Greedy Algorithm for Finding Minimum Dominating Set

Email: kuangwj@sjtu.edu.cn,
School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University.

***Abstract*—Minimum dominating set (MDS) can be applied to solve many optimization problems in practice, such as wireless networking. In this report, we solve the problem of finding a MDS for a graph by greedy strategy.**

***Index Terms*—minimum dominating set (MDS), greedy algorithms (GA).**

I. Introduction

Dominating sets are of practical interest in several areas. In wireless networking, dominating sets are used to find efficient routes within ad-hoc mobile networks. They have also been used in document summarization, and in designing secure systems for electrical grids. Enumerating all minimal dominating sets allows immediate solution of corresponding NP-hard optimisation and counting problems. Greedy strategies are common in solving the problem of finding a minimum dominating set (MDS) for a graph

A greedy algorithm (GA), as the name suggests, always makes the choice that seems to be the best at that moment. This means that it makes a locally-optimal choice in the hope that this choice will lead to a globally-optimal solution. GA have a long history of study in combinatorial optimization and theoretical computer science. Greedy heuristics are known to produce suboptimal results on many problems.

The goal of this course project is to develop and test practical algorithms for finding a MDS for a graph by greedy strategy.

II. Preliminaries of MDS

Before we describe our first algorithm, we need to formalize some well-known graph theoretic concepts [1][2].

Definition 1 (Dominating Sets). A dominating set of an undirected graph $G = (V, E)$ is a set $U \subseteq V$ such that every vertex $v \in V - U$ has at least one neighbor in U.

A minimum dominating set is a dominating set of smallest size in a given graph. The size of a minimum dominating set is known as the domination number of the graph.

A minimum dominating set is always a minimal dominating set, but the converse does not necessarily hold.

Theorem 1 （Upper bounds). Let $G = (V, E)$ be a graph on n vertices, with minimum degree $\delta > 1$. Then G has a dominating set of at most $n[1 + \ln(\delta + 1)]/(\delta + 1)$ vertices.

Proof. Let $p \in [0, 1]$ be, for the moment, arbitrary. Let us pick, randomly and independently, each vertex of V with probability p. Let X be the (random) set of all vertices picked and let $Y = Y_X$ be the random set of all vertices in $V - X$ that do not have any neighbor in X. The expected value of $|X|$ is clearly np. For each fixed vertex $v \in V$, $\Pr[v \in Y] = \Pr[v\text{and its neighbours are not in}X] \leq (1 - p)^{\delta+1}$. Since the expected valuéof a sum of random variables is the sum of their expectations (even if they are not independent) and since the random variable $|Y|$ can be written as a sum of n indicator random variables $\chi_v(v \in V)$, where $\chi_v = 0$ if $v \in Y$ and $\chi_v = 0$ otherwise, we conclude that the expected value of $|X| + |Y|$ is at most $np + n(1 - p)^{\delta+1}$. Consequently, there is at least one choice of $X \subseteq V$ such that $|X| + |Y_X| \leq np+n(1-p)^{\delta+1}$. The set $U = X \cup Y_X$ is clearly a dominating set of G whose cardinality is at most this size.

For convenience we bound $1 - p \leq \mathrm{e}^{-p}$ (this holds for all nonnegative p and is a fairly close bound when p is small) to give the simpler bound

$$|U| \leq np + n\mathrm{e}^{-p(\delta+1)}. \tag{1}$$

Take the derivative of the right-hand side with respect to p and set it equal to zero. The right-hand side is minimized at

$$p = \frac{\ln(\delta + 1)}{\delta + 1}. \tag{2}$$

Formally, we set p equal to this value in the first line of the proof. We now have $|U| \leq n[1 + \ln(\delta + 1)]/(\delta + 1)$ as claimed.

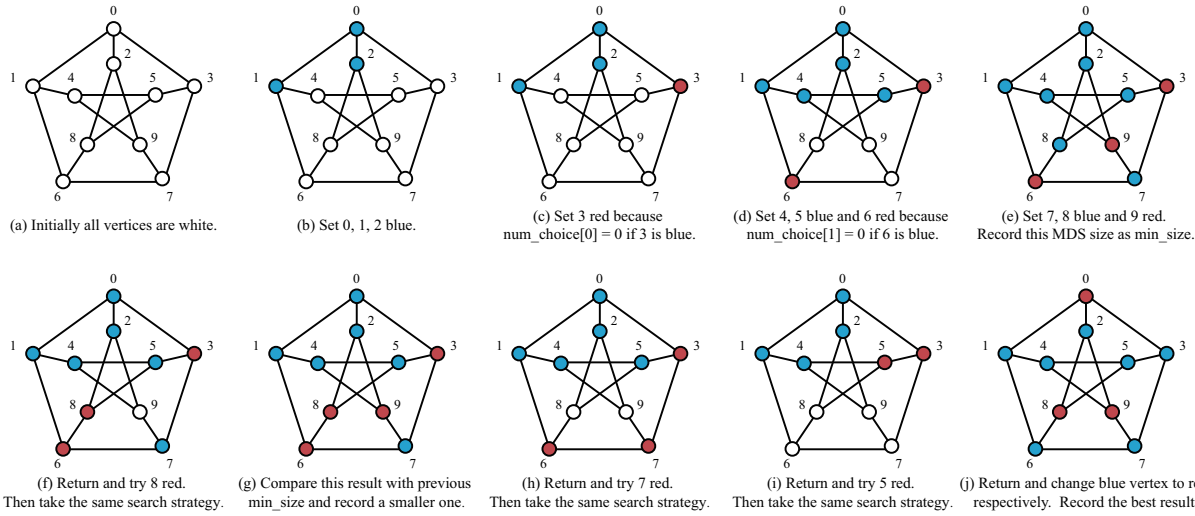
III. The MDS-GA Algorithm

In this section, we design an algorithm called MDS-GA that computes MDS based on greedy search [3]. We compare our results with the upper bounds proposed in Theorem 1.

A. Strategies and Algorithms

Table I illustrates the definitions of the parameters used in codes. We represent the relationship of vertices and edges by adjacency matrix [4]. The process of exhaustive backtrack is summarized in Algorithm 1 and Algorithm 2.

Fig.1 shows an example the schematics of greedy search. The red vertices are include in dominating set. The blue ones are excluded from dominating set. The white vertices are undecided. We first try making the vertices blue and then red. Before backtracking, change it back to white. All vertices v have a status (color), and num_choice[v] is at least 1 to ensure to be dominated. In Fig.1(a), all vertices are set white initially. Vertices 0, 1, 2 are set to blue successively. Try to set vertex 3 blue and then num_choice[0]=0, so go back and color 3 red. The coloring strategy of the remaining white vertices is the same. When all the vertices are colored, record the results.



(a) Initially all vertices are white. (b) Set 0, 1, 2 blue. (c) Set 3 red because num_choice[0] = 0 if 3 is blue. (d) Set 4, 5 blue and 6 red because num_choice[1] = 0 if 6 is blue. (e) Set 7, 8 blue and 9 red. Record this MDS size as min_size.

(f) Return and try 8 red. Then take the same search strategy. (g) Compare this result with previous min_size and record a smaller one. (h) Return and try 7 red. Then take the same search strategy. (i) Return and try 5 red. Then take the same search strategy. (j) Return and change blue vertex to red respectively. Record the best result.

Fig. 1. Schematics of greedy search process: The red vertices are include in dominating set. The blue ones are excluded from dominating set. The white vertices are undecided.

TABLE I
NOTATIONS FOR PROGRAM REPRESENTATION

Parameter	Definition
n	the order of given graph $G(V, E)$
adj_matrix	adjacency-matrix representation
Delta	the maximal degree $\Delta(G)$
delta	the minimal degree $\delta(G)$
num_choice[v]	number of times v could be dominated if all white vertices were red ones.
dom	$1 \times n$ vector whose element is 0 for white, -1 for blue and 1 for red
min_dom	the minimal dominating set
min_size	the size of minimal dominating set

Algorithm 1 The min_dom_set function.

1: **If** u is the last vertex:
2: **Return** results.
3: **Else**:
4: **If** try vertex u as blue successfully:
5: decrease num_choice[v] for each neighbour v of u as recorded in adj_matrix;
6: **Else**: Set u red;
7: **Return** min_dom_set($u + 1$).

Algorithm 2 The greedy_search function.

1: Call min_dom_set function with $u = 0$.
2: Backtrack to the latest blue vertex (denoted as u) and change it to red.
3: Change vertices included in backtracking path back to white.
4: Call min_dom_set function with updating u.
5: Compare and record the best result.

Then go back to the latest blue vertex, change it to red (like Fig.1(f)). Set the subsequent vertices white and try to color them blue again.

B. Implementation and Performance

We establish a case study with graph shown in Fig.1, where the adjacency matrix is

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{3}$$

Table II shows the choice of dominating set step-by-step the of Fig.1. The size of the MDS is 3. The derived upper bound of given graph is 5. We also change the order of the graph such as taking $n = 100$, which satisfies Theorem 1 as well.

IV. CONCLUSIONS

In this paper, we solve the problem to find the MDS by GA and the results of this program satisfies the upper bound of MDS proposed in Theorem 1.

REFERENCES

[1] N. Alon and J. H. Spencer, *The probabilistic method*. John Wiley & Sons, 2004.
[2] V. G. Vizing, "Some unsolved problems in graph theory," *Russian Mathematical Surveys*, vol. 23, no. 6, p. 125, 1968.
[3] "Csc 425/520 class notes: Spring 2014 - lecture 3: A simple dominating set algorithm." http://webhome.cs.uvic.ca/~wendym/courses/425/14/, accessed May. 2, 2019.
[4] A. J. Ryba and R. L. K. Kruse, *Data structures and program design in C++*. Prentice Hall, 1999.

TABLE II
IMPLEMENTATION OF GIVEN GRAPH

u	$\Gamma(u)$	$\Gamma(u) \cap \{v \in G \mid v \text{ is blue}\}$	dom	num_choice
0	[1 2 3]	[]	[-1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]	[3 2 2 2 3 3 3 3 3 3]
1	[0 4 6]	[0]	[-1. -1. 0. 0. 0. 0. 0. 0. 0. 0.]	[2. 2. 2. 2. 2. 3. 2. 3. 3. 3.]
2	[0 8 9]	[0]	[-1. -1. -1. 0. 0. 0. 0. 0. 0. 0.]	[1. 2. 2. 2. 2. 3. 2. 3. 2. 2.]
3	[0 5 7]	[0]	[-1. -1. -1. 1. 0. 0. 0. 0. 0. 0.]	[1. 2. 2. 2. 2. 3. 2. 3. 2. 2.]
4	[1 5 9]	[1]	[-1. -1. -1. 1. -1. 0. 0. 0. 0. 0.]	[1. 1. 2. 2. 2. 2. 2. 3. 2. 1.]
5	[3 4 8]	[4]	[-1. -1. -1. 1. -1. -1. 0. 0. 0. 0.]	[1. 1. 2. 1. 1. 2. 2. 3. 1. 1.]
6	[1 7 8]	[1]	[-1. -1. -1. 1. -1. -1. 1. 0. 0. 0.]	[1. 1. 2. 1. 1. 2. 2. 3. 1. 1.]
7	[3 6 9]	[]	[-1. -1. -1. 1. -1. -1. 1. -1. 0. 0.]	[1. 1. 2. 0. 1. 2. 1. 3. 1. 0.]
8	[2 5 6]	[2 5]	[-1. -1. -1. 1. -1. -1. 1. -1. -1. 0.]	[1. 1. 1. 0. 1. 1. 0. 3. 1. 0.]
9	[2 4 7]	[2 4 7]	[-1. -1. -1. 1. -1. -1. 1. -1. -1. 1.]	[1. 1. 1. 0. 1. 1. 0. 3. 1. 0.]
8	[2 5 6]	[2 5]	[-1. -1. -1. 1. -1. -1. 1. -1. 1. 0.]	[1. 1. 1. 0. 1. 1. 0. 3. 1. 0.]
9	[2 4 7]	[2 4 7]	[-1. -1. -1. 1. -1. -1. 1. -1. 1. 1.]	[1. 1. 1. 0. 1. 1. 0. 3. 1. 0.]
7	[3 6 9]	[]	[-1. -1. -1. 1. -1. -1. 1. 1. 0. 0.]	[1. 1. 2. 0. 1. 2. 1. 3. 1. 0.]
8	[2 5 6]	[2 5]	[-1. -1. -1. 1. -1. -1. 1. 1. -1. 0.]	[1. 1. 1. 0. 1. 1. 0. 3. 1. 0.]
9	[2 4 7]	[2 4]	[-1. -1. -1. 1. -1. -1. 1. 1. -1. 1.]	[1. 1. 1. 0. 1. 1. 0. 3. 1. 0.]
5	[3 4 8]	[4]	[-1. -1. -1. 1. -1. 1. 0. 0. 0. 0.]	[1. 1. 2. 1. 1. 2. 2. 3. 1. 1.]
6	[1 7 8]	[1]	[-1. -1. -1. 1. -1. 1. 1. 0. 0. 0.]	[1. 1. 2. 1. 1. 2. 2. 3. 1. 1.]
7	[3 6 9]	[]	[-1. -1. -1. 1. -1. 1. 1. -1. 0. 0.]	[1. 1. 2. 0. 1. 2. 1. 3. 1. 0.]
8	[2 5 6]	[2]	[-1. -1. -1. 1. -1. 1. 1. -1. -1. 0.]	[1. 1. 1. 0. 1. 1. 0. 3. 1. 0.]
9	[2 4 7]	[2 4 7]	[-1. -1. -1. 1. -1. 1. 1. -1. -1. 1.]	[1. 1. 1. 0. 1. 1. 0. 3. 1. 0.]
4	[1 5 9]	[1]	[-1. -1. -1. 1. 1. 0. 0. 0. 0. 0.]	[1. 1. 2. 2. 2. 2. 2. 3. 2. 1.]
5	[3 4 8]	[]	[-1. -1. -1. 1. 1. -1. 0. 0. 0. 0.]	[1. 1. 2. 1. 1. 2. 2. 3. 1. 1.]
6	[1 7 8]	[1]	[-1. -1. -1. 1. 1. -1. 1. 0. 0. 0.]	[1. 1. 2. 1. 1. 2. 2. 3. 1. 1.]
7	[3 6 9]	[]	[-1. -1. -1. 1. 1. -1. 1. -1. 0. 0.]	[1. 1. 2. 0. 1. 2. 1. 3. 1. 0.]
8	[2 5 6]	[2 5]	[-1. -1. -1. 1. 1. -1. 1. -1. -1. 0.]	[1. 1. 1. 0. 1. 1. 0. 3. 1. 0.]
9	[2 4 7]	[2 7]	[-1. -1. -1. 1. 1. -1. 1. -1. -1. 1.]	[1. 1. 1. 0. 1. 1. 0. 3. 1. 0.]
2	[0 8 9]	[0]	[-1. -1. 1. 0. 0. 0. 0. 0. 0. 0.]	[1. 2. 2. 2. 2. 3. 2. 3. 2. 2.]
3	[0 5 7]	[0]	[-1. -1. 1. 1. 0. 0. 0. 0. 0. 0.]	[1. 2. 2. 2. 2. 3. 2. 3. 2. 2.]
4	[1 5 9]	[1]	[-1. -1. 1. 1. -1. 0. 0. 0. 0. 0.]	[1. 1. 2. 2. 2. 2. 2. 3. 2. 1.]
5	[3 4 8]	[4]	[-1. -1. 1. 1. -1. -1. 0. 0. 0. 0.]	[1. 1. 2. 1. 1. 2. 2. 3. 1. 1.]
6	[1 7 8]	[1]	[-1. -1. 1. 1. -1. -1. 1. 0. 0. 0.]	[1. 1. 2. 1. 1. 2. 2. 3. 1. 1.]
7	[3 6 9]	[]	[-1. -1. 1. 1. -1. -1. 1. -1. 0. 0.]	[1. 1. 2. 0. 1. 2. 1. 3. 1. 0.]
8	[2 5 6]	[5]	[-1. -1. 1. 1. -1. -1. 1. -1. -1. 0.]	[1. 1. 1. 0. 1. 1. 0. 3. 1. 0.]
9	[2 4 7]	[4 7]	[-1. -1. 1. 1. -1. -1. 1. -1. -1. 1.]	[1. 1. 1. 0. 1. 1. 0. 3. 1. 0.]
1	[0 4 6]	[0]	[-1. 1. 0. 0. 0. 0. 0. 0. 0. 0.]	[2. 2. 2. 2. 2. 3. 2. 3. 3. 3.]
2	[0 8 9]	[0]	[-1. 1. -1. 0. 0. 0. 0. 0. 0. 0.]	[1. 2. 2. 2. 2. 3. 2. 3. 2. 2.]
3	[0 5 7]	[0]	[-1. 1. -1. 1. 0. 0. 0. 0. 0. 0.]	[1. 2. 2. 2. 2. 3. 2. 3. 2. 2.]
4	[1 5 9]	[]	[-1. 1. -1. 1. -1. 0. 0. 0. 0. 0.]	[1. 1. 2. 2. 2. 2. 2. 3. 2. 1.]
5	[3 4 8]	[4]	[-1. 1. -1. 1. -1. -1. 0. 0. 0. 0.]	[1. 1. 2. 1. 1. 2. 2. 3. 1. 1.]
6	[1 7 8]	[]	[-1. 1. -1. 1. -1. -1. -1. 0. 0. 0.]	[1. 0. 2. 1. 1. 2. 2. 2. 0. 1.]
7	[3 6 9]	[6]	[-1. 1. -1. 1. -1. -1. -1. -1. 0. 0.]	[1. 0. 2. 0. 1. 2. 1. 2. 0. 0.]
8	[2 5 6]	[2 5 6]	[-1. 1. -1. 1. -1. -1. -1. -1. 1. 0.]	[1. 0. 2. 0. 1. 2. 1. 2. 0. 0.]
9	[2 4 7]	[2 4 7]	[-1. 1. -1. 1. -1. -1. -1. -1. 1. 1.]	[1. 0. 2. 0. 1. 2. 1. 2. 0. 0.]
0	[1 2 3]	[]	[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]	[3 3 3 3 3 3 3 3 3 3]
1	[0 4 6]	[]	[1. -1. 0. 0. 0. 0. 0. 0. 0. 0.]	[2. 3. 3. 3. 2. 3. 2. 3. 3. 3.]
2	[0 8 9]	[]	[1. -1. -1. 0. 0. 0. 0. 0. 0. 0.]	[1. 3. 3. 3. 2. 3. 2. 3. 2. 2.]
3	[0 5 7]	[]	[1. -1. -1. -1. 0. 0. 0. 0. 0. 0.]	[0. 3. 3. 3. 2. 2. 2. 2. 2. 2.]
4	[1 5 9]	[1]	[1. -1. -1. -1. -1. 0. 0. 0. 0. 0.]	[0. 2. 3. 3. 2. 1. 2. 2. 2. 1.]
5	[3 4 8]	[3 4]	[1. -1. -1. -1. -1. -1. 0. 0. 0. 0.]	[0. 2. 3. 2. 1. 1. 2. 2. 1. 1.]
6	[1 7 8]	[1]	[1. -1. -1. -1. -1. -1. -1. 0. 0. 0.]	[0. 1. 3. 2. 1. 1. 2. 1. 0. 1.]
7	[3 6 9]	[3 6]	[1. -1. -1. -1. -1. -1. -1. -1. 0. 0.]	[0. 1. 3. 1. 1. 1. 1. 1. 0. 0.]
8	[2 5 6]	[2 5 6]	[1. -1. -1. -1. -1. -1. -1. -1. 1. 0.]	[0. 1. 3. 1. 1. 1. 1. 1. 0. 0.]
9	[2 4 7]	[2 4 7]	[1. -1. -1. -1. -1. -1. -1. -1. 1. 1.]	[0. 1. 3. 1. 1. 1. 1. 1. 0. 0.]