Received 2 October 2022, accepted 27 October 2022, date of publication 4 November 2022, date of current version 9 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3219484

RESEARCH ARTICLE

Two Heuristic Algorithms for the Minimum Weighted Connected Vertex Cover Problem Under Greedy Strategy

QIPENG XIE, YUCHAO LI, SENGUI HU, YUE ZHU, AND HONGQIANG WANG
School of Science, Chang'an University, Xi'an, Shaanxi 710064, China

Corresponding author: Yuchao Li (liyuchao@chd.edu.cn)

This work was supported in part by the National Training Program of Innovation and Entrepreneurship for Undergraduates under Grant S202010710164, in part by the Natural Science Basic Research Program of Shaanxi under Grant 2021JQ-219, and in part by the Fundamental Research Funds for the Central Universities Chang'an University in Chinese (CHD) under Grant 300102121104.

ABSTRACT The *Minimum Weighted Connected Vertex Cover problem*(MWCVC) is to find a subset $F \subset V(G)$ with minimum weight in a node-weighted graph G, such that when removing the set F, the inducing graph of remaining vertices holds no edges, and the graph induced from set F in G is required to be connected. This problem comes from the classical combinatorial problem in graph theory, i.e., the Vertex Cover Problem. A large number of results on algorithms for the MWCVC problem have been reported. In this paper, we proposed two heuristic algorithms, denoted as VCC and LCVCC, to find a connected vertex cover set in a general weighted graph. The time complexity of both two algorithms are less than $O(n^4)$. We compare these two algorithms with two known heuristic algorithms GR and GD (proposed by Dagdeviren in 2021) on connected graphs, and draw a conclusion that both of VCC and LCVCC perform better than GR or GD. Relatively speaking, LCVCC is expected to have better performance in dense graphs than VCC.

INDEX TERMS Minimum weighted connected vertex cover problem, heuristic algorithm, greedy strategy, algorithm performance.

I. INTRODUCTION

All the graphs we considered are simple, without loops and multi-edges. Given a graph G, we denote the vertex set and edge set of G by $V(G)$ and $E(G)$. For $v \in G$, we use $d(v)$ to represent the degree of vertex v. Δ is used to denote the maximum degree and δ is used to denote the minimum degree of the vertex in G. If $S \subset V$, then $G[S]$ is used to represent the graph induced from S, and $E(G[S])$ is the edge set of $G[S]$. For a vertex $v \in V$, we use $N(v)$ to represent the neighbor vertex set of v and for a vertex set $S \subset V$, we define its neighbor by $N(S) = \bigcup_{v \in S} N(v) \setminus S$.

The *Minimum Vertex Cover problem* (MVC) is to find a subset VC of $V(G)$ as small as possible and the inducing graph of remaining vertices holds no edges when removing the VC set from the graph, i.e. $E(G[V(G) \setminus VC]) = 0$. Moreover, if the VC set is required to be connected in the original graph, which means a path at least can be found between any two vertices of the VC set, the problem becomes the *Minimum Connected Vertex Cover problem* (MCVC), which was first introduced by Garey and Jonson in 1976 [4] and is NP-hard to be approximated within $10\sqrt{5} - 21$ [3]. We use CVC to represent the subset we select as the solution. Both of the problem above are applied in Wireless sensor networks (WSNs), signal station construction, terminal connection and resources transportation by pipeline, etc.

A more complex form of MVC problem is to give weights for every vertex in graph. This problem has a name of *Minimum weighted vertex cover problem* (MWVC) [5]. In this situation, denoting the vertex cover set as F, if the induced graph $G[F]$ is connected, the set F is a solution for *Minimum Weighted Vertex Cover problem* (MWCVC). Fujito proved that for any ε, MWCVC can not be approximated within $(1-\varepsilon)\ln n$ unless $NP \subset DTITM(n^{O(\log\log n)})$ [6]. Shimizu et al. in 2016 give a heuristic algorithm for MWVC problem on undirected weighted graph [7]. For MCVC

The associate editor coordinating the review of this manuscript and approving it for publication was Sun-Yuan Hsieh.

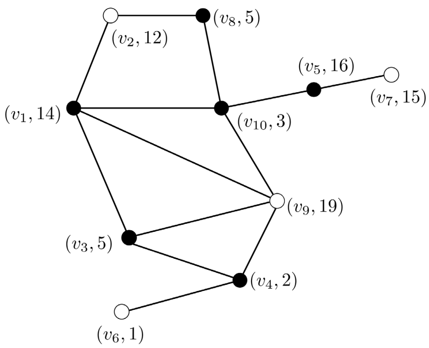


FIGURE 1. **Example for algorithm GR.**

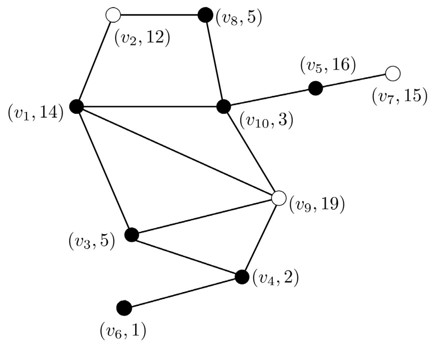


FIGURE 2. **Example for algorithm GD.**

problem, Zhang et al. propose a two stage algorithm that a greedy algorithm and a configuration checking method are used [8]. Dagdeviren gives a Hybrid genetic algorithm in 2021 to solve MWCVC problem [1].

In section 2, we introduced two known heuristic algorithms and propose two new algorithms for MWCVC problem. We also provide examples for those four algorithms. In section 3, we investigate the performance of these algorithms from the different aspects such as the number of vertices, the weight of selected vertices and cost time of algorithm. Section 4 is a short conclusion and our future work.

II. TWO PROPOSED HEURISTIC ALGORITHM

In this section, we first introduce two simple greedy heuristic algorithms GR and GD proposed in 2021 by Dagdeviren [1]. Secondly, we propose two heuristic algorithmss for MWCVC also under greedy strategy.

A. TWO KNOWN HEURISTIC ALGORITHMS FOR MWCVC PROBLEM

Both GR and GD are two stage algorithms and based on the algorithms for solving weighted connected dominating set problem in [2]. In each algorithm, all vertices are initially WHITE. When an vertex is selected into CVC set, it turns BLACK and all of its neighbors are colored GRAY. These two algorithms both are started with a vertex, and then select GRAY vertices by some greedy strategy in the first stage. When there is no WHITE vertices, choose additional GRAY vertices having the smallest weight until all edges are covered. The difference between them is the greedy strategy. GR chooses the vertex among GRAY vertices with minimum ratio $r(v) = \frac{w(v)}{\sum\limits_{u \in N[v]} w(u)}$, while GD choose the vertex which have the most WHITE neighbors, i.e. the maximum $d(v, WHITE) = |N(v, WHITE)|$. If two vertices have the same ratio or same number of WHITE neighbors, just choose the vertex with the smaller weight.

A simple connected weighted graph with 10 vertices is used to show how these two algorithms work, see Figure 1. (v_i, w_i) denotes the vertex v_i and its weight. GR first chooses v_{10}, since its rate $\frac{3}{3+5+19+16+14} = 0.0526$ is the minimum one among the all nodes. Then among all neighbors of v_{10}, v_1 has the minimum rate $\frac{14}{50} = \frac{7}{25}$. After that v_5, v_9, v_4 and v_1 are selected step by step. Finally algorithm stops, because $N(S)$ are all isolated vertices, that means $S = \{v_1, v_3, v_4, v_5, v_8, v_{10}\}$ has been a connected vertex cover set already.

When executing GD, see Figure 2, v_1, v_4, v_{10} all have the largest degree but v_{10} has the minimum weight, so v_{10} is the first one to be selected. At the second step, v_1 is selected because it has two neighbors v_2 and v_5, and its weight is less than v_9. Then GD chooses v_3, v_4 and v_5. Then in the second stage, choose v_6 and v_8. In this example, GD chooses $\{v_1, v_3, v_4, v_5, v_6, v_8, v_{10}\}$ that forms a connected vertex cover set.

B. THE FIRST PROPOSED HEURISTIC ALGORITHM VCC

We propose a two-stage heuristic *Vertex Cover and Connectivity* (VCC) algorithm to compute a connected vertex cover set with relatively minimum weight by finding a vertex cover set first and selecting more vertices into this set to ensure its connectivity. In the first stage, the algorithm selects the most cost-efficient vertex iteratively until all the edges has been covered. Let S be the selected vertex set the cost-effectiveness of vertex u is defined as $p(u) = \frac{w(u)}{d_{G[V \setminus S]}(u)}$, so VCC always chooses the vertex with the smallest value of cost-effectiveness. At the end of the first stage, the selected vertex set S is a vertex cover set, VCC will then find the most cost-efficient vertices to ensure the connectivity of S in the second stage. We use $\kappa(S)$ to represent the number of components of the graph $G[S]$. In this case, the cost-effectiveness is defined as $p(u) = \frac{w(u)}{\kappa(S) - \kappa(S \cup \{u\})}$. The algorithm is described as Algorithm 1.

Figure 3 gives an example for VCC algorithm. v_{12} is the first vertex to be selected because its cost-effectiveness is 0.25 that is smallest among all vertices' cost-effectiveness. After v_{12} is selected, all edges incident to it are covered. Then algorithm updates the cost-effectiveness and selects v_5 (or v_{11}). Repeat this operation until all edges are covered. In this graph, the vertex cover set selected in the first stage of this greedy algorithm also is a connected vertex cover set, i.e. $\kappa(S) = 1$. So, VCC does not need to select more vertices in the second stage, which means the second *while* loop does not work on this graph. The solution given by VCC algorithm is $\{v_{12}, v_5, v_{11}, v_4, v_{14}, v_2, v_{15}, v_6, v_{13}, v_{10}, v_1\}$, and the total weight is 109.

Algorithm 1 VCC Algorithm

Require: A node weighted graph $G = (V, E)$, $w : V \to R^+$
Ensure: A connected vertex cover set S

$S = \emptyset$
while $E(G[V \setminus S]) \neq \emptyset$ **do**
  Find a vertex u with $\min_{v \in V \setminus S} p(u)$, i.e. $\min_{v \in V \setminus S} \frac{w(u)}{d_{G[V \setminus S]}(u)}$, if two vertices are same, then select any one of them.
  $S = S \cup \{u\}$
end while
while $\kappa(S) \neq 1$ **do**
  Find a vertex u with $\min_{v \in V \setminus S} \frac{w(u)}{\kappa(S) - \kappa(S \cup \{u\})}$
  $S = S \cup \{u\}$
end while
Output: S

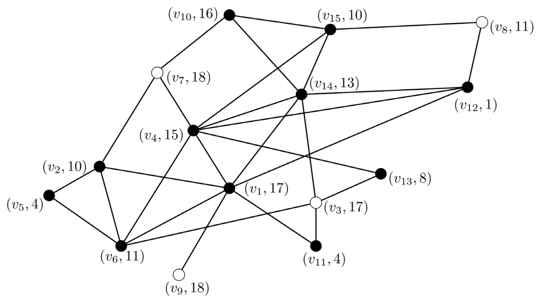


FIGURE 3. Example for algorithm VCC.

The time complexity of VCC algorithm is equal to $O(n^2)$, where n is the number of vertices. In the worst situation for the first *while* loop, it may run $n + (n-1) + (n-2) + \cdots + 2 + 1 = O(n^2)$ times. For the second *while*, it runs $\frac{n}{2} + \frac{n-1}{2} + \cdots = O(n^2)$ times in the worst situation. In fact, these two situations can not happen together, because the more vertices selected in the first *while* loop, the better the connectivity of S will be, which makes the second *while* loop selects less vertices.

C. THE SECOND PROPOSED HEURISTIC ALGORITHM LCVCC

VCC Algorithm is a simple greedy algorithm that hardly gives an optimal solution, especially when the number of vertex is large. We design *Local Connected Vertex Cover and Connectivity* (LCVCC) algorithm is designed to improve the VCC algorithm. The main idea is to find a local connected vertex cover set for a part of graph by Algorithm 2 rather than a select single vertex in every iteration. It begins with a labelled vertex u. If not all vertices in $N(u)$ are isolated in $G[N(u)]$, LCVCC algorithm finds a vertex cover set in $G[N(u)]$ and labels the vertices in the set. Then iteratively update the labeled vertex set by adding these new labeled vertices into the previous vertex cover set. The algorithm searches the neighbors of this labeled vertex set again and then labels more vertices, until the neighbor set forms an independent set. After that, LCVCC searches for more vertices to ensure the connectivity of labeled vertices under the same strategy as VCC, until the labeled vertex set forms a connected vertex cover set. We show how to find a (local) connected vertex cover set for a labeled set.

Algorithm 2 Algorithm to Find a Local Connected Vertex Cover Set

Require: A node weighted graph $G = (V, E)$, $w : V \to R^+$, a labeled vertex set L_I
Ensure: A labeled vertex set L

$S = L_I$, $S' = \emptyset$
while $E(G[N(S)]) \neq \emptyset$ **do**
  while $E(G[N(S) \setminus S']) \neq \emptyset$ **do**
    Find a vertex v with $\min_{v \in N(u) \setminus S} \frac{w(v)}{d_{G[N(S)]}(v)}$
    $S' = S' \cup \{v\}$
  end while
  $S = S \cup S'$
end while
Output: $L = S$

For Algorithm 2, we have following lemmas.

Lemma 1: The labeled vertex set L computed by Algorithm 2 is a connected vertex cover set for the graph $G[L \cup N[L]]$.

Proof: Since the loop in Algorithm 2 stops only when there is no edge between neighbor vertices of labeled set. The labeled set L obviously is a vertex cover set for graph $G[L \cup N(L)]$. Furthermore, all of labeled vertices are neighbors of the labeled set in the previous iteration, which leads to that at least one path can be found between a labeled vertex and the first given vertex set L_I. For any two of labeled vertices, we can use two such kind of paths to link them up, which means the labeled set is not only a vertex cover set, but also a connected vertex cover set.

Lemma 2: The time complexity of Algorithm 2 is less than $O((|L| + |N(L)| - |L_I|)^2)$.

Proof: We denote by N_i the neighbor vertices searched in the ith iteration, $i = 1, 2, \cdots, s$. With the property of greedy algorithm, we know the time complexity of the ith iteration is $O(|N_i|^2)$. So the entire costed time is $O(\sum_{i=1}^{s} |N_i|^2)$, and we have

$$O(\sum_{i=1}^{s} |N_i|^2) < O((\sum_{i=1}^{s} |N_i|)^2) = O((|L| + |N(L)| - |L_I|)^2). \tag{1}$$

Then we propose LCVCC algorithm (Algorithm 3) to compute a connected vertex cover set in graph by using Algorithm 2.

Here we run LCVCC on the example, in Figure 4. First compute all vertices' significant value and v_1 is the first one to be choose. When taking v_1 as the first initial vertex, algorithm 2 labels it, then searches its neighbor. Thus, $N_1 = \{v_6, v_2, v_4, v_{14}, v_{12}, v_{11}\}$ and a vertex cover set of $\{v_{12}, v_6, v_{14}\}$ is picked out. Then algorithm labels those vertices and searches the new neighbor set $N_2 = \{v_5, v_2, v_4, v_9, v_{10}, v_{15}, v_8, v_3\}$ and $\{v_{15}, v_5, v_{11}\}$ are labeled. Then all the neighbor of labeled vertex set are disjoint.

Algorithm 3 LCVCC Algorithm

Require: A node weighted connected graph $G = (V, E)$, $w : V \to R^+$

Ensure: A connected vertex cover set S

$S = \emptyset$
while $E(G[V \setminus S]) \neq \emptyset$ **do**
  $G' = G[V \setminus S]$
  For every vertex $v \in V(G')$, use Algorithm 2 to compute out corresponding labeled vertex set L_v
  Find vertex u, which has the smallest value of $\frac{w(L_u)}{\sum_{v \in L_u} d(v)}$
  $S = S \cup L_u$
end while
while $\kappa(S) \neq 1$ **do**
  Find vertex $u = arg\ min\{u \in V \setminus S \mid \frac{w(u)}{\kappa(S) - \kappa(S \cup u)}\}$
  $S = S \cup u$
end while
Output S

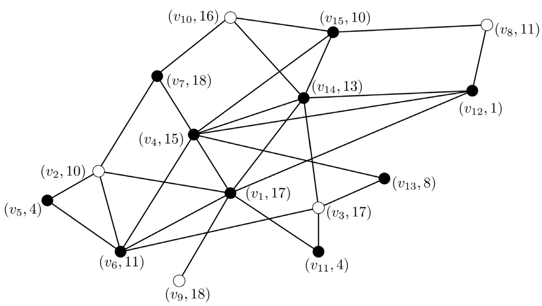


FIGURE 4. Example for algorithm LCVCC.

So the algorithm computes the significant value again and selects the next initial v_{13}, so is the v_7. Then the algorithm gets a vertex cover set $\{v_1, v_{12}, v_6, v_{14}, v_{15}, v_5, v_{11}, v_{13}, v_7\}$, then the algorithm selects v_{14} to ensure the connectivity and outputs the solution $\{v_1, v_{12}, v_6, v_{14}, v_{15}, v_5, v_{11}, v_{13}, v_7, v_4\}$ with total weight 101.

Theorem 3: The vertex set S given by LCVCC algorithm is a connected vertex cover set.

Proof: Assume, for contradiction, that S is a solution given by algorithm LCVCC, but not a vertex cover set for G. Then there must be an edge in $G[V \setminus S]$, in which case additional vertices in $V \setminus S$ will be selected, contradicting the fact that S is a solution given by algorithm LCVCC. Likewise, the second *while* loop ensures the connectivity of the solution S.

Theorem 4: The time complexity of LCVCC algorithm is less than $O(n^4)$.

Proof: We assume the first *while* loop in LCVCC algorithm runs s times in total, P^i is used to represent the vertex set labeled in ith iteration. To find the specific initial vertex in ith iteration, the algorithm runs on every vertex whose corresponding labeled vertex set have the size of $|P^i_j|$. By using lemma 2 we have their cost time is $O(|P^i_j|^2)$, where $j = 1, 2, \cdots, n - \sum_{k=1}^{i-1} |P_k|$, representing the remaining vertices that the algorithm has to operate on. Thus, when selecting the ith initial vertex, the time cost is

$$\sum_j O(|P^i_j|^2) < n \cdot O(max_j\{|P^i_j|^2\}). \tag{2}$$

Then we have the entire time cost of the first *while* loop is less than

$$\begin{aligned} &\sum_i n \cdot O(max_j\{|P^i_j|^2\}) \\ &< n \cdot O(\sum_i max_j\{|P^i_j|^2\}) \\ &< n \cdot O(n \cdot n^2) \\ &= O(n^4). \end{aligned} \tag{3}$$

Because in any iteration, the amount of remaining vertices is less than n, and in each iteration at least one vertex is labeled resulting $i < n$.

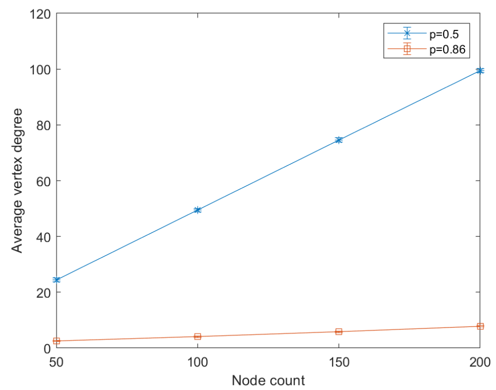


FIGURE 5. Average degree of graphs of different order.

III. PERFORMANCE OF ALGORITHMS

Those four algorithms are implemented in MATLAB to test their performance. The used graphs are undirected and connected, with different scales. We compare these four algorithms (GR, GD, VCC and LCVCC) on three aspects: the number of vertices in graph, the weight of the connected vertex set selected and time cost. For every plotted point, we test the algorithm for 100 times and use mathematical expectation as the value and the variance as the error to compare their stability.

The graphs are randomly generated. For example, if we need a graph of order n, we first generate n vertices. Then, for arbitrary two vertices, we generate a random number between 0 and 1. If the random number is larger than a given number p, then we add an edge between those two vertices. Obviously, the smaller the p is, the denser the graphs are.

The weight of the vertices are random number between 0 and 1. Figure 5 shows the average degree of the graph with 50, 100, 150 and 200 vertices when p is 0.5 or 0.86. From Figure 5 we know the average degree increases linearly with n, and p affects the slope of the line.

As n (the order of graph) increases, the number and the total weight of the selected vertices also increase. When $p = 0.86$, denoted the number of the selected vertices by ST, it shows that $ST_{GR} = ST_{GD} > ST_{VCC} = ST_{LCVCC}$ considering the

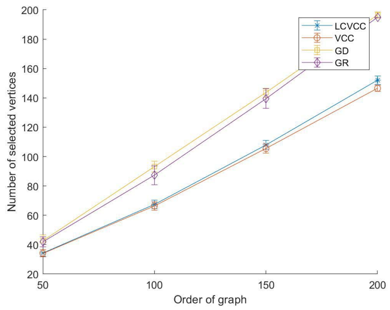


FIGURE 6. Number of selected vertices against order when $p = 0.86$.

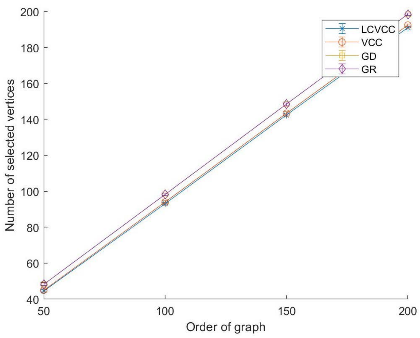


FIGURE 7. Number of selected vertices against order when $p = 0.5$.

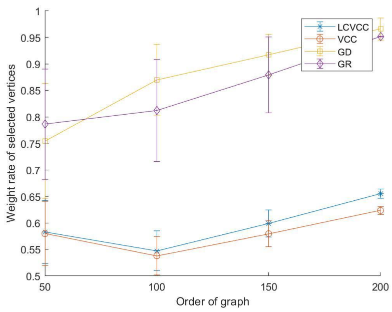


FIGURE 8. Number of selected vertices against order when $p = 0.86$.

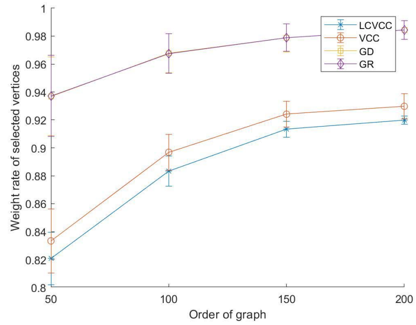


FIGURE 9. Weight rate of selected vertices against order when $p = 0.5$.

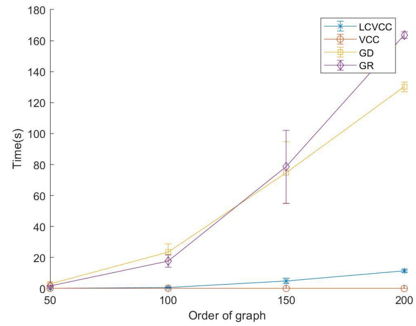


FIGURE 10. Time cost when $p = 0.86$.

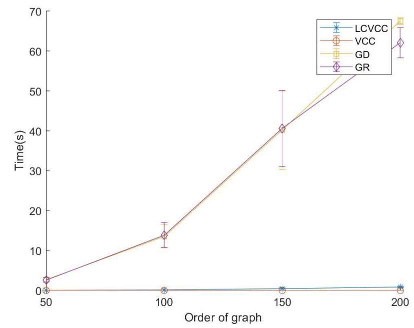


FIGURE 11. Time cost when $p = 0.5$.

error bar, see Figure 6. If $p = 0.5$, despite the number of selected vertices by four algorithms are very close, it holds that $ST_{GR} = ST_{GD} > ST_{VCC} > ST_{LCVCC}$, see Figure 7.

The selected weight rate WR is defined as $\frac{\sum_{v \in S} w(v)}{\sum_{v \in V} w(v)}$, here S is the connected vertex set selected by algorithm. $WR = 1$ means the algorithm selected the all vertices. When $p = 0.86$, see Figure 8, it holds that $WR_{GD} > WR_{GR} > WR_{LCVCC} = WR_{VCC}$. When $p = 0.5$, Figure 9 shows that $WR_{GD} = WR_{GR} > WR_{VCC} > WR_{LCVCC}$. By comparing Figure 8 and 9, it can be concluded that VCC and LCVCC performs better than GD or GR, and LCVCC is expected to have better performance in dense graphs than VCC. Notice that the error bar is smaller, which means the solution is stable.

As for the time cost, VCC is the fastest algorithm among these four algorithms, see Figure 10 and Figure 11. It can be also seen LCVCC performs much better in dense graphs than other algorithms.

We also investigated the performance of these algorithms on random graphs, Cartesian product graphs, Strong product graphs, Interval graphs, Unit disk graphs and Kneser graphs. The definitions of the last five graphs are:

- The *Cartesian product graph* $G \square H$ of graphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$ is a graph with vertex set $V(G) \times V(H)$ such that any two vertices (u, v) and (x, y) are adjacent if and only if $u = x$ and $vy \in E(H)$ or $ux \in E(G)$ and $v = y$.
- The *Strong product graph* $G \boxtimes H$ of graphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$ is a graph with vertex set $V(G) \times V(H)$ such that any two vertices (u, v) and (x, y) are adjacent if and only if $u = x$ and $vy \in E(H)$, or $ux \in E(G)$ and $v = y$, or $ux \in E(G)$ and $vy \in E(H)$.
- Given a set of intervals on the real line, an *interval graph* is an undirected graph in which a vertex for each interval and an edge between vertices whose intervals intersect.
- A *unit disk graph* is the intersection graph of a family of unit disks in the Euclidean plane.
- The *Kneser graph* $K(n, k)$ is the graph whose vertices correspond to the k-element subsets of a set of n

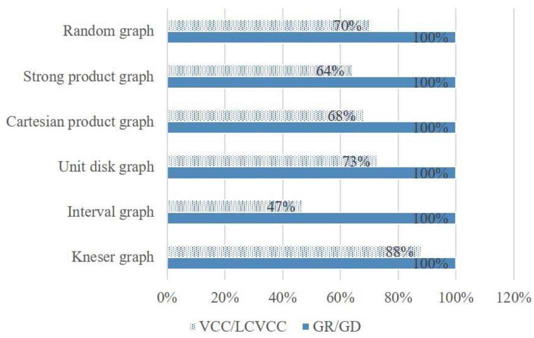


FIGURE 12. **Conclusion of numerical experiment.**

elements, and where two vertices are adjacent if and only if the two corresponding sets are disjoint. The Kneser graph $K(5, 2)$ is isomorphic to the Petersen graph.

In our test, for Cartesian product graph and Strong product graph, we focus on the situation when graph G and H are both paths. Figure 12 shows the weight of vertex set selected by VCC and LCVCC compared with GR and GD on random graphs and five special graphs. We can see that VCC and LCVCC reduce the weight of selected vertex set by 20% to 50% compare with GR and GD.

IV. CONCLUSION AND FUTURE WORK

In this paper, we introduce two heuristic algorithms (GR and GD), and propose two new heuristic algorithms (VCC and LCVCC) to solve MWCVC problem, and then compare their performances. Algorithm VCC and LCVCC are expected to have much better performance. In sparse graphs, algorithm VCC and LCVCC performance similar, but much better than GR or GD, and VCC costs the minimum time among these algorithms. In dense graphs, algorithm LCVCC has the best performance. In our test, VCC and LCVCC always give better solutions than GR or GD. We would like to explore more to figure out if better solutions given by VCC and LCVCC all the way.

To improve the performance of LCVCC on sparse graphs, we would like to try different strategy to choose the initial vertex set and the cost-effectiveness function.

REFERENCES

[1] Z. A. Dagdeviren, "Weighted connected vertex cover based energy-efficient link monitoring for wireless sensor networks towards secure Internet of Things," *IEEE Access*, vol. 9, pp. 10107–10119, 2021.

[2] Z. A. Dagdeviren, D. Aydin, and M. Cinsdikici, "Two population-based optimization algorithms for minimum weight connected dominating set problem," *Appl. Soft Comput.*, vol. 59, pp. 644–658, Jun. 2017.

[3] B. Escoffier, L. Gourvès, and J. Monnot, "Complexity and approximation results for the connected vertex cover problem in graphs and hypergraphs," *J. Discrete Algorithms*, vol. 8, no. 1, pp. 36–49, Mar. 2010.

[4] M. R. Garey and D. S. Johnson, "The rectilinear Steiner tree problem is NP-complete," *SIAM J. Appl. Math.*, vol. 32, no. 4, pp. 826–834, 1977.

[5] M. R. Garey and S. J. David, *Computers and Intractability*. New York, NY, USA: W.H. Freeman, 1979.

[6] T. Fujito, "On approximability of the independent/connected edge dominating set problems," *Inf. Process. Lett.*, vol. 79, no. 6, pp. 261–266, Sep. 2001.

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

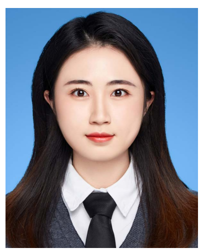
[8] Y. Zhang, J. Wu, L. Zhang, P. Zhao, J. Zhou, and M. Yin, "An efficient heuristic algorithm for solving connected vertex cover problem," *Math. Problems Eng.*, vol. 2018, pp. 1–10, Sep. 2018.

QIPENG XIE was born in Zhejiang, China, in 2001. He is currently a Senior in information and computing science with Chang'an University. He participated in the National Training Program of Innovation and Entrepreneurship for Undergraduates, in 2020, and the Fundamental Research Funds for the Central Universities CHD, in 2021. His research interests include algorithms in graph theory and numerical solutions of nonlinear partial differential equations.

YUCHAO LI was born in Xinjiang, China, in 1988. She received the B.A. degree in information and computing science and the M.S. degree in applied mathematics from the Beijing University of Chemical Technology, China, in 2011 and 2014, respectively, and the Ph.D. degree in computation mathematics from Xi'an Jiaotong University, China, in 2019.

She joined Chang'an University, in 2019, as an Assistant Professor, with a focus on algorithms for cover problems in graph theory.

SENGUI HU was born in Fujian, China, in 2001. He majors in information and computing science with Chang'an University. He participated in the National Training Program of Innovation and Entrepreneurship for Undergraduates, in 2020, and the Fundamental Research Funds for the Central Universities CHD, in 2021. His research interests include graph theory and image processing and artificial intelligence.



YUE ZHU was born in Jiangsu, China, in 2000. She received the bachelor's degree in mathematics and applied from Chang'an University. She participated in the National Training Program of Innovation and Entrepreneurship for Undergraduates, in 2020. After graduation, she will go to the Tongji University to pursue the master's degree in solving the independence test problem. Also, she won the National Second Prize in Chinese Mathematical Contest in Modeling and the Meritorious prize in American Mathematical Contest in Modeling, in 2022.

HONGQIANG WANG was born in Hebei, China, in 2000. He entered Chang'an University, in 2019, and majors in mathematics and applied mathematics. He participated in the National Training Program of Innovation and Entrepreneurship for Undergraduates, in 2020. After graduation, he will go to the Northwestern Polytechnical University to pursue the master's degree in harmonic analysis. He won the National Second Prize in Chinese Mathematical Contest in Modeling, in 2020.