Genetic and Particle Swarm Optimisation Algorithms for Secure Domination Problem

Sista Gopala Krishna(421258)
Shaik Anas Hameed(421255)
Mayank Kumar Cam(421218)
Under the guidance: Mr. P. Sunil

Department of Computer Science and Engineering
NIT Andhra Pradesh

May 5, 2024

List of Contents

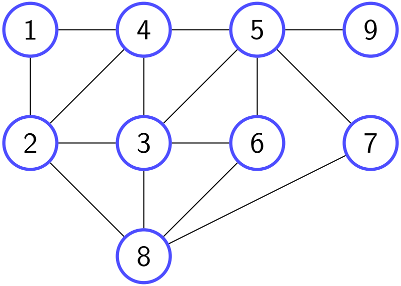
Abstract

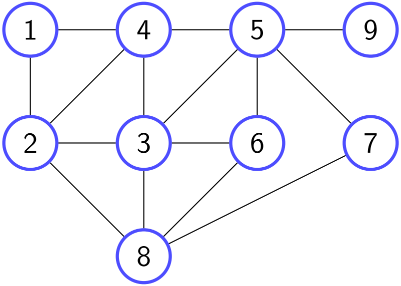
In a simple, undirected graph $G(V, E)$, a subset of vertices D is called a dominating set of G if every vertex in $V - D$ has a neighbor in D. The minimum cardinality of a dominating set of G is called domination number denoted by $\gamma(G)$. The problem of determining $\gamma(G)$ of a graph G, known as the domination problem is NP-hard but different metaheuristic algorithms have been proposed to solve it. A set $S \subseteq V(G)$ is a secure dominating set (SDS) of G if for each vertex $v \in V(G)S$ there exists a vertex $u \in S$ such that v is adjacent to u and the set $(S - \{u\}) \cup \{v\}$ is a dominating set of G. The minimum cardinality of a SDS of G is called secure domination number and is denoted by $\gamma_s(G)$. Given a graph G, determining $\gamma_s(G)$ is termed as secure domination problem (SDOM). SDOM is known to be NP-hard which implies no polynomial time algorithm for the same.

Abstract (Contd.)

To the best of our knowledge, unlike domination problem, there are no known metaheuristic algorithms for SDOM problem. In order to counter this, we propose a genetic algorithm and particle swarm optimisation algorithm-based solutions to solve SDOM problem. The proposed algorithms uses a heuristic to generate a population of feasible solutions and generate a better feasible solution by passing it through various steps of genetic and partical swarm optimisation algorithms. Performance of the proposed genetic and particle swarm optimisation algorithms for SDOM problem has been tested 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 model, a prominent model for graph generation and Harwell-Boeing (HB) dataset, a well-known dataset for graph problems.

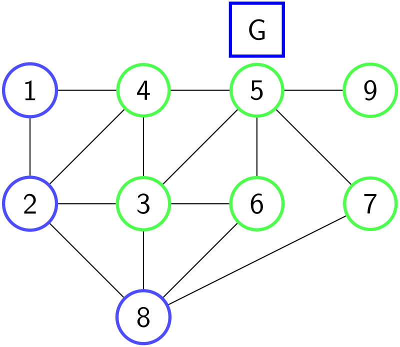
Motivation and Introduction



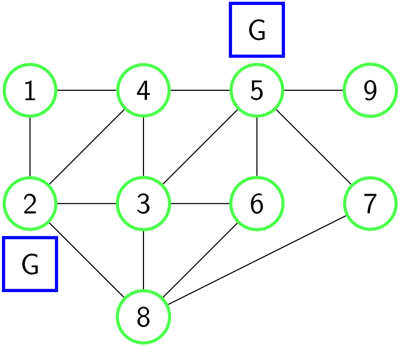


Deployment of Security Guards

Deployment of Security Guards



Deployment of Security Guards



Motivation

Deployment of security or military personnels

Example:

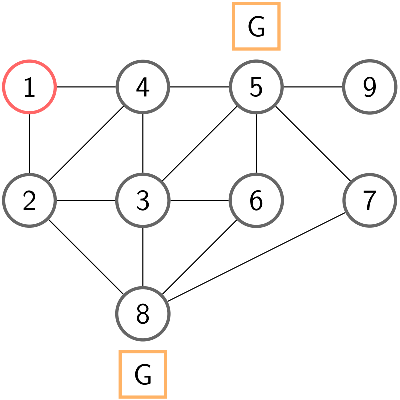


Figure 1: Graph G_1

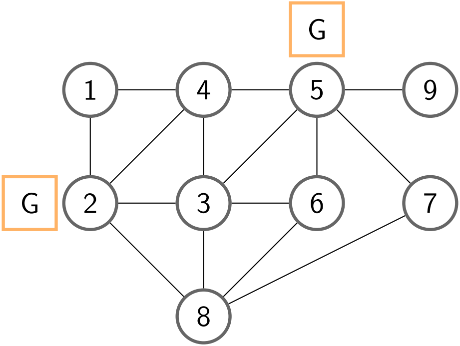
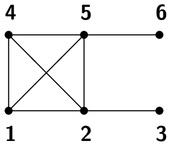


Figure 2: Graph G_2

Graph: A *graph* is an ordered pair $G(V, E)$ comprising of a set V of vertices, and a set E of edges. We consider simple, connected and undirected graphs.



Graph G

Open neighbourhood: $N(v) = \{u \in V(G) : (u, v) \in E(G)\}$.
Example: $N(1) = \{2, 4, 5\}$

Closed neighbourhood: $N[v] = N(v) \cup \{v\}$.

Example: $N[1] = \{1, 2, 4, 5\}$

Degree of a vertex: $\text{Degree}(v) = |N(v)|$.
Example: $\text{Degree}(1) = |N(1)| = |\{2, 4, 5\}| = 3$

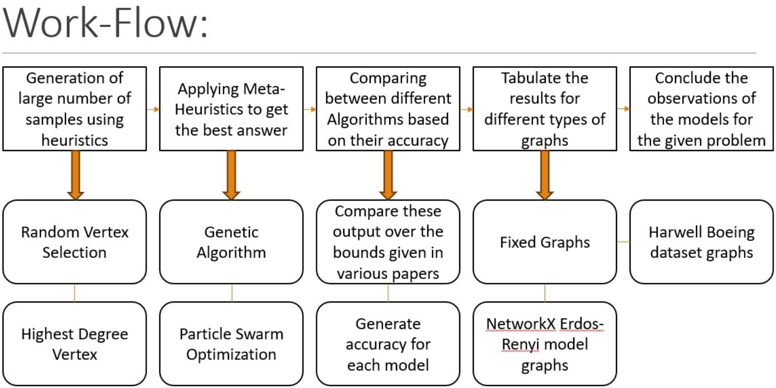


Figure 3: Workflow

- The study of domination in graphs began in 1958 by Claude Berge. But officially **dominating set** term was introduced by Oysten Ore in 1962 [7].

Definition

A subset S of V is called a **dominating set** of graph G if every vertex in $V \setminus S$ is adjacent to a vertex in S and the minimum cardinality of S is the **domination number** of the graph G denoted by $\gamma(G)$.

- The concept is being used extensively in wireless sensor networks, facility placement, server placement etc [7].

Dominating Set : An Example

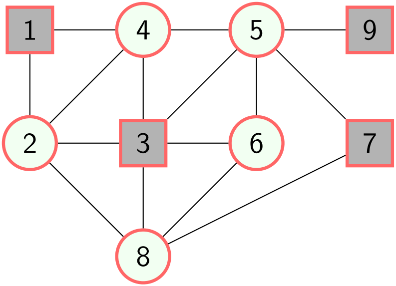


Figure 4: Graph G

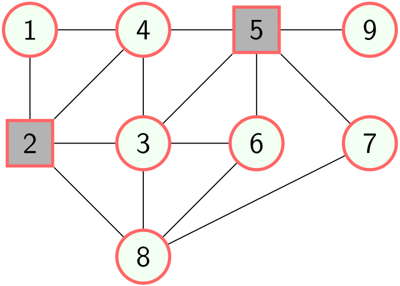


Figure 5: Graph G with γ-set

$\{1,3,7,9\}$, $\{2,7,9\}$, $\{3,7,9\}$, $\{1,5,8\}$, $\{2,5\}$, $\{2,8,9\}$,etc are dominating sets of G.

$\{2,5\}$ is a minimum dominating set of G. Hence $\gamma(G) = 2$.

The **optimization version** of domination problem is given below.

Minimum Domination Problem (MIN-DOM)
Instance : A graph $G = (V, E)$.
Solution : Domination number of G i.e., $\gamma(G)$.

The decision version of domination problem is defined as follows.

Domination Decision Problem (DOM)
Instance : A graph $G = (V, E)$ and a positive integer $k \leq |V|$.
Question : Does G has a dominating set of size at most k?

MIN-DOM has been proven **NP-hard** whereas DOM is proven to be **NP-complete** [7].

Secure domination was introduced by E.J. Cockayne et al in 2005 [4].

Definition

A dominating set $S \subseteq V(G)$ is said to be a *Secure Dominating Set (SDS)* in G if for every $u \in V(G) \setminus S$ there exists $v \in S$ such that $(u, v) \in E(G)$ and $(S \setminus \{v\}) \cup \{u\}$ is a dominating set in G. The *secure domination number* of graph G is the minimum cardinality of a *SDS*, and is denoted by $\gamma_s(G)$. [4]

Example

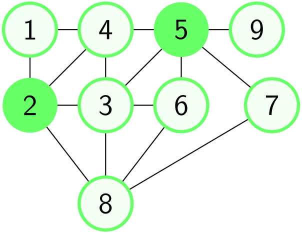


Figure 6: Graph G_1

Example

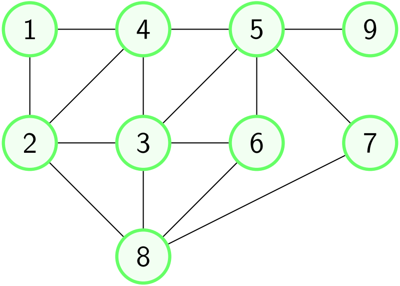


Figure 7: Graph G

$\{1,3,7,9\}, \{2,5,7,9\}$ and $\{1,5,8\}$ are secure dominating sets of graph G.

Goal: Find a secure dominating set of G with minimum size.

Example

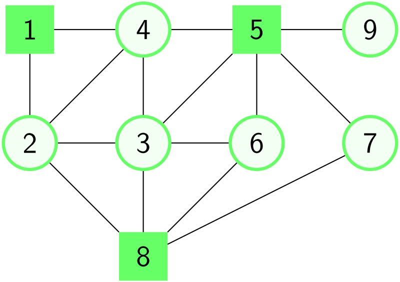


Figure 8: Graph G with γ_s-set

$\{1, 5, 8\}$ is a minimum secure dominating set of G.
Hence $\gamma_s(G) = 3$.

Optimization and Decision Versions: Secure Domination Problem

The **optimization version** of secure domination problem is given below.

Minimum Secure Domination Problem (MIN-SDOM)
Instance : A graph $G = (V, E)$.
Solution : Secure domination number of G i.e., $\gamma_s(G)$.

The decision version of secure domination problem is defined as follows.

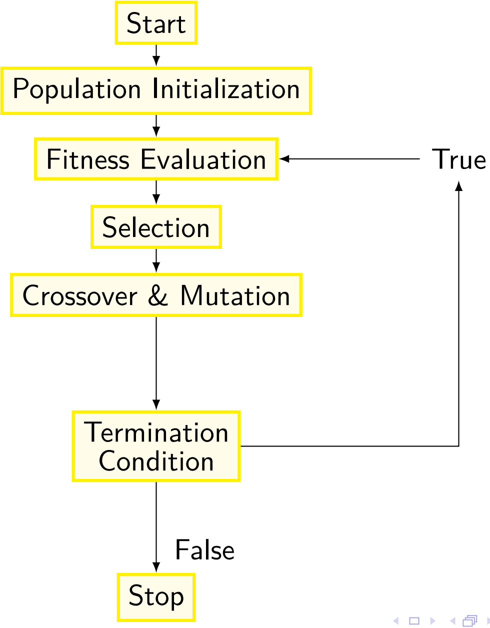
Secure Domination Decision Problem (SDOM)
Instance : A graph $G = (V, E)$ and a positive integer $k \leq |V|$.
Question : Does G has a secure dominating set of size at most k?

MIN-SDOM has been proven as **NP-hard** whereas SDOM is proven to be **NP-complete** [7].

1. For any graph G, $\gamma_s(G) \leq \gamma(G) + \beta_0(G) - 1$. [5]
2. For two path graphs, P_m and P_k, $\gamma_s(P_m \Box P_k) \leq \lceil \frac{mk}{3} \rceil + 2$, where $P_m \Box P_k$ is the Cartesian product of graphs P_m and P_k. [4]

Literature Review (Contd...)

- Secure domination problem (MIN-SDOM) is NP-hard for bipartite graphs and split graphs [1].
- MIN-SDOM is NP-hard for subclasses of bipartite graphs [6].
- MIN-SDOM is NP-hard for doubly chordal graphs [6].
- MIN-SDOM is linear time solvable for block graphs [2].
- MIN-SDOM is APX-hard for graphs with maximum degree 4, which shows the difficulty of obtaining an apprximation algorithm to solve MIN-SDOM. [6]
- No metaheuristic algorithm has been given to solve MIN-SDOM.
- Hence, we propose a genetic algorithm and partical swarm optimisation algorithm to solve MIN-SDOM.


Start
Population Initialization
Fitness Evaluation
True
Selection
Crossover & Mutation
Termination Condition
False
Stop

Genetic Algorithm for Secure Domination Problem

Genetic algorithms simulate the process of natural selection which means those species who can adapt to changes in their environment are able to survive and reproduce and go to next generation. It uses artificial construction of search algorithms.

Generating Parameters for Genetic Algorithm

- Initial Population (*initial_pop*)
- Number of Solutions (*num_sol*)
- Cost obtained in current population (*present_min_val*)
- Intermediate population (*intermediate_pop*)
- Best Solution (best_sol)
- Optimal Cost (*optimal_val*)

Pseudocode for Genetic Algorithm

Algorithm 1 Pseudocode for Genetic Algorithm

Input: A simple, undirected graph G
Output: Best Approximate Solution

```
Initialize num_sol
Create initial_pop
present_min_val ← minimum cost on initial_pop
optimal_val ← present_min_val
while termination condition do
    intermediate_pop ← Crossover & Mutation on initial_pop
    initial_pop ← intermediate_pop
    present_min_val ← minimum cost on initial_pop
    if optimal_val > present_min_val then
        optimal_val ← present_min_val
        best_sol ← present_optimal_sol
return best_sol
```

MIN-SDOM : Heuristic for Initial Population Generation

In this heuristic, we first find a dominating set of the given graph G by repeating the following step until the graph is empty.

Step 1. Select a vertex v of the graph G arbitrarily and remove vertex v and its neighbors $N(v)$ from G.

- Let D be the set of vertices selected.
- Every vertex of G not in D is adjacent to a vertex of D.
- Therefore D is a dominating set of G.

Step 2. If D is a secure dominating set of G, we are done. Otherwise a subset of vertices from $V \setminus D$ are added to D to make it a secure dominating set of G.

Algorithm 2 Pseudocode for Heuristic

Input: A simple, undirected graph G
Output: A secure dominating set of G

$R \leftarrow$ `FindDominatingSet`(G)

Determine whether R is a secure dominating set of G using `IsSecureDomSet`(G,R) algorithm

If R is a secure dominating set of G then return R

Else

  Add sufficient number of vertices of $V(G) \setminus R$ to R to make it a secure dominating set S of G using `Set-SecureDomSet`(G,R) algorithm.

  Return S

Pseudo Code for `FindDominatingSet()` Procedure

Algorithm 3 Pseudocode for `FindDominatingSet`(G) Algorithm

Input: A simple, undirected graph $G(V, E)$
Output: A dominating set of G

$D \leftarrow \emptyset$
while $V(G) \neq \emptyset$ **do**
  select a vertex $v \in V(G)$
  $D \leftarrow D \cup \{v\}$
  $V(G) \leftarrow V(G) \setminus N[v]$
  $E(G) \leftarrow E(G) \setminus \{(u, v) : u \in V(G) \text{ or } v \in V(G)\}$
return D

Pseudo Code for `IsSecureDomSet()` Procedure

Algorithm 4 Pseudocode for `IsSecureDomSet`(G,D) Algorithm

Input: A simple, undirected graph G and a set $D \subseteq V(G)$
Output: If it is Secure Domination set return *True* else *False*

```
D' ← V(G) − D
S, S' ← D, D'
Initialize count3 and count4 to 0
for j in 1 to lengthofD' - 1 do
    for i in 1 to lengthofD - 1 do
        if D'[j] in adjacencyList[D[i]] then
            swap D[i] and D'[j]
            X, Y = lengthofD - 1, lengthofD' - 1
            Initialize count1 and count2 to 0
            while X and Y greater than or equal to 0 do
                if D'[Y] in adjacencyList[D[X]] then
                    Decrement Y
                    Increment count1
                    X = lengthofD - 1
                else
                    Decrement X
                    Increment count2
            if count1 is equal to lengthofD' then
                Increment count3
    if count3 greater than or equal to 1 then
        Increment count4
    count3 = 0
if count4 is equal to lengthofS' then
    return True
else
    return False
```

Algorithm 5 Pseudocode for `Set-SecureDomSet`(G,D) Algorithm

```
Input: A simple, undirected graph G and a set D ⊆ V(G)
Output: Secure Domination set S
  D' ← V(G) − D
  S, S' ← D, D'
  Initialize count3 and count4 to 0
  R and R' are empty lists
  for j in 1 to lengthofD' - 1 do
      for i in 1 to lengthofD - 1 do
          if D'[j] in adjacencyList[D[i]] then
              swap D[i] and D'[j]
              X, Y = lengthofD - 1, lengthofD' - 1
              Initialize count1 and count2 to 0
              while X and Y greater than or equal to 0 do
                  if D'[Y] in adjacencyList[D[X]] then
                      Decrement Y
                      Increment count1
                      X = lengthofD - 1
                  else
                      Decrement X
                      Increment count2
              if count1 is equal to lengthofD' then
                  Increment count3
      if count3 greater than or equal to 1 then
          Increment count4
      else
          R.append(S'[j])
      count3 = 0
  if count4 is equal to lengthofS' then
      return S
  else
      R' ← union of non-duplicates(R) and S
      return R'
```

Selection Operator

- Selection operator is applied to select best possible candidates for crossover and mutation.
- Initial population of 1000 feasible solutions is generated using the proposed Heuristic.
- Next, we select two solutions among the 1000 solutions using tournament selection operator.

Crossover and Mutation Operator

- Crossover and Mutation are the two important operations in Genetic Algorithm. Here, we use the uniform crossover operation.
- Uniform crossover operation is performed on the best two solutions S_1 and S_2 selected from 1000 solutions after performing the tournament selection operation.
- Result of the crossover operation is a set of two child solutions which may or may not be feasible.
- Next, we perform the feasibility test to check if the two child solutions are feasible.

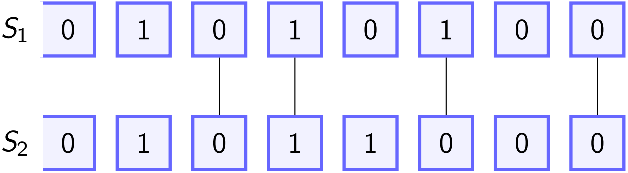


Figure 9: Selected Solutions $S_1 \& S_2$

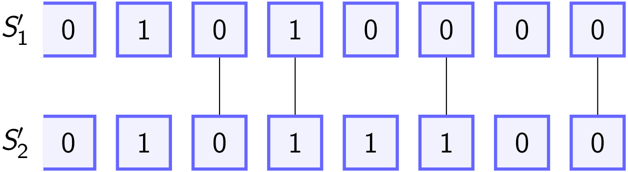


Figure 10: Solutions $S_1' \& S_2'$ After Crossover

Is S_1' a Valid Secure Dominating Set of graph G?

0	1	0	1	0	0	0	0

Figure 11: Solution S_1' after Crossover

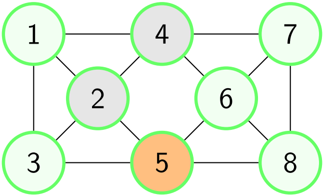


Figure 12: Graph G labelled as per S_1'

Is S_2' a Valid Secure Dominating Set of graph G?

0	1	0	1	1	1	0	0

Figure 13: Solution S_2' after Crossover

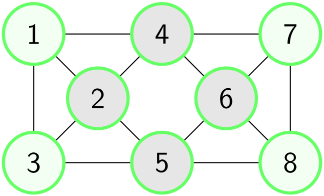


Figure 14: Graph G labelled as per S_2'

Pseudocode for Feasibility Check Procedure

Algorithm 6 Feasibility check

Input: A simple, undirected graph G and a set $C \subseteq V(G)$
Output: Modified feasible child solution

```
bool = IsSecureDomSet(G, C)
if bool then
    return C
else
    Set-SecureDomSet(G, C)
```

Solutions S_1' and S_2' after Feasibility check

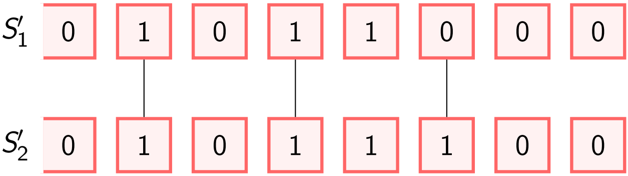


Figure 15: Solutions S_1' and S_2' after Feasibility check

Pseudocode for Partical Swarm Optimization Algorithm

Initialize population
for $t = 1$: maximum generation
    for $i = 1$: population size
        if $f\left(x_{i,d}(t)\right) < f\left(p_i(t)\right)$ **then** $p_i(t) = x_{i,d}(t)$
            $f\left(p_g(t)\right) = \min_i \left(f\left(p_i(t)\right)\right)$
        end
        for $d = 1$: dimension
            $v_{i,d}(t+1) = wv_{i,d}(t) + c_1 r_1 \left(p_i - x_{i,d}(t)\right) + c_2 r_2 \left(p_g - x_{i,d}(t)\right)$
            $x_{i,d}(t+1) = x_{i,d}(t) + v_{i,d}(t+1)$
            if $v_{i,d}(t+1) > v_{\max}$ **then** $v_{i,d}(t+1) = v_{\max}$
            else if $v_{i,d}(t+1) < v_{\min}$ **then** $v_{i,d}(t+1) = v_{\min}$
            end
            if $x_{i,d}(t+1) > x_{\max}$ **then** $x_{i,d}(t+1) = x_{\max}$
            else if $x_{i,d}(t+1) < x_{\min}$ **then** $x_{i,d}(t+1) = x_{\min}$
            end
        end
    end
end

Process of PSO

- Initialize Parameters inertia weight (w), two positive constants (c_1, c_2) and two random parameters within [0-1] (r_1, r_2)[10].
- Intitialize Position (x_i and Velocity (V_i) randomly for each particle.
- Evaualte Fitness $f(x_i^t)$. If fitness Value is better than global best then update global best.
- Calculate Particle Position and velocity using formulas show in above algorithm.
- Evaualte Fitness $f(x_i^t)$ and find the current best and update to next iterations. Finally output is *gBest* (gobal best) and x_i^t position.
- After doing pso we get particles of the swarm in float values. we have to set the Thresholds for the particles (0,1) and need to perform the feasibility check.

Pseudocode for Fitness function

Algorithm 7 Fitness check

Input: a set $C \subseteq V(G)$ and initial population P of simple graph G
Output: Return new population

```
weight = sum(feasible child solution C values)
if weight < min(initial population weights) then
    return P union C
else
    return P
```

- Initially, the experiments were carried out on paths, cycle graphs and ladder graphs for which the optimal values are known to prove the algorithm's correctness.

Graph	**Optimal Results**
Path	$\lceil \frac{3n}{7} \rceil$ [3]
Cycle	$\lceil \frac{3n}{7} \rceil$ [1]
Ladder ($P_2 \square P_n$)	$\lceil \frac{3n+1}{4} \rceil$ [9]

Table 1: Standard graphs with known $\gamma_s(G)$

Experimental results for special graphs (GA)

Graph	n	$\gamma_s(G)$	Obtained Results		
			# of iterations		
			1000	10000	100000
Path	13	6	6	6	6
	22	10	11	10	10
	64	28	36	35	32
	100	43	57	54	52
Cycle	13	6	6	6	6
	22	10	11	10	10
	64	28	34	32	31
	100	43	56	53	51
Ladder Graph	14	6	7	6	6
	22	9	10	9	9
	64	25	32	28	25
	100	38	50	46	43

Table 2: Results obtained for special graphs

- Performance of the proposed genetic and Partical Swarm Optimisation algorithms will be tested on random graphs generated using Erdös-Rényi model and Harwell Boeing data sets as it is used by most of the researchers working in this domain.

Theorem

For every simple, undirected graph G, $\gamma_s(G) \leq \gamma_2(G)$.

Proof.

Let S be a 2-dominating set of G. For every vertex u not in S there exists a neighbor v in S such that $(S - \{v\}) \cup \{u\}$ is a dominating set of G. Hence S is also a secure dominating set of G. Therefore, $\gamma_s(G) \leq \gamma_2(G)$. □

Experimental results for NetworkX Erdös-Rényi model graphs (GA) (contd..)

n	p	**UB**	**Obtained Result**
17	0.1	17	12
	0.2	21	7
20	0.1	20	10
	0.2	24	8
24	0.1	24	12
	0.2	26	8
25	0.1	25	12
	0.2	27	10
62	0.1	74	21
	0.2	53	12
78	0.1	84	26
	0.2	47	14
94	0.1	89	29
	0.2	47	18

Table 3: Results obtained for NetworkX Erdös-Rényi model graphs.

Experimental results for Harwell Boeing data sets graphs (GA)

Graph	$V(G)$	$E(G)$	**UB**	**Obtained Result**
*rgg*010	10	76	6	1
*jgl*011	11	76	7	3
*jgl*009	9	50	7	2
impcol_b	59	312	63	22
*ibm*32	32	126	31	11
*fidap*005	27	279	13	4
*dwt*_59	59	163	71	25
*curtis*54	54	291	58	19
*can*_73	73	225	78	35
*bcspwr*02	49	108	47	26
*bcspwr*01	39	85	37	19

Table 4: Results obtained for Harwell Boeing data sets graphs.

Experimental results for Harwell Boeing data sets graphs (GA)

Graph	$V(G)$	$E(G)$	**UB**	**Obtained Result**
*ash*85	85	304	91	30
*dwt*_162	162	672	194	52
*west*0156	156	371	167	66
*bfw*62*b*	62	342	74	23
*dwt*_72	72	147	68	41
*fidapm*05	42	520	20	17
*can*_24	24	92	23	6
*dwt*_66	66	193	63	22
*bfw*62*a*	62	450	67	18
*can*_61	61	309	43	12
*bcsstk*03	112	376	86	42

Table 5: Results obtained for Harwell Boeing data sets graphs.

Experimental results for special graphs (PSO)

Graph	n	$\gamma_s(G)$	Obtained Results		
			# of iterations		
			1000	10000	100000
Path	13	6	6	6	6
	22	10	10	10	10
	64	28	35	33	30
	100	43	55	51	47
Cycle	13	6	6	6	6
	22	10	10	10	10
	64	28	33	31	29
	100	43	55	52	46
Ladder Graph	14	6	6	6	6
	22	9	10	9	9
	64	25	31	25	25
	100	38	58	45	41

Table 6: Results obtained for special graphs

Experimental results for NetworkX Erdös-Rényi model graphs (PSO) (contd..)

n	p	UB	Obtained Result
17	0.1	17	12
	0.2	21	7
20	0.1	20	10
	0.2	24	9
24	0.1	24	12
	0.2	26	9
25	0.1	25	11
	0.2	27	11
62	0.1	74	21
	0.2	53	15
78	0.1	84	24
	0.2	47	17
94	0.1	89	27
	0.2	47	22

Table 7: Results obtained for NetworkX Erdös-Rényi model graphs.

Experimental results for Harwell Boeing data sets graphs (GA)

Graph	$V(G)$	$E(G)$	**UB**	**Obtained Result**
*rgg*010	10	76	6	1
*jgl*011	11	76	7	5
*jgl*009	9	50	7	6
impcol_b	59	312	63	14
*ibm*32	32	126	31	11
*fidap*005	27	279	13	4
*dwt_*59	59	163	71	25
*curtis*54	54	291	58	19
*can_*73	73	225	78	35
*bcspwr*02	49	108	47	26
*bcspwr*01	39	85	37	19

Table 8: Results obtained for Harwell Boeing data sets graphs.

Experimental results for Harwell Boeing data sets graphs (GA)

Graph	$V(G)$	$E(G)$	**UB**	**Obtained Result**
*ash*85	85	304	91	28
*dwt*_162	162	672	194	51
*west*0156	156	371	167	69
*bfw*62*b*	62	342	74	24
*dwt*_72	72	147	68	39
*fidapm*05	42	520	20	14
*can*_24	24	92	23	8
*dwt*_66	66	193	63	23
*bfw*62*a*	62	450	67	19
*can*_61	61	309	43	14
*bcsstk*03	112	376	86	35

Table 9: Results obtained for Harwell Boeing data sets graphs.

Experimental Results

- Since there are no known non-trivial bounds on the secure domination number of general graphs, based on the fact $\gamma_s(G) \leq \gamma_2(G)$ we are using the results obtained for determining 2-domination number of graph in [8] for the comparison purpose and consider it as upper bound (UB).
- The upper bound is $\frac{2*ln(\delta+1)+1}{(\delta+1)} * n$ where n is the number of vertices and δ is the minimum degree of graph
- Experiments show that the result obtained by the proposed algorithm for solving SDOM problem is very much less than the upper bound, emphasizing that the algorithm is effective.

Experimental Results

- PSO algorithm purpose is to find the global optimum of the fitness function defined in a given area. GA algorithm modifies a population of individual solutions repeatedly.
- a Genetic Algorithms complexity is $O(g(nm + nm + n))$ with g the number of generations, n the population size and m the size of the individuals. Therefore the complexity is on the order of $O(gnm))$.
- PSO Algorithm complexity depends on swarm size S and no of dimensions D. Complexity is $O(S * D)$. However, this can vary depending on the specific problem and algorithm variations.

Conclusion

- We have proposed a genetic algorithm based solution for solving secure domination problem which is NP-hard.
- Effectiveness of the proposed meta-heuristic algorithm is tested on the random graphs generated using NetworkX Erdős-Rényi model, a popular model for graph generation and Harwell Boeing (HB) graph dataset.
- The obtained results are much lower than the upper bound. This fact emphasizes that the algorithm is efficient and designing better metaheuristic algorithms for the problem remains open.

[1] A.P. Burger, A.P. Devilliers and J.H. Van Vuuren, *"on Minimum Secure Dominating Sets of Graphs"*, Quaestiones Mathematicae, 39(2) (2016) 189-202.

[2] D. Pradhan and A. Jha, *"on Computing a Minimum Secure Dominating Set in Block Graphs"*, Journal of Combinatorial optimization, 35 (2018) 613-631.

[3] E.J. Cockayne, P.J.P. Grobler, W.R Grundlingh, J. Munganga and J.H. Van vuuren, *"Protection of a Graph"*, Utilitas Mathematica, 67 (2005).

[4] E.J.Cockayne, P.J.P.Grobler, W.R.Grundlingh, J.Munganga, and J.H.van Vuuren, *"Protection of a graph."* Utilitas Mathematica **67** (2005): 19-32.

[5] H.B.Merouane, and M. Chellali, "On secure domination in graphs." Information Processing Letters **1150** (2015): 786-790.

[6] H. Wang, Y. Zhao, and Y. Deng. *"The complexity of secure domination problem in graphs"*, Discussiones Mathematicae Graph Theory, 38(2) (2018) 385-396.

[7] K.S.Booth, and J.H. Johnson, *"Dominating sets in chordal graphs."* SIAM Journal on Computing **11.1** (1982): 191-199.

[8] M. Alfred Raju and P. V. Subba Reddy, *Solving 2-domination problem in graphs using genetic algorithm*. Accepted for presentation in the 4$^{\mathrm{th}}$ International Conference on Advanced Engineering Optimization Through Intelligent Techniques to be held at SVNIT Surat, India from 28-30 September, 2023.

[9] M. Haythorpe and A. Newcombe, *"The Secure Domination Number of Cartesian Products of Small Graphs with Paths and Cycles"*, Discrete Applied Mathematics, 309 (2022) 32-45.

[10] Dai, H.P., Chen, D.D. and Zheng, Z.S., 2018. Effects of random values for particle swarm optimization algorithm. Algorithms, 11(2), p.23.