A new hybrid approach based on genetic algorithm for minimum vertex cover

Sinem Çınaroğlu
Computer Engineering Department
Necmettin Erbakan University
Konya, Turkey
soren@konya.edu.tr

Sema Bodur
Computer Engineering Department
Ege University
İzmir, Turkey
sema.bodur@ege.edu.tr

Abstract—Minimum vertex cover (MVC) problem is a NP-Hard optimization problem which we often encounter in real life applications like wireless sensor networks, graph theory, bioinformatics, social network analysis etc. To deal with this optimization problem in an efficient way, we introduce a new Hybrid Genetic algorithm (NHGA) to solve MVC problem. In this study, the proposed algorithm has also been tested on DIMACS benchmarks, BHOSLIB benchmarks and random graphs. Performance of NHGA is then compared with the simple Genetic algorithm (GA) and Clever Greedy algorithm. Consequently, experimental results show that NHGA finds better solutions than other algorithms for MVC, since it offers near to optimal solutions.

Keywords—minimum vertex cover; genetic algorithm; clever greedy algorithm; hybrid genetic algorithm

I. INTRODUCTION

A vertex cover (VC) for a graph is a set of vertices, such that every edge in the graph is incident with at least one vertex in that set. The minimum vertex cover (MVC) is defined as the vertex cover with the smallest size. A graph can have multiple VC but the value of MVC is unique. The problem of finding minimum vertex cover plays a key role in real life applications such as bioinformatics, finance, social networks, communications, analysis of data, network security, scheduling etc… Some problems that use ideas of vertex cover have additional or modified constraints compared to vertex cover. For example; a shop owner wants to minimize the cost of installing a camera and wants to buy as few as possible cameras, while still covering all of the areas. In this case, a minimum vertex cover would be needed. The problem of finding the minimum vertex cover for a graph is a NP-Hard problem and cannot be solved optimally in polynomial time. Therefore, various alternative approaches have been considered by researchers to solve these problems. These approaches are heuristics algorithms, approximation techniques, meta-heuristics algorithms, parameterized algorithms as well as direct algorithms. For direct algorithms; there are the Maximum Degree Greedy algorithm, Depth First Search (DFS), the Edge Deletion (ED) algorithm, the ListLeft (LL) algorithm, the ListRight (LR) algorithm and etc… For the intelligent algorithms; Genetic algorithm (GA), Greedy Heuristic, Simulated Annealing algorithm, Kernelization, Tabu Search, and the Ant Colony algorithm [1].

In order to solve MVC problem efficiently, a hybrid genetic algorithm called NHGA is proposed in this paper. The NHGA combines new repair technique with a classical GA. We have developed new heuristic vertex offspring from the given graph. The performance of all algorithms is compared with the output solution of the Clever Greedy algorithm and simple GA. The rest of the paper has been organized as follows. Section II describes MVC problem definition and presents the literature review. Section III explains the GA and Clever Greedy. Section IV presents our proposed technique. In Section V, we analyze the performance of our technique and also compared to GA and Clever Greedy on some instances of the DIMACS, BHOSLIB and random graphs. Numerical results and approximation rate are also included. Section VI contains the conclusions.

II. RELATED WORK

Problem definition [2]: $G=(V,E)$ is an undirected graph, where $V = \{1, 2, 3, ..., n\}$ is the set of vertices and $E \subseteq V \times V$ the set of edges. An edge between vertices $u, v$ is denoted by the pair $(u, v) \in E$, and we define the adjacency matrix $(e_{u,v})$ according to

\[
e_{u,v} = \begin{cases} 
1, & \text{if } (u, v) \in E \\
0, & \text{otherwise}
\end{cases}
\]

Feasible solution: A set $V'$ of nodes such that $\forall (u, v) \in E : u \in V' \lor v \in V'$

Objective function: The size $|V'|$ of vertex cover set $V'$

Optimal solution: A vertex cover set $V'$ that minimizes $|V'|$

Karp [3] showed that minimum vertex cover is NP-Hard. The vertex cover problem can be proved to be NP-complete by a reduction from 3-SAT or, as Karp did, by a reduction from the Clique problem. In fact, even approximating optimal solutions within a factor of $1.3606$ is NP-hard [4].

Various heuristic techniques such as GAs, simulated annealing, ant colony optimization and kernelization, have been applied to solve the MVC problem in the recently. In this
section, the GA based studies for the solution of the vertex cover problem are as follows.

Bäck and Khuri [2] show experimentally that a GA performs very well on instances of sizes \( n=100 \) and \( n=202 \) of the PapadimitriouSteiglitz graph. The results found by the GA are better than those obtained from the best traditional heuristic, the vercov algorithm.

Kalapala, Pelikan and Hartmann [5] analyzed performance of the branch-and-bound (BB) algorithm, hierarchical Bayesian optimization algorithm (hBOA), the simple GA and the parallel simulated annealing (PSA). In their work they show performance for standard classes of random graphs and transformed SAT instances.

Milanovic [6] has presented his solving way the generalized vertex cover problem (GVCP) by GA. In this paper the experiments were carried out on randomly generated instances with up to 500 vertices and 10000 edges. The performance of the GA was compared with the CPLEX solver and 2-approximation algorithm based on LP relaxation.

Chandu [7] has proposed Map-Reduce implementation that helps to run the GA for generalized vertex cover problem. In these implementation fitness computation operations, crossover operations and mutation operations are distributed among all the machines in Hadoop cluster running reduce phase.

Recently, a hybrid heuristic for the MWVC (Minimum Weight Vertex Cover) problem was developed by Singh and Gupta [8]. Initially, a steady-state GA is used to generate valid vertex cover solutions. Later, a heuristic is applied to these solutions that tries to eliminate the redundant vertices from them. They have compared their algorithm against the ACO technique of Shyu, Lin and Yin and have produced better results.

Kotecha and Gambhava [9] have presented a hybrid GA for solving the vertex cover problem. They have added local optimization technique to the GA, and also developed a new heuristic vertex crossover operator especially for the vertex cover problem.

Hongwei, Xuezhou, and Zheng [10] have developed a hybrid GA for the vertex cover problems in which scan-repair and local improvement techniques are used for local optimization. The experimental results indicate that SLGA can obtain excellent and quality solutions of problem instances with different size and the experimental design methods which can effectively improve the GA.

III. COMPARED ALGORITHMS

A. Genetic Algorithm (GA)

GA is family of meta-heuristics that are natural optimization mechanisms based on the genetic processes of biological organisms. It has been developed by John Holland in the University of Michigan, USA. These algorithms use techniques inspired by the Darwinian theory of evolution such as crossover, mutation and natural selection. The evolution usually starts from a population randomly or heuristically initialized. In each generation, the quality of the solutions is evaluated by a fitness function; individuals are stochastically selected and are modified by crossover and mutation operators to form a new population. This process continues until the optimal solution is found or a termination criteria set by the user is met [11]. The optimization mechanism of the GA system is based on four important features. The first one is the fitness function that is applied when children are choosing the next population. The second one is the encoding scheme of the problem that is an appropriate representation of candidate solutions. The other two; crossover operator and mutation operator are the most important operator in a GA. Crossover is a recombination process in which exchange of segments of between two pairs of chromosomes called parents. Mutation maintains the genetic diversity between successive generations of population. A local optimum is avoided by using a mutation operator which prevents the over-similarity of the chromosomes in the population, thus should be selected carefully. After mutation and crossover, the next step is choosing the elite individual. If the best solution of current generation is better than the elite individual of all previous populations, then it is saved as elite individual [12]. The general framework of the GA systems is presented in Algorithm 1.

Algorithm 1: GA

```
Initialization of chromosomes
while not Termination_Criteria() do
    Fitness_Function()
    Selection()
    Crossover()
    Mutation()
    Population_update()
    Elitism()
end while
Output_data()
```

B. Clever Greedy Algorithm

Clever Greedy algorithm is a technique used to solve optimization problems. The greedy method is quite powerful and works well for a wide range of problems. This algorithm works on each step by making the decision that seems most promising at any moment; it never reconsiders this decision; whatever situation may arise later. That is, it makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution [13]. However, generally greedy algorithm does not provide globally optimized solutions. The algorithm pseudo-code is presented in Algorithm 2.

Algorithm 2: Clever Greedy

```
Input: G(V,E)
Output: Set \( V_c \) subset of \( V \), vertex cover
\( V_c \leftarrow \varnothing \)
while \( E \neq \varnothing \)
    \( c \leftarrow \max \text{DegreID} \)
    \( V_c = V_c \cup \{c\} \)
    \( E = E - \{e \in E | c = (c,u), u \in V\} \)
end while
```
This algorithm always takes the vertex highest degree, adds it to the cover set, removes it from the graph, and repeats.

IV. PROPOSED METHOD

In this section we present the proposed algorithm, the pseudo-code of the algorithm.

A. Terminology

Let $G = (V, E)$, where $G$ is undirected graph, $V$ denotes a vertex set and $E$ denotes an edge set. $N(v)$ refer to neighbor of $v$, for each $v \in V$. Degree is defined as the number of neighbors for vertex $v \in V$. $d(v)$ refer to degree of vertex $v \in V$. Adjacency matrix of given graph $G$ is defined by $A[i, j]$.

B. The Proposed Algorithm

The generation of the initial population plays an important role in the performance of the GA. In the classical GA the initial population consists of randomly generated individuals (chromosomes). Random generation of individuals for MVC problem may result in individuals whose entire initial population is not vertex-covered. This negatively affects the performance of the GA for MVC. In such a case, the crossover and mutation procedures to achieve diversity are also insufficient. A novelty has been introduced to generate the initial population of the classical GA that finds the optimal or near optimal solution of the MVC problem with the proposed algorithm and the performance of the simple GA which has been tried to improve through the novelty. The architecture of the proposed method is given in Fig. 1.

The NHGA algorithm starts with finding the degree of each vertex by means of the adjacency matrix $A[i, j]$. Then, a vertex $v$ is randomly selected from set $V$. After adding the vertex to vertex cover list $V_c$, all adjacent edges to this vertex are deleted. If there is an unvisited edge and the degree of all adjacent vertices to the selected vertex is 0, then a vertex having a maximum degree is finding in graph $G$. If there is an unvisited edge and the degree of all adjacent vertices to the selected vertex is not 0, a vertex having a maximum degree is selected among the degree of all adjacent vertices to the selected vertex. The latest selected vertex is added to the list $V_c$ and is proceeded by the above process until the edge set $E$ has no edges i.e. up to $A[i, j] \neq 0 \forall i, j \in V$. The pseudo-code of the proposed algorithm NHGA is given in Algorithm 3.

The NHGA is employed for different startup vertices. Candidate solutions which are the vertex cover are obtained from the employment of NHGA. Another candidate solution is obtained from the employment of Clever Greedy algorithm. An initial population is an integration of these direct candidate solutions and randomly generated candidate solutions. GA is conducted with this initial population. The used GA is the same as the classical GA except that the initial population is created.

Fig. 1 The architecture of the NHGA
Algorithm 3: NHGA

\textbf{Input}: \( G(V, E) \)
\textbf{Output}: Set \( V' \) subset of \( V \), vertex cover

\( V' \leftarrow \emptyset \)
//Calculate the degree for each vertex

Pick any vertex \( c \in V \), \( c \neq c \)
\[ E = E - \{ e \in E | e = (c, u), u \in V \} \]
\( d(c) \leftarrow 0 \)

\textbf{foreach} \( v_j \in N(c) \)
\[ d(v_j) \leftarrow d(v_j) - 1 \]
\textbf{end foreach}

\textbf{while} \( (E \neq \emptyset) \)
\[ \max \text{Degree} \leftarrow -\infty \]
\[ \max \text{DegreeID} \leftarrow 0 \]
\textbf{if} \( \forall v_i \in N(c) \) and \( d(v_i) = 0 \) \textbf{then}
\[ \text{foreach} \ v_j \in V \]
\textbf{if} \( d(v_j) > \text{max Degree} \) \textbf{then}
\[ \max \text{Degree} \leftarrow d(v_j) \]
\[ \max \text{DegreeID} \leftarrow v_j \]
\textbf{end if}
\textbf{end foreach}
\textbf{else}
\[ \text{foreach} \ v_j \in N(c) \]
\textbf{if} \( d(v_j) > \text{max Degree} \) \textbf{then}
\[ \max \text{Degree} \leftarrow d(v_j) \]
\[ \max \text{DegreeID} \leftarrow v_j \]
\textbf{end if}
\textbf{end foreach}
\textbf{end if}
\[ c \leftarrow \text{max DegreeID} \]
\[ V' = V' \cup \{ c \} \]
\[ E = E - \{ e \in E | e = (c, u), u \in V \} \]
\[ d(c) \leftarrow 0 \]
\textbf{foreach} \( v_j \in N(c) \)
\[ d(v_j) \leftarrow d(v_j) - 1 \]
\textbf{end foreach}
\textbf{end while}

The parameters used in the GA are as follows:

- \textit{Solution representation}: For \( N \)-vertices graph, the chromosome is an array of length \( N \). Each gene in this array corresponds to a vertex. Chromosomes are represented in binary as strings of 0s and 1s, where 1s denote the inclusion and 0s the exclusion in the vertex cover \( V_c \).
- \textit{Generation size}: 1000
- \textit{Population size}: 200
- \textit{Crossover}: One-point crossover with a rate of 0.8
- \textit{Mutation}: Bit string mutation with a rate of 0.07

V. EXPERIMENTAL RESULTS

In this section, we refer to the DIMACS\(^1\), BHOSLIB\(^2\) and random graphs\(^3\) data sets used to evaluate the effectiveness of NHGA mentioned in the previous section, as well as the comparison results of the proposed NHGA with GA and Clever Greedy algorithm.

A. Benchmark Datasets Used

The data used for our experiments are 15 instances of DIMACS benchmark, 2 instances of BHOSLIB benchmark and 3 instances of random graphs.

B. Results for Datasets

All algorithms were implemented in C# language and the experimental results were obtained on a machine Intel Core i7 2.8 GHz CPU with 16GB of RAM. To be more reliable results NHGA was run 25 times. To measure the solution quality, we record the approximation ratio which is the ratio of the obtained solution to the optimal solution. Hence the smaller the approximation ratio, the better the quality.

Obtained experimental results are summarized in Table 1. The table has the following components: the instances, number of vertices, number of edges, optimal solutions, the size of vertex cover solution found by each algorithm and approximation ratio of each algorithm. Table 1 also shows that proposed NHGA yields better than Clever Greedy algorithm and GA on all instances of DIMACS benchmark, BHOSLIB benchmark and random graphs or equal results to the optimal.

VI. CONCLUSIONS

Minimum vertex cover is a classic graph optimization problem that is NP-Hard even to approximate well. In this paper, we developed a new hybrid genetic algorithm called NHGA. We have used NHGA which creates different children and designed especially for VCP. NHGA gives better solutions when compared to GA and Clever Greedy algorithm. Our approach gives best solutions for DIMACS, BHOSLIB benchmark and random graphs instances. At the same time our experiments were conducted to evaluate the performance of the proposed table.

In the future we will extend dimensions of our work to weighted graphs.

---

1. https://turing.cs.hbg.psu.edu/txn131/vertex_cover.html
3. https://turing.cs.hbg.psu.edu/txn131/vertex_cover.html
### Table I. Experimental Results

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Vertices</th>
<th>Edges</th>
<th>Optimal MVC</th>
<th>Greedy</th>
<th>GA</th>
<th>NHGA</th>
<th>Approximation Ratios</th>
</tr>
</thead>
<tbody>
<tr>
<td>brock200_2.clq.compliment</td>
<td>200</td>
<td>10024</td>
<td>188</td>
<td>192</td>
<td>192.4</td>
<td>190.04</td>
<td>1.021277, 1.023404, 1.010851</td>
</tr>
<tr>
<td>brock200_4.clq.compliment</td>
<td>200</td>
<td>6811</td>
<td>183</td>
<td>187</td>
<td>188.8</td>
<td>186.72</td>
<td>1.021858, 1.031694, 1.020328</td>
</tr>
<tr>
<td>C125.9.clq-compliment</td>
<td>125</td>
<td>787</td>
<td>91</td>
<td>93</td>
<td>98.16</td>
<td>92</td>
<td>1.021978, 1.078681, 1.010989</td>
</tr>
<tr>
<td>C250.9.clq-compliment</td>
<td>250</td>
<td>3141</td>
<td>206</td>
<td>214</td>
<td>219.12</td>
<td>212.12</td>
<td>1.038835, 1.063689, 1.029709</td>
</tr>
<tr>
<td>c-fat200-1.clq-compliment</td>
<td>200</td>
<td>18366</td>
<td>188</td>
<td>188</td>
<td>188.64</td>
<td>188</td>
<td>1, 1.003404, 1</td>
</tr>
<tr>
<td>c-fat200-2.clq-compliment</td>
<td>200</td>
<td>16665</td>
<td>176</td>
<td>176</td>
<td>177.04</td>
<td>176</td>
<td>1, 1.005909, 1</td>
</tr>
<tr>
<td>frb30_15_5</td>
<td>450</td>
<td>17794</td>
<td>183</td>
<td>187</td>
<td>186.72</td>
<td>186.72</td>
<td>1.021858, 1.031694, 1.020328</td>
</tr>
<tr>
<td>frb35_17_2</td>
<td>595</td>
<td>27847</td>
<td>183</td>
<td>188</td>
<td>188.64</td>
<td>188</td>
<td>1, 1.003404, 1</td>
</tr>
<tr>
<td>graph50_06</td>
<td>50</td>
<td>857</td>
<td>38</td>
<td>38</td>
<td>38</td>
<td>38</td>
<td>1, 1, 1</td>
</tr>
<tr>
<td>graph100_01</td>
<td>100</td>
<td>3465</td>
<td>60</td>
<td>60</td>
<td>64.6</td>
<td>60</td>
<td>1, 1.076667, 1</td>
</tr>
<tr>
<td>graph500_05</td>
<td>500</td>
<td>9356</td>
<td>290</td>
<td>290</td>
<td>439.8</td>
<td>290</td>
<td>1, 1.516552, 1</td>
</tr>
<tr>
<td>hamming6_2</td>
<td>64</td>
<td>192</td>
<td>32</td>
<td>32</td>
<td>39.5</td>
<td>32</td>
<td>1, 1.234375, 1</td>
</tr>
<tr>
<td>hamming8_2</td>
<td>256</td>
<td>1024</td>
<td>128</td>
<td>128</td>
<td>209.9</td>
<td>209.9</td>
<td>1.01875, 1.0175, 1.0125</td>
</tr>
<tr>
<td>jhonson8_2_4</td>
<td>28</td>
<td>168</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>1, 1, 1</td>
</tr>
<tr>
<td>jhonson16_2_4</td>
<td>120</td>
<td>1680</td>
<td>112</td>
<td>113</td>
<td>112.5</td>
<td>112</td>
<td>1, 1.008929, 1.004464</td>
</tr>
<tr>
<td>keller4</td>
<td>171</td>
<td>5100</td>
<td>160</td>
<td>163</td>
<td>162.8</td>
<td>162</td>
<td>1, 1.01875, 1.0175, 1.0125</td>
</tr>
<tr>
<td>keller5</td>
<td>776</td>
<td>7471</td>
<td>749</td>
<td>760</td>
<td>762.2</td>
<td>757.56</td>
<td>1.014686, 1.017623, 1.011429</td>
</tr>
<tr>
<td>phat300_3</td>
<td>300</td>
<td>11460</td>
<td>264</td>
<td>268</td>
<td>283.1</td>
<td>267.96</td>
<td>1.015152, 1.072348, 1.015</td>
</tr>
<tr>
<td>phat700_2</td>
<td>700</td>
<td>122922</td>
<td>656</td>
<td>659</td>
<td>684.2</td>
<td>658.96</td>
<td>1.004573, 1.042988, 1.004512</td>
</tr>
</tbody>
</table>

### Acknowledgment
The authors would like to thank to Assoc. Prof. Dr. Orhan Dağdeviren for many helpful discussions.

### References


