-Rahman Hedar holds a Doctor of Informatics degree from Kyoto University, Japan in 2004. He also received his B.Sc. and M.Sc. (Mathematics) from Assiut University, Egypt in 1993 and 1997, respectively. He is currently an associate professor at Computer Science Department in Jamoum, Umm Al-Qura University, Makkah, Saudi Arabia. He is also an associate professor of artificial intelligence in Assiut University since January 2012. His research includes meta-heuristics, global optimization, machine learning, data mining, bioinformatics, graph theory and parallel programming. He has published over 70 papers in international journals and conferences. From July 2005 to July 2007, he was a JSPS research fellow in Kyoto University, Japan.

Rashad Ismail is an assistant professor at King Khalid University, Saudi Arabia. He obtained his Ph.D. in Scientific Computing from Assiut University, Egypt, 2011, M.Sc. from University of Damascus, Syria, 2007, and B.Sc. in Mathematics from University of Taiz, Yemen, 1998. Research interests include Graph theory and its Applications, the Minimum Dominating Set Problem, Maximum Independent Set Problem, Maximum Cliques and the Minimum Spanning Tree and Fuzzy graph.

Gamal A. El-Sayed is currently an assistant professor at Umm Al-Qura University, Saudi Arabia, and Assiut University, Egypt (on leave). He received his Ph.D. degree in Computer Engineering from University of Connecticut, USA. His research interests include distributed computing/systems, wireless sensor networks deployment/management, fault-tolerance, pervasive computing, high performance computing, and real-time systems.

Khalid M. Jamil Khayyat received his Ph.D. degree in electrical and computer engineering from University of Victoria, Canada. He also received his master degree form Colorado State University, USA, and his B.Sc. from Umm Al-Qura University, Saudi Arabia. He is now an assistant professor at Umm Al-Qura University, Saudi Arabia. His research interests are computer networks performance modelling, wireless ad hoc networks, wireless sensor networks and wireless applications.

المنارة للاستشارات

Affiliations

Abdel-Rahman Hedar[1,2] · Rashad Ismail[3,4] · Gamal A. El-Sayed[1,5] · Khalid M. Jamil Khayyat[6]

Rashad Ismail
rismail@kku.edu.sa

Gamal A. El-Sayed
gaelsayed@uqu.edu.sa; gamal@eng.au.edu.eg

Khalid M. Jamil Khayyat
kmkhayyat@uqu.edu.sa

1 Present Address: Department of Computer Science in Jamoum, Umm Al-Qura University, Makkah 25371, Saudi Arabia

2 Department of Computer Science, Assiut University, Assiut 71526, Egypt

3 Department of Mathematics and Computer Science, Ibb University, Ibb 70270, Yemen

4 Present Address: Department of Mathematics, King Khalid University, 61913 Mohail Assir, Saudi Arabia

5 Department of Electrical Engineering, Assiut University, Assiut 71516, Egypt

6 Department of Computer Engineering, Umm Al-Qura University, Makkah 24381, Saudi Arabia

المنارة للاستشارات

Journal of Network & Systems Management is a copyright of Springer, 2019. All Rights Reserved.

www.manaraa.com