A Self-stabilizing Memetic Algorithm for Minimum Weakly Connected Dominating Set Problems

Dangdang Niu[1] , **Minghao Yin**[2]

[1]College of Information Engineering, Northwest A&F University, Yangling, Shaanxi 712100, China;
[2]School of Computer Science and Information Technology, Northeast Normal University, Changchun 130117, China;
niudd@nwafu.edu.cn, ymh@nenu.edu.cn

Abstract

The minimum weakly connected dominating set problem (MWCDSP), a variant of the classical minimum dominating set problem, has extensive real-world applications, such as mobile ad hoc networks, distributed sensor network, etc. To address this problem, a 0-1 integer linear programming (ILP) model and a framework of memetic algorithm (MA) for MWCDSP are proposed. Specially, a self-stabilizing algorithm is integrated to the MA for MWCDSP, which transforms the infeasible solutions created by the operations of initializing populations, crossover and mutation etc., into minimal feasible solutions in linear time. Moreover, a simple local search algorithm is used to refine the solutions obtained by the self-stabilizing algorithm in MA. The experimental results on three kinds of graph instances show that our MA for MWCDSP significantly outperforms the famous CPLEX, the self-stabilizing algorithm `MWCDS`, the genetic algorithm and the local search procedure.

1 Introduction

Consider an undirected, connected graph $G = (V, E)$, where V denotes the set of vertices, E denotes the set of edges. A Dominating Set (DS) is a subset $S \subseteq V$ such that for every vertex $v \in V$, either $v \in S$, or there exist an edge $(u, v) \in E$ for some $u \in S$. If the subgraph induced by a DS is connected, we call it Connected Dominating Set (CDS). The subgraph weakly induced by W ($W \subseteq V$) is defined as $S_w = (N[W], E \cap (W \times N[W]))$, where the set $N[W]$ collects one-hop neighbor of W and itself. The dominating set W is a Weakly Connected Dominating set (WCDS) if the subgraph weakly induced by W is connected. The minimum weakly connected dominating set problem (MWCDSP) is to identify a weakly connected dominating set W of minimum size.

The minimum weakly connected dominating set problem was introduced in 1997 by Dunbar et al. [Dunbar *et al.*, 1997]. The concept of WCDS has been widely used for clustering in mobile ad hoc networks and distributed sensor network [Pathan and Seon, 2006; Han and Jia, 2007], which is a better method for clustering than the CDS in general [Yu *et al.*, 2012]. The WCDS is very suitable for cluster formation [Han and Jia, 2007], which was used to reduce the number of cluster heads in the network [Pathan and Seon, 2006] and solve secure clustering problem which plays an important role in distributed sensor networks [Pathan and Seon, 2006; Du *et al.*, 2012]. As finding the minimum WCDS in an arbitrary graph is a NP-Hard problem [Dunbar *et al.*, 1997], most of the above works only proposed polynomial distributed algorithm for WCDS construction in wireless ad hoc networks. In addition to the computational challenge of NP-Hard complexity for the general graphs, many researches also focus on the computational complexity of MWCDSP for some special kinds of graphs [Domke *et al.*, 2005; Lemańska and Patyk, 2008; Raczek and Cyman, 2019]. These researches further show the theoretical significance of studying MWCDSP.

Compared to the minimum dominating set problem (MDSP) and the minimum connected dominating set problem (MCDSP) [Wu *et al.*, 2022; Bonamy *et al.*, 2021], which are two classical combinatorial optimization problems, more conditions need to be considered when solving MWCDSP from their definitions. The main challenges of MWCDSP is that there are two connected modes for each pair of vertices in any feasible solution, i.e. directly connected or indirectly connected (weakly connected) through a common neighbor vertex, which induces more candidate solutions with respect to MCDSP.

In order to obtain high-quality WCDS, Some self-stabilizing algorithms are specially proposed to constructed minimal weakly connected dominating sets in connected graphs [Kamei and Kakugawa, 2007; Srimani and Xu, 2007; Ding *et al.*, 2016]. Most of the above algorithms stabilize in polynomial steps by using different daemon strategies, such as unfair central daemon, synchronous daemon, and distributed daemon etc. Specially, Ding et al. proposed a linear time self-stabilizing algorithm `MWCDS` for minimal weakly connected dominating sets, which terminates in $O(n)$ steps using a synchronous daemon for an arbitrary connected graph with n nodes [Ding *et al.*, 2016]. These self-stabilizing algorithms can quickly construct a minimal weakly connected dominating sets, but it is still difficult to apply them in solving MWCDSP for complex graphs which usually contains many local optimal solutions.

Despite the theoretical and practical significance of the MWCDSP, our literature review indicates that there are few

heuristics or exact algorithms which help to satisfy the good performance in real-world applications of MWCDSP. The memetic algorithm (MA), which is a classical heuristic method and can usually provide high-quality approximate solutions, has been widely used to solve graph optimization problems, such as minimum independent dominating set problem [Wang *et al.*, 2018b], graph coloring problem [Dokeroglu and Sevinc, 2021], sum graph coloring problem [Moalic and Gondran, 2019], etc. The above discussions motivate us to design efficient MA for MWCDSP, which is a theoretically feasible way to provide good solutions for real-world applications of MWCDSP.

In this paper, we present an ILP model which is an exact solution for MWCDSP, and a framework of MA for the MWCDSP. In our MA, the self-stabilizing algorithm `MWCDS` is used to generate initial populations by convert all random solutions to minimal WCDSs. Then selection uses the well known tournament selection operator, and a two-point crossover and mutation operations are also employed by MA. We also design a simple local search procedure to refine the solutions obtained by `MWCDS`. In our local search procedure, two score functions are used to search (*k*-1)-size WCDS after a *k*-size WCDS is found. Experimentally, our MA is competitive with the famous CPLEX, the self-stabilizing algorithm `MWCDS`, the genetic algorithm and the local search algorithm.

The reminder of this paper is organized as follows: Section 2 presents some preliminary knowledge, the ILP model and the self-stabilizing algorithm for MWCDSP . Section 3 presents the local search procedure for MWCDSP. The concrete MA for the minimum weakly connected dominating set problem are presented in Section 4. Section 5 presents the experimental results. Finally, Section 6 gives our conclusions and future works.

2 Preliminaries

2.1 Basic definitions

Consider an undirected, connected graph $G = (V, E)$, where V denotes the set of vertices, E denotes the set of edges. We shall use $N(v)$ to represent the neighbor vertex set of v. We also denote $N[v] = N(v) \cup \{v\}$. The shortest hop path from u to v is denoted as $hop(u, v)$. Then, the neighboring vertices with hop paths in [1, i] of the vertex v are collected in $N_i(v) = \{u \mid 1 \leq hop(u, v) \leq i\}$. In addition, we also denote $N_i[v] = N_i(v) \cup v$. We shall extend the above concepts to the vertex set. Given a vertex set $VS \subseteq V$, the neighboring vertices with hop paths in [1, i] of the vertex set VS is presented as $N_i(VS) = \{u \mid 1 \leq hop(u, v) \leq i, \exists\, v \in VS\}$. In addition, we denote $N_i[VS] = N_i(VS) \cup VS$. The dominating vertex set of a given vertex set S is denoted as $dom(S) = N[S]$, which collects the vertices dominated by VS. Some important definitions are described as follows.

Definition 1 (*weakly induced subgraph*, *WIS*). Given an undirected graph $G = (V, E)$, $WIS_G(W) = (N[W], E_1)$, $W \subseteq V$, $E_1 = \{E \cap (W \times N[W])\}$, subgraph $WIS_G(W)$ is called the weakly induced subgraph of graph G based on W.

Definition 2 (*weakly connected dominating set*, *WCDS*). Given an undirected, connected graph $G = (V, E)$, the weakly connected dominating set W of G is a dominating set such that $WIS_G(W)$ is connected.

Given an undirected, connected graph $G = (V, E)$, the Minimum Weakly Connected Dominating Set Problem (MWCDSP) is to find a weakly connected dominating set W of G with minimum cardinality.

2.2 The ILP model for the MWCDSP

We also describe a 0-1 integer linear programming (ILP) model for the MWCDSP in this section. Our constructing method of ILP model for MWCDSP is mainly derived from the ILP modes for minimum dominating tree problem (MDTP) in [Shin *et al.*, 2010]. Obviously, if all weights of edges are unified, the solutions of MDTP are the spanning trees of the minimum connected dominating sets. In order to maintain the weakly connectivity of solutions in ILP model for MWCDSP, we add a 2-dimensional adjacency matrix E_{weak} to collect all edges which connect V with $N_2(V)$, which means that for any vertex $v \in V$, there is an edge used to connect v with each vertex of $N_2(v)$ in E_{weak}. For any vertex pair $(u, v)(u, v \in V)$, if there exists a edge between them in E_{weak}, $E_{weak}[u, v] = 1$, $E_{weak}[u, v] = 0$ otherwise. Now, we define the following decision variables:

- Let $s = \{E_1, E_2, \ldots, E_k\}$ ($\forall E_i \subseteq E_{weak}$) be a candidate solution.
- $\forall x_{uv} \in \{0,1\}$ $(u, v \in V)$ is a binary decision variable that is equal to 1 if and only if (u, v) is selected in the optimal solution, 0 otherwise.
- $\forall y_t \in \{0,1\}$ ($t \in V$) is a binary decision variable that is equal to 1 if and only if t is selected in the optimal solution, 0 otherwise.

We obtain the following ILP model for the WSCP:

$$f(s) = 1 + \sum_{u,v \in V} x_{uv} \tag{1}$$

$$\begin{cases} x_{uv} \leq E_{weak}[u, v], & \forall u, v \in V \\ x_{uv} = x_{vu}, & \forall u, v \in V \\ y_u + y_v \geq 2 \times x_{uv}, & \forall u, v \in V \\ \sum\limits_{u,v \in S} x_{uv} \leq |S| - 1, & \forall S \subset V \\ \sum\limits_{u,v \in V} x_{uv} = \sum\limits_{t \in V} y_t - 1 & \\ \sum\limits_{t \in N[V]} y_t \geq 1, & \forall t \in V \end{cases} \tag{2}$$

The objective function of formula (1) is to minimize the sum of vertices in the weakly connected dominating set. In formula (1), the spanning tree consists of all edges with $x_{uv} = 1$, which is in correspondence with the weakly connected dominating set. Since the sum of vertices in a connected graph is the sum of edges in its spanning tree plus 1, we add 1 to $\sum_{u,v \in V} x_{uv}$ in formula (1). In formula (2), the first two constraints ensures that the set of edges in any solution is a subset of the edges in E_{weak}. The third constraint shows the relation between vertices and edges in optimal solution, that is, for each edge selected in optimal solution, the end vertices has to be selected. The fourth and fifth constraints are similar to the formulation in Minimum

Spanning Tree problem to guarantee that the solution is a tree [Shin *et al.*, 2010]. The fourth constraint requires that the arbitrary subgraph constructed by any subset of edgs set $\{x_{uv} \mid x_{uv} = 1\}$ can not be a cycle. So, combining the fourth constraint with the fifth constraint, the graph constructed by $\{x_{uv} \mid x_{uv} = 1\}$ must be a tree. The last constraint shows the constraint of the dominating set. Basically, for each vertex v, at least one of its neighbors or itself has to be selected to the optimal solution.

2.3 The self-stabilizing algorithm

The self-stabilizing algorithms can transform infeasible solutions into minimal feasible solutions. The self-stabilizing algorithm MWCDS has been proposed by Ding et al., which is a linear time complexity algorithm [Ding *et al.*, 2016]. MWCDS terminates in *O*(*n*) steps using a synchronous daemon for an arbitrary connected graph with *n* nodes. In their experiments, MWCDS can be convergence quickly, which is the best self-stabilizing algorithm for MWCDSP.

In the process of constructing meta heuristic algorithms for MWCDSP, one of the most important steps is that how to make an infeasible solution be feasible. Actually, a feasible solution S of MWCDSP must be weakly connected and all vertices not in S must be dominated by the vertices in S, so one can transform infeasible solutions into feasible solutions through adding appropriate nodes repeatedly into them with some greedy selection rules. However, the qualities of solutions can not be guaranteed. This motivates us to use the self-stabilizing algorithm MWCDS to solving the infeasible solutions in our meta heuristic algorithm.

In the MWCDS algorithm, a special node $root$ is required. Since the MWCDS algorithm will be executed repeatedly, we compute the shortest distance of any node i to $root$ before searching. For the MWCDS algorithm, two important variables need to be maintained [Ding *et al.*, 2016]:

– A boolean membership flag s_i indicating membership status of node i in a solution; The current solution S is the set of nodes with s_i = 1.

– A non-negative integer variable d_i that keeps track of the shortest distance of node i to $root$.

For any solution, we also define two additional Boolean predicates for each node $i \neq root$ [Ding *et al.*, 2016]:

$$enter_i \equiv (s_i = 0) \wedge (\forall j \in N_{\leq}(i) : s_j = 0) \quad (3)$$

$$leave_i \equiv (s_i = 1) \wedge (\exists j \in N_{\leq}(i) : s_j = 1) \quad (4)$$

In the above formulas, $N_{\leq}(i)$ denotes the set of nodes $j \in N(i)$ with $d_j \leq d_i$.

For any solution S, each vertex enters into or leave from S are decided by the formula (3) and formula (4) respectively. If $enter_i = 0 \wedge leave_i = 0$ is satisfied for each vertex i, the current solution S is a minimal weakly connected dominating set, which can be obtained in linear time [Ding *et al.*, 2016].

3 A simple local search algorithm for MWCDSP

Though some minimal weakly connected dominating set can be obtained by the MWCDS algorithm from any initial solution, the size of the solution obtained by MWCDS is strongly related to the initial solution and the selected $root$ node. A graph G and the initial solution $\{a, d, e\}$ of MWCDSP on G is depicted as Figure 1. Obviously, the solution $\{a, d, e\}$ is weakly connected, but the set of vertices $\{g, h, i\}$ are not dominated by it. So the solution $\{a, d, e\}$ is infeasible.

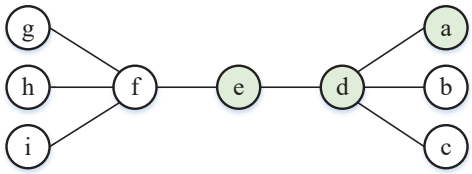

Figure 1: An exapmle of infeasible MWSCDP solution on the graph.

If we use MWCDS to solve the minimal weakly connected dominating set with the initial solution $\{a, d, e\}$ on G, in which the root a is randomly selected, a minimal weakly connected dominating set $\{a, b, c, e, g, h, i\}$ is obtained. Obviously, $\{a, b, c, e, g, h, i\}$ is weakly connected, and all vertices are dominated. The result of MWCDS for Figure 1 is shown as Figure 2.

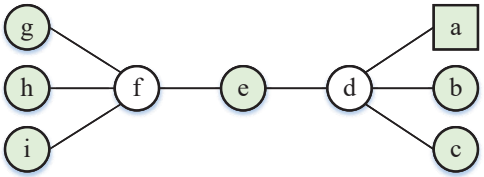

Figure 2: The solution obtained by MWCDS with the root node a for the solution in Figure 1.

The size of the solution in Figure 2 is 7. Actually, the optimal solution of MWCDSP on G is $\{d, f\}$, whose size is 2. So a minimal weakly connected dominating set can be got based on the MWCDS algorithm, but the quality of it can not be guaranteed. We need make more explorations for the solution obtained by MWCDS.

In this section, we introduce a simple local search algorithm to refine the solutions obtained by MWCDS, which is shown as Algorithm 1. In Algorithm 1, two assessment functions are designed for discriminate different vertices.

– **Cscore**: $Cscore$ in this paper is used to assess the weakly connected value of any vertex v not in the current solution S. Given a graph G, for a vertex v not in the current solution S on G, $Cscore(v)$ indicates how many weakly connected components of the weakly induced subgraph of S on G are weakly connected with v. If we add the vertex v not in S into S, $Cscore(v)$ weakly connected components will be weakly connected as a new weakly connected subgraph.

– **Dscore**: $Dscore$ is widely used in the local search algorithms for solving the optimization problems which are required for dominating all vertices [Shin *et al.*, 2010; Wang *et al.*, 2018a; Wang *et al.*, 2017]. For a vertex v, $Dscore(v)$ = $N[S_v]$–$N[S]$, in which S is the current solution and S_v is the solution after changing v's status. From the above definition, $Dscore(v)$ is a positive number when v is added to the cur-

Algorithm 1 *LS*

Input: an initial weakly connected dominating set *W*, and the maximum iteration number *ITEM_NUM*
Output: a better weakly connected dominating set *R*

1: *R*←*W*;
2: **for** *it* = 0; *it* < *ITER_NUM*; *it*++ **do**
3: **while** *W* is a WCDS **do**
4: *R*←*W*;
5: *w*←select *w* from *W* with the greatest *Dscore*, breaking ties randomly;
6: *W*←*W**w*;
7: **end while**
8: *w*←select *w* from *W* with the greatest *Dscore*, breaking ties randomly;
9: *W*←*W**w*;
10: *v*←select *v* from $V - W$ with the greatest $Cscore$ value, breaking ties randomly;
11: *W*←*W*∪*v*;
12: **end for**
13: **return** *R*;

Algorithm 2 *MA*

Input: an undirected and connected graph $G = (V, E)$
Output: a weakly connected dominating set *R*

1: $RS = init\_popolution(G)$;
2: **for** each solution S in RS **do**
3: MWCDS(G, S);
4: $LS(G, S)$;
5: **end for**
6: update R with the best solution in RS;
7: **while** $elasped\_time < cutoof$ **do**
8: $FS = Selection(RS)$;
9: $ARS = FS$;
10: $ARS = ARS \cup Crossover(FS)$;
11: $ARS = ARS \cup Mutation(ARS)$;
12: **for** each solution S in ARS **do**
13: MWCDS(G, R);
14: $LS(G, R)$;
15: **end for**
16: update R with the best solution in ARS;
17: $RS = Create\_new\_population(ARS)$;
18: **end while**
19: **return** R;

rent solution S, and $Dscore(v)$ is a negative number when v is removed from the current solution S.

The core idea of the local search algorithm LS is that it searches a better solution with $k - 1$ vertices after finding a solution with k vertices. In algorithm 1, after a feasible solution W is obtained in line 3, the vertices with greatest $Dscore$ are repeatedly selected and remove from W until W becomes an infeasible solution through the loops in lines 3-7. After executing the loops in lines 3-7, an infeasible solution with $|R - 1|$ vertices is obtained. Then Algorithm 1 intends to search a feasible solution with $|R - 1|$ vertices through exchanging the vertex with the greatest $Dscore$ and the vertex with the greatest $Cscore$ in lines 8-11.

4 The memetic algorithm for MWCDSP

In this section, we will introduce the memetic algorithm for MWCDSP, which is shown as Algorithm 2.

Given a connected and undirected graph $G = (V, E)$, a chromosome of 0-1 sequence for $|V|$ vertices is in correspondence with a MWCDSP solution. For example, the solution with vertex sequence $[a, b, c, d, e, f, g, h, i]$ in Figure 1 is in correspondence with a chromosome $[1, 0, 0, 1, 1, 0, 0, 0, 0]$. In MA, each initial chromosome i.e. solution is generated by randomly assigning 0 or 1 to each vertex in line 1. A vertex v with a random value 1 means that v is selected into the solution, otherwise v is not in the solution. The initial individual S maybe not a feasible solution, so we use MWCDS to transform S into a minimal weakly connected dominating set in line 3, which is further refined by the local search of algorithm 1 in line 4. After the final initial population is identified, MA updates the found best solution R in line 6.

The binary tournament selection is used in line 8. In the crossover of line 10, a two-point crossover is used. In the two-point crossover, two parent chromosomes $Parent_1$ and $Parent_2$ are firstly selected, and two random positions $point_1$, $point_2$ (let $point_1 < point_2$) are chosen from the chromosome as cut points. The values of vertices from $point_1$ to $point_2$ in $Parent_1$ and $Parent_2$ are exchanged with each other, and two children are generated, which will be added into ARS in line 10. For example, Given two parent chromosomes $Parent_1 = [1, 0, 0, 1, 0, 1, 0, 0, 0]$ and $Parent_2 = [0, 0, 1, 0, 0, 1, 0, 0, 0]$, if $point_1 = 3$ and $point_2 = 6$, two children $[1, 0, 1, 0, 0, 1, 0, 0, 0]$ and $[0, 0, 0, 1, 0, 1, 0, 0, 0]$ will be obtained by exchanges the two subsequences from the third position to the sixth position in $Parent_1$ and $Parent_2$ respectively.

The mutation is used to extend ARS in line 11, in which some vertex is randomly selected for reversing its status from each randomly selected chromosome. After executing the crossover and mutation, MA also uses MWCDS and LS to obtain feasible solutions from ARS in lines 12-14. Then, the best solution and new population are updated in lines 16-17.

5 Experimental Evaluation

In this section, we present the experimental results obtained with the proposed memetic algorithm (MA), which will be compared with the famous CPLEX, the self-stabilizing algorithm named MWCDS, the local search algorithm LS in section 3, and the genetic algorithm GA which is constructed by removing LS from MA.

5.1 Benchmark instances

There are three benchmarks selected in experiments, including random UDG benchmarks, LPNMR'09 benchmarks and Common UDG benchmarks, which are described as follows.

• Random UDG benchmarks (24 instances): This group of benchmarks is randomly generated with the method used in [Jovanovic and Tuba, 2013], which is derived from ad hoc network clustering problems. The ad hoc network clustering problems are also used to test MWCDS in [Ding *et al.*, 2016].

Table 1: Comparative results of MA with four competitors on the random UDG graph.

Area(N×N) Nodes	R	CPLEX		MWCDS			GA			LS			MA		
		Solu.	Stat.	Best	Avg	Time	Best	Avg	Time	Best	Avg	Time	Best	Avg	Time
	15	**6**	46.98	**6**	6.6	<0.01	**6**	6	<0.01	**6**	6	<0.01	**6**	6	<0.01
	16	**6**	89.13	**6**	6	<0.01	**6**	6	<0.01	**6**	6	<0.01	**6**	6	<0.01
80	17	**6**	12.37	**6**	6	<0.01	**6**	6	<0.01	**6**	6	<0.01	**6**	6	<0.01
15	18	**6**	8.65	**6**	6	<0.01	**6**	6	<0.01	**6**	6	<0.01	**6**	6	<0.01
	19	**6**	6.67	**6**	6	<0.01	**6**	6	<0.01	**6**	6	<0.01	**6**	6	<0.01
	20	**4**	10.75	5	5.5	<0.01	**4**	4	<0.01	**4**	4	<0.01	**4**	4	<0.01
	20	**8**	2369.32	**8**	8.5	<0.01	**8**	8	<0.01	**8**	8.2	<0.01	**8**	8	<0.01
	21	**6**	1883.2	**6**	6.8	<0.01	**6**	6	<0.01	**6**	6	<0.01	**6**	6	<0.01
100	22	**5**	Feasible	**5**	6.6	<0.01	**5**	5	<0.01	6	6	0.28	**5**	5	0.02
20	23	**6**	2539.74	7	7.4	<0.01	**6**	6	<0.01	7	7	<0.01	**6**	6	0.08
	24	**6**	2598.65	**6**	7.3	<0.01	**6**	6	<0.01	**6**	6	<0.01	**6**	6	<0.01
	25	**5**	Feasible	6	6.4	<0.01	**5**	5	<0.01	**5**	5.8	0.05	**5**	5	<0.01
	20	N/A	N/A	9	9	<0.01	**7**	7	<0.01	8	8.8	0.04	**7**	7	0.06
	21	N/A	N/A	8	8.8	<0.01	**7**	7	<0.01	8	8.2	0.01	**7**	7	0.02
100	22	N/A	N/A	8	8.8	<0.01	**6**	6	<0.01	8	8.5	0.01	**6**	6	0.13
25	23	N/A	N/A	7	8.5	<0.01	**6**	6	<0.01	**6**	6.8	0.02	**6**	6	0.01
	24	N/A	N/A	7	7.8	<0.01	**6**	6	<0.01	**6**	6	<0.01	**6**	6	<0.01
	25	N/A	N/A	8	8.7	<0.01	**7**	7	<0.01	8	8.5	0.67	**7**	7.1	5.72
	25	N/A	N/A	12	12.3	<0.01	**11**	11	<0.01	**11**	11.5	0.01	**11**	11	0.04
	26	N/A	N/A	10	12	<0.01	10	10	<0.01	**9**	10.2	0.02	**9**	9	0.05
150	27	N/A	N/A	**11**	12.6	<0.01	**11**	11	<0.01	**11**	12.2	0.02	**11**	11	9.64
30	28	N/A	N/A	**10**	11.9	<0.01	**10**	10	<0.01	11	11.5	0.01	**10**	10	0.1
	29	N/A	N/A	**9**	10.6	<0.01	**9**	9	<0.01	**9**	10	<0.01	**9**	9	<0.01
	30	N/A	N/A	**9**	10.6	<0.01	**9**	9	<0.01	**9**	10	<0.01	**9**	9	<0.01

• LPNMR'09 benchmarks (9 instances): This group of benchmarks has been used on the Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09), we select the problem instances that have had a satisfactory solution (the solution is known) given in the LPNMR benchmark, just like [Jovanovic and Tuba, 2013].

• Common UDG benchmarks (41 instances): This group of benchmarks was developed by Jovanovic et.al in [Jovanovic and Tuba, 2013], which is derived from ad hoc network clustering problems.

5.2 Experiment preliminaries

In this section, we perform intensive experiments to evaluate our proposed MA. We compare MA with the CPLEX, MWCDS [Ding *et al.*, 2016], LS and GA on 74 benchmark instances.

CPLEX is a general MIP solver, which is widely used to solve many important constrained optimization problems [Sun *et al.*, 2018; Niu *et al.*, 2021; Wang *et al.*, 2013]. The version of CPLEX used to solve the ILP model in subsection 2.2 in our experiments is v12.9.

MWCDS is a linear time self-stabilizing algorithm for solving minimal weakly connected dominating set problem, which terminates in $O(n)$ steps by using a synchronous demon [Ding *et al.*, 2016]. The solutions obtained by MWCDS are partly decided by the random initial solutions, so we run MWCDS 10 times independently for each instance with different random seeds in our experiments.

Our proposed MA, GA and LS are implemented in the C++ programming language. All experiments were performed on the Linux Ubuntu with Intel(R) Core(TM) i9-10900 CPU @2.80GHz and 32GB memory. Each of the above three solvers run 10 times independently for each instance with different random seeds, until the time limit (300 s) is reached. The running time of CPLEX is limited to 3600s, which refers to the cutoff limit of CPLEX in [Sun *et al.*, 2018]. The size of RS and FS, the crossover rate and the mutation rate in MA or GA are set to be 30, 20, 0.9 and 0.1, respectively. The parameter $ITER\_NUM$ (LS in MA) is set to be 100.

For each instance, *Best* denotes the cost of best solution found, *Avg* denotes the average weight of the solution obtained in 10 runs, and *Time* denotes the running time of each solver for obtaining its best solutions. The column is marked as "N/A", when the algorithm fails to provide a solution. If the CPLEX is not terminated within 3600s, but a solution is provided, then the running time is marked as "Feasible".

5.3 Experimental results

Table 1 shows the comparative results of five solvers on the random UDG benchmarks. In Table 1, CPLEX completely solves 10 instances, i.e., the optimal solutions of them are found and proved by CPLEX. CPLEX gives undefined solutions for 2 instances. For the remaining 14 instances, CPLEX can not solve them, since the fourth constraint in formula (2) creates too many restrictions. For the four incomplete algorithms in Table 2, MWCDS, GA, LS and MA obtains minimum best solutions for 13 instances, 23 instances, 17 instances and 24 instances, respectively. Obviously, MA significantly outperforms the competitors on these random UDG benchmarks.

Table 2 shows the comparative results on the LPNMR'09 benchmarks. In Table 2, CPLEX can not solve any instances. For the four incomplete algorithms in Table 2, MWCDS, GA, LS and GA obtain minimum best solutions for 0 instance, 1 instance, 8 instances and 9 instances, respectively. Obviously, MA is still the best solver for MWCDSP.

Table 3 shows the comparative results on the common UDG benchmark. Since the inefficiency of CPLEX and the inaccuracy of MWCDS, we do not list their results in Table 3. In Table 3, 'B.', 'A.' and 'T.' are the abbreviations of 'Best',

Table 2: Comparative results of MA with four competitors on the LPNMR'09 graph.

Ins.	CPLEX		MWCDS			GA			LS			MA		
	Solu.	Stat.	Best	Avg	Time	Best	Avg	Time	Best	Avg	Time	Best	Avg	Time
40×200	N/A	N/A	8	9.4	<0.01	6	6	<0.01	**5**	5.0	<0.01	**5**	5.0	<0.01
45×250	N/A	N/A	8	9.2	<0.01	**5**	5	1.4	**5**	5.0	<0.01	**5**	5.0	<0.01
50×250(1)	N/A	N/A	11	11.9	<0.01	8	8	<0.01	**7**	7.0	<0.01	**7**	7.0	<0.01
50×250(2)	N/A	N/A	9	11.3	<0.01	7	7	<0.01	**6**	6.0	0.04	**6**	6.0	<0.01
55×250	N/A	N/A	10	12.3	<0.01	8	8	0.3	**7**	7.0	<0.01	**7**	7.0	<0.01
60×400	N/A	N/A	10	11	<0.01	7	7	<0.01	**6**	6.0	0.01	**6**	6.0	<0.01
70×250	N/A	N/A	17	19.4	<0.01	13	13	1.2	12	12.8	0.4	**11**	11.0	0.05
80×500	N/A	N/A	14	15.9	<0.01	9	9	35.5	**8**	8.0	0.05	**8**	8.0	0.05
90×600	N/A	N/A	16	17.8	<0.01	10	10.2	76.4	**9**	9.0	0.02	**9**	9.0	<0.01

'Avg' and 'Time' respectively. From the results of Table 3, GA is the worst solver. Comparing to LS, MA can search better solutions on 31 instances. For the remaining 10 instances, LS and MA find the same best solutions. In one word, MA is competitive with the other four solvers for the aspect of the competition of obtained best solutions.

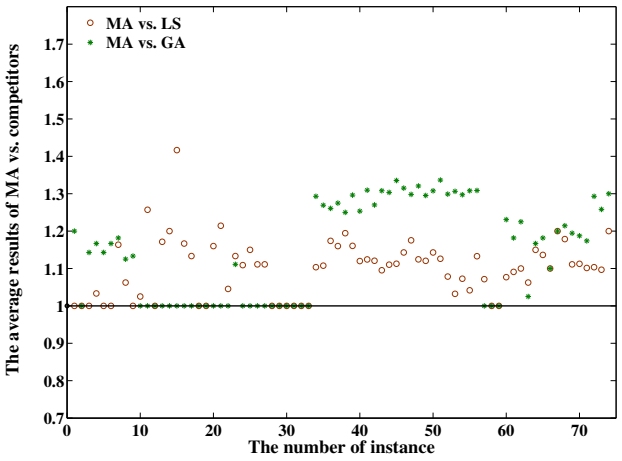

Figure 3: The comparison of 'AVG' valuses of MA, GA and LS.

On the other hand, The competitive results of 'AVG' values in Tables 1-3 for MA vs. competitors (LS and GA are considered) are shown in Figure 2. The values in x-axis are the numbers of instances. The values in y-axis are computed by AVG(competitor) /AVG(MA). From Figure 2, MA also significantly outperforms LS and GA for the 'AVG' values.

From the experimental results, MA shows some edge over LS and GA. The main reason is that, LS uses greedy approaches and so suffers from diversity, while MA with crossover could bring more diversity. The solution quality of GA mainly relies on the minimal WCDSs obtained by MWCDS which are usually not accurate enough for most complexity graphs, while MA with local search procedure could further improve the local optimal solution obtained by MWCDS.

6 Conclusion

In this paper, a 0-1 integer linear programming model is designed for the minimum weakly connected dominating set problem (MWCDSP), and we also introduce the framework of memetic algorithm (MA) for MWCDSP, which comprises of initializing population, crossover, mutation. In our MA, a linear time self-stabilizing algorithm MWCDS is used to transform infeasible solutions into minimal feasible solutions. We also introduce a local search algorithm which is used to refine the feasible solutions obtained by MWCDS, in which two score functions are designed for MWCDSP. Experimental results on three types of test problems suggest that our MA can solve MWCDSP efficiently.

In the future, we would like to further improve the memetic algorithm for MWCDSP by some other ideas, such as configuration checking, dynamic connectivity maintenance etc., and try to solve other massive instances.

Table 3: Comparative results of MA with two incomplete competitors on the common UDG graph.

Area(N×N) Nodes	R	GA			LS			MA		
		B.	A.	T.	B.	A.	T.	B.	A.	T.
400 80	60	16	16	0.3	14	14	<0.01	**13**	13	1.3
	70	13	13	1.6	**11**	12	<0.01	**11**	11	<0.01
	80	9	9.8	62.2	**8**	8.8	<0.01	**8**	8	<0.01
	90	**8**	8.2	103.9	**8**	8.5	<0.01	**8**	8	<0.01
	100	**7**	7	0.4	**7**	7.5	<0.01	**7**	7	<0.01
	110	**6**	6	100.9	**6**	6	<0.01	**6**	6	<0.01
	120	**5**	5	0.2	**5**	5	<0.01	**5**	5	<0.01
600 100	80	18	18	38.7	17	18	<0.01	**15**	15	3.4
	90	17	17	48.6	16	16.5	<0.01	**14**	14	17.9
	100	14	14	28.1	13	13.8	<0.01	**12**	12	0.1
	110	13	13	5.9	**11**	12.5	<0.01	**11**	11	0.2
	120	11	11	23.3	**10**	11	<0.01	**10**	10	<0.01
700 200	70	37	37.5	83.2	31	32	<0.01	**29**	29	85.1
	80	31	31.2	84.3	26	27.2	<0.01	**24**	24.8	21.4
	90	26	26	147.1	23	24	<0.01	**20**	20	6
	100	21	21.5	11	19	20	<0.01	**18**	18	22
	110	19	19	33.1	**16**	17.8	<0.01	**16**	16	0.3
	120	16	16.2	120.7	14	15.2	0.1	**13**	13.8	39.6
1000 200	100	37	37.5	70.5	31	32	<0.01	**29**	29	74.1
	110	33	33	40.7	28	28.8	<0.01	**26**	26	7
	120	29	29	41.8	25	27	<0.01	**23**	23	23.4
	130	25	25.5	28	23	23.2	<0.01	**20**	20	2.6
	140	22	22.5	72.9	20	21.5	<0.01	**18**	18	84.4
	150	21	21	15.4	18	18.8	<0.01	**16**	16.2	97.8
	160	18	18.8	29.3	16	16.8	<0.01	**15**	15	24.1
1500 250	130	49	49.5	67.3	41	42.5	<0.01	**37**	37.8	62.5
	140	44	44.2	99.6	38	39	<0.01	**34**	34.8	49.1
	150	41	41.2	87.6	34	34.5	<0.01	**31**	31.5	99
	160	37	37.8	35.2	30	32.2	<0.01	**29**	29	33.7
2000 300	200	43	43.8	48.9	35	36.5	<0.01	**32**	32.8	24.1
	210	40	40.5	84	34	35.2	<0.01	**30**	30.8	44.5
	220	37	37	136.9	31	33.5	<0.01	**28**	28.5	97.7
	230	35	35	33.7	29	29.8	<0.01	**26**	26.5	80.6
2500 350	200	64	64.5	136	51	55.8	<0.01	**49**	49.8	26
	210	59	59.5	134.4	50	52	<0.01	**44**	45.5	135.5
	220	57	57.2	51.2	46	48.2	<0.01	**42**	42.8	101
	230	52	53	183.5	43	44	<0.01	**40**	40.8	68.4
3000 400	210	80	81	114.3	**61**	64	0.3	**61**	62	59.9
	220	76	76.8	116.4	62	63.5	<0.01	**59**	59.2	88.1
	230	72	72.2	79.6	56	57.5	0.1	**54**	55.2	79.3
	240	66	67	94.6	55	58	0.1	**51**	51.2	149.5

References

[Bonamy *et al.*, 2021] Marthe Bonamy, Linda Cook, Carla Groenland, and Alexandra Wesolek. A tight local algorithm for the minimum dominating set problem in outerplanar graphs. *arXiv preprint arXiv:2108.02697*, 2021.

[Ding *et al.*, 2016] Yihua Ding, James Z Wang, and Pradip K Srimani. A linear time self-stabilizing algorithm for minimal weakly connected dominating sets. *International Journal of Parallel Programming*, 44(1):151–162, 2016.

[Dokeroglu and Sevinc, 2021] Tansel Dokeroglu and Ender Sevinc. Memetic teaching–learning-based optimization algorithms for large graph coloring problems. *Engineering Applications of Artificial Intelligence*, 102:104282, 2021.

[Domke *et al.*, 2005] Gayla S Domke, Johannes H Hattingh, and Lisa R Markus. On weakly connected domination in graphs ii. *Discrete Mathematics*, 305(1-3):112–122, 2005.

[Du *et al.*, 2012] Hongjie Du, Weili Wu, Shan Shan, Donghyun Kim, and Wonjun Lee. Constructing weakly connected dominating set for secure clustering in distributed sensor network. *Journal of combinatorial optimization*, 23(2):301–307, 2012.

[Dunbar *et al.*, 1997] Jean E Dunbar, Jerrold W Grossman, Johannes H Hattingh, Stephen T Hedetniemi, and Alice A McRae. On weakly connected domination in graphs. *Discrete Mathematics*, 167:261–269, 1997.

[Han and Jia, 2007] Bo Han and Weijia Jia. Clustering wireless ad hoc networks with weakly connected dominating set. *Journal of Parallel and Distributed Computing*, 67(6):727–737, 2007.

[Jovanovic and Tuba, 2013] Raka Jovanovic and Milan Tuba. Ant colony optimization algorithm with pheromone correction strategy for the minimum connected dominating set problem. *Computer Science and Information Systems*, 10(1):133–149, 2013.

[Kamei and Kakugawa, 2007] Sayaka Kamei and Hirotsugu Kakugawa. A self-stabilizing approximation algorithm for the minimum weakly connected dominating set with safe convergence. In *Proceedings of the 1st International Workshop on Reliability, Availability, and Security*, pages 57–66, 2007.

[Lemańska and Patyk, 2008] Magdalena Lemańska and Agnieszka Patyk. Weakly connected domination critical graphs. *Opuscula Mathematica*, 28(3):325–330, 2008.

[Moalic and Gondran, 2019] Laurent Moalic and Alexandre Gondran. The sum coloring problem: a memetic algorithm based on two individuals. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 1798–1805. IEEE, 2019.

[Niu *et al.*, 2021] Dangdang Niu, Bin Liu, and Minghao Yin. Local search for weighted sum coloring problem. *Applied Soft Computing*, 106:107290, 2021.

[Pathan and Seon, 2006] Khan Pathan and Choong Seon. A key-predistribution-based weakly connected dominating set for secure clustering in dsn. *Springer-Verlag*, 2006.

[Raczek and Cyman, 2019] Joanna Raczek and Joanna Cyman. Weakly connected roman domination in graphs. *Discrete Applied Mathematics*, 267:151–159, 2019.

[Shin *et al.*, 2010] Incheol Shin, Yilin Shen, and My T Thai. On approximation of dominating tree in wireless sensor networks. *Optimization Letters*, 4(3):393–403, 2010.

[Srimani and Xu, 2007] Pradip K Srimani and Zhenyu Xu. Self-stabilizing algorithms of constructing spanning tree and weakly connected minimal dominating set. In *27th International Conference on Distributed Computing Systems Workshops (ICDCSW'07)*, pages 3–3. IEEE, 2007.

[Sun *et al.*, 2018] Wen Sun, Jin-Kao Hao, Xiangjing Lai, and Qinghua Wu. Adaptive feasible and infeasible tabu search for weighted vertex coloring. *Information Sciences*, 466:203–219, 2018.

[Wang *et al.*, 2013] Yang Wang, Jin-Kao Hao, Fred Glover, and Zhipeng Lü. Solving the minimum sum coloring problem via binary quadratic programming. *arXiv preprint arXiv:1304.5876*, 2013.

[Wang *et al.*, 2017] Yiyuan Wang, Shaowei Cai, and Minghao Yin. Local search for minimum weight dominating set with two-level configuration checking and frequency based scoring function. *Journal of Artificial Intelligence Research*, 58:267–295, 2017.

[Wang *et al.*, 2018a] Yiyuan Wang, Shaowei Cai, Jiejiang Chen, and Minghao Yin. A fast local search algorithm for minimum weight dominating set problem on massive graphs. In *IJCAI*, pages 1514–1522, 2018.

[Wang *et al.*, 2018b] Yiyuan Wang, Jiejiang Chen, Huanyao Sun, and Minghao Yin. A memetic algorithm for minimum independent dominating set problem. *Neural Computing and Applications*, 30(8):2519–2529, 2018.

[Wu *et al.*, 2022] Xinyun Wu, Zhipeng Lü, and Fred Glover. A fast vertex weighting-based local search for finding minimum connected dominating sets. *INFORMS Journal on Computing*, 34(2):817–833, 2022.

[Yu *et al.*, 2012] Jiguo Yu, Nannan Wang, and Guanghui Wang. Constructing minimum extended weakly-connected dominating sets for clustering in ad hoc networks. *Journal of Parallel and Distributed Computing*, 72(1):35–47, 2012.