Solving 2-domination problem in graphs using genetic algorithm

Alfred Raju M* and P. Venkata Subba Reddy

Department of Computer Science and Engineering, National Institute of Technology Warangal, Warangal, 506004, Telangana, India.

*Corresponding author(s). E-mail(s): ar22csr2r02@student.nitw.ac.in; Contributing authors: pvsr@nitw.ac.in;

Abstract

In a simple, undirected graph $\boldsymbol{G(V,E)}$, a *2-dominating set* is a subset $\boldsymbol{R}$ of vertices such that every vertex $\boldsymbol{u}$ in $\boldsymbol{V-R}$ has at least two neighbours in $\boldsymbol{R}$. The *2-domination number* denoted by $\boldsymbol{\gamma_2(G)}$ is the minimum cardinality of a 2-dominating set of $\boldsymbol{G}$. Given a graph $\boldsymbol{G}$, determining $\boldsymbol{\gamma_2(G)}$ is termed as *2-domination problem* (*2-DOM*). 2-DOM is known to be NP-hard but there are no known metaheuristic algorithms for solving the problem. In order to counter this, we propose a genetic algorithm-based solution which gives a near-optimal solution for 2-DOM problem. The proposed algorithm uses a heuristic to generate a population of feasible solutions and generate a better feasible solution by passing it through various steps of genetic algorithm. Initially, experiments were carried out on graphs for which optimal values are known to verify the algorithm's effectiveness. The experiments were also carried out on random graphs generated using *Erdős-Rényi* graph model, a prominent graph model for graph generation and *Harwell-Boeing* (HB) dataset, a well-known dataset for graph problems. Since there exists no metaheuristic algorithms for solving the 2-DOM problem, our results set a benchmark for future research on the problem.

Keywords: NP-hard, dominating set, 2-domination number, genetic algorithm.

1 Introduction

The concept of domination was first introduced by Claude Berge, in 1957 [1] paving a way for further exploration and research in the field of domination theory. Let $G(V,E)$ be a simple, undirected graph where $V(G), E(G)$ (or simply V, E) represents set of vertices and edges respectively. For a vertex $u \in V(G)$, the *open neighborhood* denoted by $N(u)$ is the set $\{v : (v, u) \in E(G)\}$ and its *degree* denoted $deg(u)$ is $|N(u)|$. The *closed neighborhood* of u is $N[u] = \{u\} \cup N(u)$. $\Delta(G), \delta(G)$ represent the highest and smallest degrees of graph G respectively. A subset R of V is called a *dominating set* of graph G if every vertex in $V - R$ is adjacent to a vertex in R and the minimum cardinality of R is terned as *domination number* of the graph G denoted by $\gamma(G)$. Finding domination number of a graph is proven NP-hard and there has been vigorous research for proposing algorithms for solving the same. Therefore, many metaheuristic algorithms exists in the literature for solving the problem [2][3]. Due to its applications in various fields, numerous other variants have been proposed which can be adapted according to the requirements [4][5]. One of these variants is the *2-domination.* A subset R of V is called a *2-dominating set* if every vertex in $V - R$ has at least two vertices from R adjacent to it. *2-domination number* denoted by $\gamma_2(G)$ is the minimum cardinality of a 2-dominating set of graph G. Given a graph G, determining $\gamma_2(G)$ is termed as *2-domination problem* (*2-DOM*).

The first article on 2-domination (also called *double domination*) was published in 1996 [6]. This laid a foundation for the research on multi-domination concept and eventually k-domination was also proposed, where k is an integer. 2-DOM is proven NP-hard [7] which implies polynomial time algorithm does not exist for general graphs. 2-DOM is NP-hard for split graphs, chordal bipartite graphs and planar graphs [7]. However exact values for $\gamma_2(G)$ when G is a cycle is obtained in [8]. An upper bound on 2-domination number interms of Δ and δ is obtained in [9]. Bounds on $\gamma_2(G)$ when G is a cactus or toroidal graph has been well-studied in [8][10]. Special graph classes for which $\gamma = \gamma_2$ is investigated in [11]. General bounds for k-domination in graphs were specified in [12] but were improved in [9].

Since the 2-DOM problem has applications in communication networks, network design, sensor placement and other social networks, fault tolerant systems or backup servers where a network can be maintained with a backup in case of server failure [5]. Proposing an efficient polynomial algorithm can help tackle the limitations faced in applications. To the best of our knowledge, there are no metaheuristic algorithms to solve 2-DOM problem. So, a genetic algorithm based approach has been proposed to solve 2-DOM problem. The proposed algorithm can produce a near optimal solution in polynomial time. Rest of the paper is organised as follows: First a brief introduction on genetic algorithm is given. Next, the proposed approach is discussed followed by the experimental results and conclusion.

2 Genetic Algorithm

A *genetic algorithm* is a metaheuristic that borrows ideas from the evolution and natural selection processes. By emulating the concepts of natural selection and genetics, it is frequently employed to solve challenging optimization problems. Each individual entity in the population is a *chromosome* which is a collection of *genes.* In the work proposed, the *label* assigned to each vertex in the graph is the value of each gene. Though the algorithm doesn't guarantee an optimal solution to a problem, it is capable of generating a near optimal solution. The series of genetic operations are : *initial*

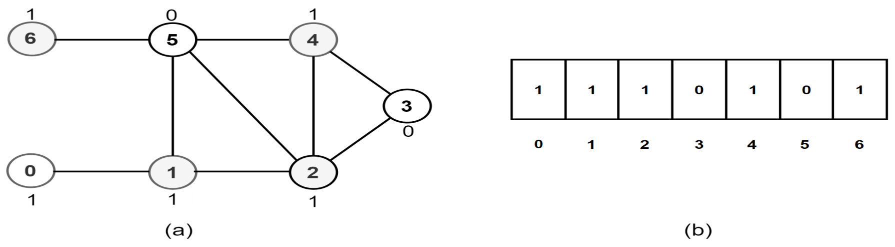

Fig. 1 (a) Graph with 2-dominating set {0,1,2,4,6} (b) Solution representation

Algorithm 1 Construct Solution

Input: A simple, undirected graph G

Output: A 2-dominating set of G

```
1:  Declare $label[0 \ldots |V(G)| - 1]$
2:  $V' = V$
3:  while $V' \neq \phi$ do
4:      Select a vertex $u$ randomly from $V'$
5:      if $deg(u) \geq 2$
6:          $label[u] = 0$
7:          Select two unlabelled neighbors $v, w$ of $u$ and set $label[v]$=1, $label[w]$=1
8:          $V' = V' \setminus \{u, v, w\}$ and $E' = E' \setminus \{$edges with $u, v$ or $w$ as an end point$\}$
9:      else
10:         $label[u] = 1$
11:         $V' = V' \setminus \{u\}$ and $E' = E' \setminus \{$edges with $u$ as an end point$\}$
12:     update degrees of vertices in $V'$
13: end while
14: return $label$
```

population generation, selection, crossover and *mutation*. Each operation may be carried out using different ways specific to the problem. In the proposed work random selection, a two-point crossover and a single-bit mutation are used.

2.1 Parameters

- **Initial Population** (*inpop*)**:** It is the initial population generated using the construct solution which contains a group of feasible solutions.
- **Fitness Score** (*fscore*)**:** It is the summation of all the labels attributed to the vertices.
- **Iteration Best** (*itr_best*)**:** It is the minimum *fscore* among the solutions in particular generation.
- **Termination Condition:** Termination condition is the number of iterations and its value is considered as 100000.
- **Global Best or Overall Best** (*gbest*)**:** It is the global best which is the minimum *fscore* obtained from all the generations after the termination condition.

3 Construct Solution

We propose a binary heuristic for finding a 2-dominating set of a graph G, where $0, 1$ are used to label the vertices. In the proposed method, the vertices in a 2-dominating set are assigned label '1' and label '0' otherwise. According to the definition of 2-DOM, every '0' labelled vertex must be adjacent to at least 2 vertices labelled '1'.

In the proposed heuristic, a vertex u is randomly chosen and is labelled '0' if its degree is at least 2 i.e $deg(u) \geq 2$ and two of the neighbors of u are labelled with '1'. If the degree of selected vertex u is less than 2 then vertex u is labelled '1'. Next, the labelled vertices are removed from the graph and the set of unlabelled vertices is updated. Degree of all vertices in the updated graph is calculated. The procedure is repeated until all the vertices are labelled. Clearly, the set generated by the heuristic is a 2-dominating set of graph G. Pseudocode for construct solution is given in Algorithm 1 and its working is illustrated with an example below.

A graph G with vertex set $\{0, 1, ..., n-1\}$, where $n = |V(G)| - 1$, is provided as input to Algorithm 1 and $label[0, ..., n-1]$ is declared as a solution array such that $label[u]$ is the label assigned to vertex u. For the graph shown in Figure 1(a), initially $V' = \{0, 1, 2, 3, 4, 5, 6\}$. If vertex 3 is randomly chosen then $label[3] = 0$ as $deg(u) = 2$. Next, two neighbors 2 and 4 of vertex 3 are selected and labelled as '1' i.e $label[2] = 1$ and $label[4] = 1$. Now, we have $V' = \{0, 1, 5, 6\}$. In the next iteration, suppose that vertex 5 is chosen randomly and since $deg(5) = 2$ in the updated graph, $label[5] = 0$ and the neighboring vertices 1 and 6 are labelled '1' i.e $label[1] = 1$ and $label[6] = 1$. The labelled vertices are removed from V' and the updated vertex set $V' = \{0\}$. Since $deg(0) < 2$ in the updated graph, vertex 0 gets label '1' i.e $label[0] = 1$. Next $V' = \phi$ and the set of vertices with label '1' is $\{0, 1, 2, 4, 6\}$. Each of the remaining vertices 3 and 5 are adjacent to at least two vertices with label '1' in G. Hence, $\{0, 1, 2, 4, 6\}$ is a 2-dominating set of with $fscore = \sum_{i=0}^{6} label[i] = 1+1+1+0+1+0+1 = 5$. Figure 1(b) shows the obtained feasible solution representation.

4 Proposed Solution

The psuedocode for the proposed solution using genetic algorithm for solving 2-DOM problem is given in Algorithm 2 and is explained in detail in this section.

4.1 Initial Population

In the proposed solution, an initial population (*inpop*) of 1000 feasible solutions is generated by iterating the construct solution function 1000 times. Next, genetic operations are performed on the generated population to refine them and obtain a feasible solution with better $fscore$ value since 2-DOM is a minimization problem.

4.2 Selection Operation

This operation selects two items from *inpop* which are referred as the parents $P1$ and $P2$ throughout the paper. The selection can be done using different methods such as tournament selection, roulette wheel selection, random selection and so on. In this

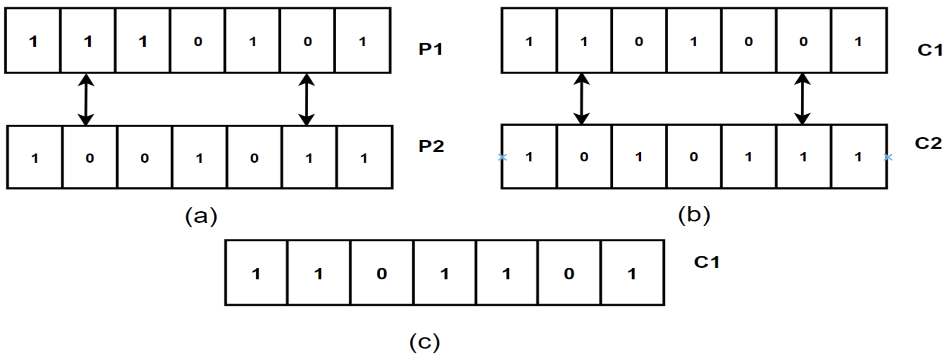

Fig. 2 (a) Selected Parents for Crossover (b) Obtained Children after Crossover (c) C1 after Mutation

work, *random selection* is considered after observation as it was capable of generating better solution than the other methods mentioned. The $fscore$ of selected parents $P1$ and $P2$ is calculated by adding the labels that were previously assigned to the vertices of graph.

4.3 Crossover Operation

The two randomly selected parents from the selection operation serves as the input for the current phase. The crossover operation is performed on the selected parents $P1$ and $P2$ where a pointer is placed by selecting the genes randomly. Based on the placement of the pointer, the crossover operation is performed. A *two-point crossover* is used in this work where two points x_1, x_2 are chosen randomly and the genes present in between the chosen pointers are swapped thus altering chosen chromosomes. The corresponding parents are modified after the crossover and the outputs are referred as the children i.e $P1 \longrightarrow C1$ and $P2 \longrightarrow C2$. It is to note that solution represented by the obtained children chromosomes need not be feasible i.e., need not be 2-dominating sets of G and requires a thorough checking which is done in the next phase. Figure 2(b) shows the chromosomes after crossover operation is performed.

4.4 Feasibility and Mutation

The outputs from the crossover operation are referred as children $C1$ and $C2$ and since the obtained outputs from crossover operation may contain infeasible solutions, it is imperative to perform feasibility check on them. For example, the solution represented by $C1$ shown in Figure 2(b) gives a set $\{0, 1, 3, 6\}$ of vertices with label 1. The vertex 4 with label '0' is dominated by only one vertex with label '1'. Hence $\{0, 1, 3, 6\}$ does not give a 2-dominating set and the solution represented by $C1$ is not feasible. Therefore the solution represented by $C1$ needs to be modified and is done using the Feasibility check procedure given in Algorithm 3. The solution represented by $C2$ shown in Figure 2(b) gives a set $\{0, 2, 4, 5, 6\}$ of vertices with label 1. Clearly $\{0, 2, 4, 5, 6\}$ is a 2-dominating set and hence $C2$ is a feasible solution.

Algorithm 2 Proposed Algorithm

```
Input: Simple, undirected graph $G$
Output: 2-domination number $\gamma_2(G)$
 1: Declare $itr\_best, gbest$
 2: Generate initial population of size 1000 using Algorithm 1
 3: $itr = 1$
 4: while $itr \leq 100000$ do
 5:     Select two parents $P1, P2$ randomly from $inpop$
 6:     Perform two-point crossover on $P1, P2$
 7:     Feasibility check on $C1, C2$ using Algorithm 3
 8:     Update $itr\_best$
 9:     if $itr\_best < gbest$
10:         $gbest = itr\_best$
11:     $itr = itr + 1$
12: end while
13: return $gbest$
```

The feasibility function searches for the '0' labelled vertices and checks whether it is dominated by at least two vertices with label '1'. If the obtained children are feasible, $fscore$ is calculated and compared with that of the corresponding parent i.e. $fscore$ of $C1$ is compared with $P1$ and $fscore$ of $C2$ with $P2$. If $C_i < P_i$ where $i = 1, 2$, the child chromosome is replaced with that of parent in the generated population. If the child is infeasible, then it undergoes mutation where minimum number of genes are altered to obtain a feasible solution. If any '0' labelled vertex is not adjacent to two vertices labelled '1', then the '0' labelled vertex is altered to '1' and the example is illustrated in Figure 2(c). The process is repeated until all the vertices adhere to the $2 - DOM$ rules. The fitness score of obtained child after mutation is calculated and compared with the respective parent as previously specified. The genetic operations are iterated till the termination condition is met which is 100000 iterations in this work. At the end of each iteration, the minimum fitness score among $P1, P2, C1, C2$ is noted which is the iteration best $fscore$ (itr_best). The iteration best is compared with the minimum $fscore$ obtained so far i.e. global best($gbest$) and updated if $itr\_best < gbest$. So at the end of algorithm, $gbest$ contains the best $fscore$ i.e. minimum $fscore$ value obtained in all the generations.

5 Experimental Results

Tests have been conducted on *Intel i*5 12th generation machine and executed in C++ language. Initially, the experiments were carried out on cycle graphs for which the optimal values are known to prove the algorithm's correctness and were extended to *Erdős-Rényi* graph models and *HB* graphs. For cycle graphs C_n on n vertices, the optimal solution $\gamma_2(C_n) = \frac{n}{2}$ [8] and for random graphs, the upper bound is $\frac{2ln(\delta+1)+1}{(\delta+1)} * n$ where n is the number of vertices and δ is the minimum degree of graph

Algorithm 3 Feasibility Check and Mutation

Input: Simple, undirected graph G and a child chromosome $C[0, ..., |V(G)| - 1]$
Output: A feasible child chromosome $C[0, ..., |V(G)| - 1]$

```
1: Declare $sum = 0$
2: for $v \in V(G)$ do
3:     if $C[v] = 0$
4:         for $u \in N(v)$
5:             $sum = sum + C[u]$
6:         if $sum < 2$
7:             $C[v] = 1$
8: end for
9: return $C$
```

[9] [?]. Table 1 & Table 2 shows the results obtained for cycle graphs and *Erdős-Rényi* graph models respectively. The algorithm produces optimal solutions for the cycle graphs which proves its correctness and gives solutions which are far less than the upper bounds specified in the literature for *Erdős-Rényi* graph models and more than 100 *HB* graphs with vertices upto 2300 whose resulted are not mentioned in the paper. This shows the efficiency of proposed genetic algorithm for solving 2-DOM problem. Since there are no metaheuristic algorithms proposed in the literature for solving the 2-domination problem, this work sets a benchmark for further research on it.

Table 1 Results for known optimal values of some graphs

Graphs	$n = \|V\|$	Optimal Value	Obtained Results
Cycle Graphs	8	4	4
	11	6	6
	16	8	8
	20	10	10
	25	13	13
	50	25	25
	95	48	48

Table 2 Results Obtained for *Erdős-Rényi* graphs

$n = \|V\|$	p	Upper Bound	Obtained Results
25	0.2	24	11
	0.3	24	10
100	0.2	50	23
	0.3	35	18
300	0.2	62	32
	0.3	41	20
500	0.2	67	33
	0.3	51	21

6 Conclusion

In this paper, a genetic algorithm based solution has been proposed for solving 2-DOM problem which is NP-hard. Since there are no metaheuristic algorithms for 2-DOM, the algorithm's efficiency has been evaluated on cycle graphs, where it produced accurate values. Experiments were also performed on *Erdős-Rényi* graph models and *Harwell Boeing* graphs and found that the obtained results are within the bounds. These set a benchmark for further research on metaheuristics for solving the 2-DOM problem. Also the proposed genetic algorithm based solution can be used with necessary modifications for solving other similar domination problems.

References

[1] Berge., C.: Two theorems in graph theory and some applications. Proceedings of the National Academy of Sciences of the United States of America. **44**(3), 247–251 (1957)

[2] Hedar, A.-R., Ismail., R.: Hybrid genetic algorithm for minimum dominating set problem. In: Zaimis, E. (ed.) Computational Science and Its Applications–ICCSA 2010:, vol. 10, pp. 457–467. Springer, Heidelberg (2010)

[3] Nitash, C.G., Singh., A.: An artificial bee colony algorithm for minimum weight dominating set. IEEE Symposium on Swarm Intelligence., 1–7 (2014)

[4] Chellali, M., Rad, N.J., Sheikholeslami, S., Volkmann, L.: Varieties of roman domination. Structures of domination in graphs. **66**, 273–307 (2021)

[5] T.W.Haynes, S.H., P.Slater: Fundamentals of domination in graphs. CRC press. (1998)

[6] Harary, F., Haynes, T.W.: Nordhaus-gaddum inequalities for domination in graphs. Discrete Mathematics. **155**(1-3), 99–105 (1996)

[7] Argiroffo, G., Leoni, V., Torres, P.: On the complexity of {k}-domination and k-tuple domination in graphs. Information Processing Letters. **115**(6-8), 556–561 (2015)

[8] Chellali, M.: Bounds on the 2-domination number in cactus graphs. Opuscula Mathematica. **26**(1), 5–12 (2016)

[9] Bujtás, C., Jaskó, S.: Bounds on the 2-domination number. Discrete Applied Mathematics. **242**, 4–15 (2018)

[10] Shaheen, S. Ramy: Bounds for the 2-domination number of toroidal grid graphs. International Journal of Computer Mathematics. **86**(4), 584–588 (2009)

[11] Hansberg, A., Volkmann., L.: On graphs with equal domination and 2-domination numbers. Discrete Mathematics. **308**(11), 2277–2281 (2008)

[12] Caro, Y., Roditty., Y.: A note on the k-domination number of a graph. International Journal of Mathematics and Mathematical Sciences. **13**, 205–206 (1990)

Table 3 Results obtained for Harwell Boeing graphs

Graph	Number of Vertices	Number of Edges	Upper Bound	Obtained Results
can24	24	92	23	11
jagmesh_2	1009	3937	952	230
ash85	85	304	91	37
ash219	219	438	234	62
ash292	292	1250	276	109
ash331	331	662	353	82
ash608	608	1216	648	146
ash958	958	1916	1021	228
bcspwr04	274	943	327	139
bcspwr05	443	1033	529	263
bp_0	822	3276	877	360
can73	73	225	78	32
can144	144	720	111	47
can161	161	769	123	58
can187	187	839	158	65
can1054	1054	6625	806	208
can1072	1072	6758	819	215
curtis54	54	291	58	23
dwt_59	59	163	71	29
dwt_162	162	672	194	64
dwt_193	193	1843	125	68
dwt_198	198	795	237	77
dwt_209	209	976	198	89
dwt_221	221	925	209	89
dwt_310	310	1379	293	92
dwt_346	346	1786	413	137
dwt_361	361	1657	341	110
dwt_503	503	3265	475	179
dwt_918	918	4151	866	304
dwt_992	992	8868	640	480
eris1176	1176	9864	1404	514
ibm32	32	126	31	19
jagmesh_1	936	3600	883	381
jgl009	9	50	7	3
jgl011	11	76	7	3
lshp_265	265	1009	250	131
lshp_406	406	1561	383	179
lshp_1009	1009	3937	952	428
pores_1	30	180	23	11
rgg010	10	76	6	2
will57	57	281	69	21
will199	199	701	213	75
bcsstk05	153	1288	145	50
bfw62a	62	450	67	25
bfw62b	62	342	74	29
bfw398a	398	3678	425	128
bfw398b	398	2910	475	152
ck400	400	2860	478	182
dwb512	512	2500	546	173
dwg961b	961	5776	735	479
fidapm05	42	520	33	9
fs_680_1	680	2646	812	402

Graph	Number of Vertices	Number of Edges	Upper Bound	Obtained Results
gre_216a	216	876	231	75
impcol_e	225	1308	240	76
lshp_577	577	2233	545	287
lshp_778	778	3025	734	339
lshp_1270	1270	4969	1198	541
nnc261	261	1500	312	87
nnc666	666	4044	795	222
nnc1374	1374	8606	1640	458
qh882	882	3354	1053	386
rdb200	200	1120	189	51
rdb200l	200	1120	189	50
rdb450	450	2580	425	126
rdb450l	450	2580	425	126
rdb800l	800	4640	755	279
rw496	496	1859	529	164
str_0	363	2454	387	152
tols90	90	1746	33	18
tubs100	100	396	107	28
tubs1000	1000	3996	1066	385
utm300	300	3155	358	117
west0156	156	371	167	70
west0167	167	507	178	62
west0381	381	2157	322	179
west0479	479	1910	511	197
bus662	662	1568	790	364
add20	2395	17319	2858	778
bcsstk02	66	2211	10	2
bcsstk03	112	376	106	40
bcsstk04	132	1890	57	36
can61	61	309	52	15
cdde2	961	4681	1025	386
fs_183_1	183	1069	219	59
fs_183_3	183	1069	219	59
fs_183_4	183	1069	219	59
fs_183_6	183	1069	219	59
fs_541_1	541	4285	414	283
fs_541_2	541	4285	414	277
fs_680_1	680	2646	812	402
fs_760_1	760	5976	717	238
gr_30_30	900	4322	849	292
gre_1107	1107	5664	1045	449
hor_131	434	4710	410	206
illc1033	1033	4732	975	198
impcol_b	59	312	63	24
impcol_c	137	411	147	58
impcol_d	425	1339	453	163
lop163	163	935	138	79
lund_a	147	1298	125	82
lund_b	147	1294	125	82
nos4	100	347	120	54
saylr1	238	1128	254	72
shl_0	663	1687	707	257
shl_200	663	1726	707	263
shl_400	663	1712	707	264
young1c	841	4089	897	327