Randomized Greedy Algorithms for Covering Problems

Wanru Gao
Optimisation and Logistics
School of Computer Science
The University of Adelaide
Adelaide, Australia

Tobias Friedrich
Algorithm Engineering
Hasso Plattner Institute
Potsdam, Germany

Frank Neumann
Optimisation and Logistics
School of Computer Science
The University of Adelaide
Adelaide, Australia

Christian Hercher
Mathematics and Didactics
University of Flensburg
Flensburg, Germany

ABSTRACT

Greedy algorithms provide a fast and often also effective solution to many combinatorial optimization problems. However, it is well known that they sometimes lead to low quality solutions on certain instances. In this paper, we explore the use of randomness in greedy algorithms for the minimum vertex cover and dominating set problem and compare the resulting performance against their deterministic counterpart. Our algorithms are based on a parameter γ which allows to explore the spectrum between uniform and deterministic greedy selection in the steps of the algorithm and our theoretical and experimental investigations point out the benefits of incorporating randomness into greedy algorithms for the two considered combinatorial optimization problems.

CCS CONCEPTS

• **Theory of computation → Random search heuristics**;

KEYWORDS

Random search heuristics, Greedy Algorithms, Covering Problems

ACM Reference Format:
Wanru Gao, Tobias Friedrich, Frank Neumann, and Christian Hercher. 2018. Randomized Greedy Algorithms for Covering Problems. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3205455.3205542

1 INTRODUCTION

Randomization plays a key role when designing high performing algorithms. Many algorithmic studies in theoretical computer science show that incorporating the right dose of randomness can be crucial for solving computational problems. Let us recall three examples: (1) polynomial identity testing has only exponential time deterministic algorithms, but simple polynomial-time randomized algorithms, (2) counting the number of tilings of an n-by-n grid by dominos is a problem, where the best known deterministic algorithms are exponential time, but there are efficient randomized approximation algorithms [12]), and (3) the dining philosophers problem from distributed computations always ends up in a deadlock for deterministic programs [4], but simple randomized algorithms can break the problem's symmetry [11].

There are similar trends within heuristic optimization: Many popular heuristic search algorithms strongly rely on random decisions as key components of their algorithms. Examples are evolutionary algorithms [6], ant colony optimization [5] or greedy randomized adaptive search procedures [7].

In this paper, we study the impact of using randomness in greedy algorithms. (Deterministic) greedy algorithms often provide an effective and fast approach when dealing with combinatorial optimization problems. On the other hand, it is well-known that they have a bad approximation behavior on problems such as the minimum vertex cover problem for some instances. A certain move which is the best option for the current stage may only lead to the local optimum rather than the global optimum. Randomizing the greedy step might help to prevent the algorithm from producing bad solutions for these instances.

We explore the use of randomization in simple greedy algorithms for two well-known NP-hard combinatorial optimization problems, namely the minimum vertex cover (MVC) and minimum dominating set (MDS) problem [8]. For MVC it is well known that the greedy algorithm picking in each step a node that covers the largest number of still uncovered edges, obtains only a $\Theta(\log n)$-approximation of an optimal solution for a certain class of bipartite graphs. We provide a theoretical analysis of a randomized greedy approach and show that choosing the next node to be added to the set proportional to the number of uncovered edges that it can cover, results in expectation a 2-approximation for every problem instance of MVC. Furthermore, we provide a trade-off result in terms of a number of repeated runs and approximation quality and show that an α-approximation, $1 \leq \alpha < 2$, is obtained in $2\ln(2) \cdot 2^{(2-\alpha)\cdot|\mathrm{OPT}|}$ repeated runs with probability at least $1/2$.

In order to gain some insight about the randomized selection scheme in practical aspect, we embed it into the greedy algorithm

GECCO '18, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205542

for another covering problem, which is the minimum dominating set problem. The greedy approach for this problem selects the node with the maximum number of uncovered neighbours. We take this problem as another example where the randomized selection scheme is applied to the deterministic greedy approach and analyse the experimental results.

The test cases cover well-known benchmark problems and large real world graphs. The experimental results from the randomized greedy algorithms are compared with the deterministic greedy algorithm. We show that the deterministic greedy algorithm easily gets stuck in locally optimal solution. Due to its nature of favoring the best option at each step, the deterministic greedy approach highly likely ends up with optimizing along the wrong direction. With the randomized selection scheme, the algorithm keeps the possibility of getting away from the local optimum.

The experimental results demonstrate that the randomized greedy approach improves the performance of the deterministic greedy approach in most test cases. With simple repeated runs, the randomized approach can provide better results for different covering problems, especially for large and sparse graphs.

There exist some state-of-the-art algorithms for MVC that are based on the simple greedy approach [1, 2]. The randomized mechanism could provide ideas to further improve them.

The outline of the paper is as follows. In Section 2, we introduce the general set up for greedy algorithms and the kind of randomization that we consider. In Section 3, we study the randomized greedy approach for the minimum vertex cover problem from both theoretical and experimental perspective. We investigate the randomized greedy approach for minimum dominating set through experimental investigations in Section 4. Finally, we finish with some concluding remarks.

2 GREEDY ALGORITHMS AND RANDOMIZATION

Greedy algorithms play an important role for solving many optimization problems. They often provide a fast solution to the problem at hand which can be implemented in a very simple way.

Consider an optimization problem P where a potential solution consists of a subset of n components $C = \{C_1, \ldots, C_n\}$. A greedy algorithm starts with an empty set $S = \emptyset$ and adds in each step t a component $C_i \in C^t$ to the solution set S. Here C^t denotes the set of available components at time step t. To do this, a reward $r(C_i) \geq 0$, $C_i \in C^t$, is assigned to each available component.

The deterministic greedy algorithm chooses in step t a component with maximum reward $r^t(C_i)$, i.e

$$C_i = \arg \max_{C_j \in C^t} r^t(C_j).$$

We can randomize the deterministic greedy approach by selecting in step t a component $C_i \in C^t$ according to the probability distribution given by

$$p(C_i) = r(C_i)^\gamma / \sum_{C_j \in C^t} r(C_j)^\gamma \quad (1)$$

Here $\gamma \geq 0$ is a parameter that determines the amount of greediness.

Algorithm 1: Greedy algorithm for Vertex Cover

```
S := ∅;
repeat
    Choose a node u ∈ arg max_{v∈G[S]}{deg(v, G[S])}}
      covering a maximum number of uncovered edges. ;
    S := S ∪ {u}
until S is a vertex cover of G;
Return S;
```

For the special case of $\gamma = 0$, a component $C_i \in C^t$ is chosen uniformly at random. $\gamma = 1$ implies that the probability of choosing a component is proportional to its reward, and $\gamma \to \infty$ approaches the behavior of the deterministic greedy algorithm where ties are broken uniformly at random.

Note that if $\gamma > 0$, we can use $r(C_i) = 0$ for $C_i \notin C^t$ and work with the set C instead of C^t in Equation 1 and obtain the same probability distribution.

3 VERTEX COVER PROBLEM

Vertex Cover is one of the most fundamental and most studied combinatorial optimization problems. Given an undirected graph $G = (V, E)$, the goal is to find a minimum set of vertices V' such that every edge $e \in E$ is covered, i.e. $e \cap V' \neq \emptyset$ holds for all $e \in E$. We denote by $deg(v, G)$ the degree of node v in graph G. Furthermore, we denote by $G[S] = (V, E')$ where $E' = E \setminus \{e \mid e \cap S \neq \emptyset\}$ the subgraph obtained from G by removing all edges covered by nodes in the vertex set S.

Given an algorithm A for a minimization problem P, the (worst case) approximation ratio of algorithm A is given by $\max_I A(I)/OPT(I)$ where $A(I)$ denotes the objective value achieved by algorithm A and $OPT(I)$ denotes the optimal objective value for instance I. The expected (worst case) approximation ratio is the worst case ratio $\max_I E[A(I)]/OPT(I)$ where $E[A(I)]$ denotes the the expected objective value that algorithm A achieves on instance I.

3.1 Greedy Algorithms

We study simple *vertex-based* greedy algorithms. It is folklore that a deterministic vertex-based greedy algorithm (cf. Alg. 1) for vertex cover only achieves a $\Theta(\log n)$-approximation. Hence our aim is understanding how randomization can help the greedy algorithm.

For this, we consider a randomized (vertex-based) greedy algorithm which is included as Alg. 2. Our randomized greedy algorithm starts with an empty set of nodes and adds vertices until a vertex cover has been obtained. In each step, an uncovered node is selected according to the probability based on the degree of the node in the uncovered graph $G[S]$. While the deterministic vertex-based greedy algorithm can be fooled to a $\Theta(\log n)$-approximation, the randomized vertex-based greedy algorithm achieves in expectation a 2-approximation:

3.2 Analysis

We now analyze the randomized greedy algorithm for the minimum vertex cover problem. We do this for $\gamma = 1$. The following results

Algorithm 2: Randomized greedy algorithm for Vertex Cover

```
S := ∅;
repeat
    Choose a node u in G[S] according to probability
      p(u) = deg(v, G[S])^γ / Σ_{u∈G[S]} deg(v, G[S])^γ;
    S := S ∪ {u}
until S is a vertex cover of G;
Return S;
```

show that the randomized greedy algorithm with $\gamma = 1$ achieves an expected approximation ratio of 2.

Theorem 3.1. *Let $\gamma = 1$, then the randomized greedy algorithm given in Algorithm 2 has an expected approximation ratio of 2.*

Proof. Let OPT be a minimimum vertex cover. The probability of selecting a node $v \in$ OPT in the next step is given by

$$\sum_{v \in \mathrm{OPT}} deg(v, G[S]) / \sum_{v \in V} deg(v, G[S])$$

.

We have $\sum_{v \in \mathrm{OPT}} deg(v, G[S]) \geq E[S]$ as all nodes in OPT are still covering all edges and $\sum_{v \in V} deg(v, G[S]) = 2 \cdot E[S]$. This implies

$$\sum_{v \in \mathrm{OPT}} deg(v, G[S]) / \sum_{v \in V} deg(v, G[S]) \geq 1/2$$

.

A vertex cover is produced if all nodes of OPT are selected, but may be produced earlier. As in each step a node in OPT is selected with probability at least 1/2 unless a vertex cover has already been found, the expected number of nodes that the algorithm selects until a vertex cover is produced is at most $2 \cdot |\mathrm{OPT}|$. □

Repeated runs are often used for randomized algorithms to generate a solution of good quality. As a node of OPT is always included with probability 1/2 as long as a vertex cover has not been obtained, the previous idea can be generalized when considering repeated runs.

Theorem 3.2. *Let $1 \leq \alpha < 2$ and $\gamma = 1$. Then the randomzed greedy algorithm given in Algorithm 2 obtains with probability at least 1/2 in $2\ln(2) \cdot 2^{(2-\alpha)\cdot|\mathrm{OPT}|}$ runs an approximation of α.*

Proof. The number of nodes in OPT selected in the first $x = 2(\alpha-1)|\mathrm{OPT}|$ steps of Algorithm 2 is at least $\frac{x}{2} = (\alpha-1)|\mathrm{OPT}|$ with probability at least $\frac{1}{2}$. In each step the probability of selecting a node in OPT is at least $\frac{1}{2}$ and thus a binomially distributed random variable $B(x, \frac{1}{2})$ is a lower bound for the number of selected nodes in OPT. But because of symmetry of $B(x, \frac{1}{2})$ the probability of at least $\frac{x}{2}$ selected nodes in *OPT* is at least $\frac{1}{2}$.

If at least $\frac{x}{2} = (\alpha-1)|\mathrm{OPT}|$ nodes in OPT were selected after the first x steps a vertex cover is found if all the other at most $y = (2-\alpha)|\mathrm{OPT}|$ nodes are selected, too. The probability of selecting them in the next y steps of the algorithm is at least $\left(\frac{1}{2}\right)^y$. Therefore Algorithm 2 produces a vertex cover of size at most $x + y = \alpha|\mathrm{OPT}|$ with probability $P \geq \left(\frac{1}{2}\right)^{1+(2-\alpha)|\mathrm{OPT}|}$.

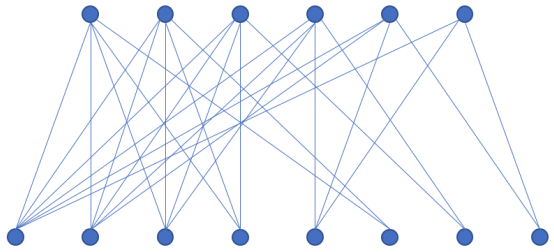

Figure 1: The worst case vertex cover of greedy algorithm. The global optimum solution is the upper vertex set but the greedy algorithm can only find the lower set which is larger than the upper one.

After $\frac{\ln(2)}{P} \leq 2\ln(2) \cdot 2^{(2-\alpha)|\mathrm{OPT}|}$ repeated runs of Algorithm 2 the probability that neither of these runs found a vertex cover of size at most $\alpha|\mathrm{OPT}|$ is at most $(1-P)^{\frac{\ln(2)}{P}} < e^{-\ln(2)} = \frac{1}{2}$. □

We have shown from a theoretical perspective that repeated runs of the randomized greedy algorithm with $\gamma = 1$ are highly beneficial. In the following, we investigate the behaviour of the randomized greedy algorithm for a wide range of settings of γ in connection with repeated runs.

3.3 Experimental Results

In this section, we include our findings from experiments for applying the randomization mechanism to greedy algorithm of the minimum vertex cover problem.

All of the experiments are executed on a machine with 48-core Authentic AMD 2.80 GHz CPU and 128 GByte RAM; note that the program uses only a single core.

There are many well-known benchmarks for covering problems. In this paper, the different approaches are evaluated based on tests on selected graphs from the DIMACS benchmark, BHOSLIB problems and some real world graphs.

The DIMACS benchmark is one of the well-known benchmarks used to test algorithms working on graph problems. It is a set of challenging problems coming from the Second DIMACS Implementation Challenge for Maximum Clique, Graph Coloring and Satisfiability [10].

The *BHOSLIB* (Benchmarks with Hidden Optimum Solutions) problems are generated from translating the binary Boolean Satisfiability problems randomly generated based on the model RB [16]. These instances have been proven to be hard to solve, both theoretically and practically.

Some undirected unweighted real world graphs are selected from the Network Data Repository [14]. These graphs are sparser and larger in size than the benchmark problems. Since they are extracted from real world networks, they reflect the characteristics of realistic networks and attract more and more attention of the researchers.

The two algorithms 1 and 2 are implemented in Java and compiled with JDK 8. The running time of the two algorithms on the same test case is comparable, therefore not reported.

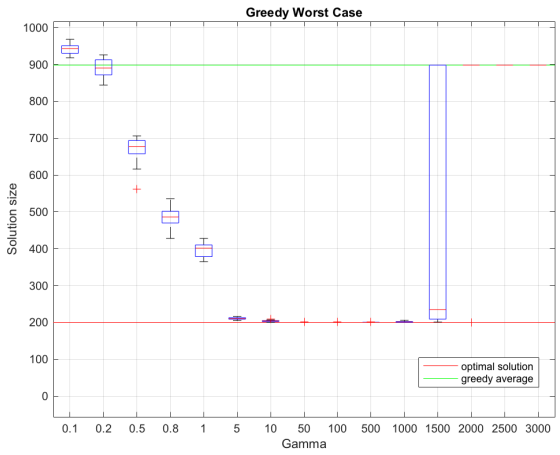

Figure 2: The experimental results from the greedy algorithm with randomization for an example worst case graph. The global optimum solution is of size 200 and indicated with the red horizontal line. The minimum vertex cover found by deterministic greedy algorithm is indicated with a green line.

3.3.1 Worst case example of deterministic greedy algorithm. There exists a counterexample which is widely used to disprove the greedy vertex cover algorithm is 2-approximated. As shown in Figure 1, consider a bipartite graph $G_b = (U \cup D, E)$, where $|U| = n$. For $i = n, \cdots, 2$, add set D_i with $\lfloor n/i \rfloor$ vertices to set D and make all vertices in D_i have degree i by connecting them with distinct node in set U.

In the worst case for greedy algorithm, the original greedy algorithm is not able to achieve a 2-approximated solution for such a bipartite graph. The global optimum for this problem is the upper set U but Algorithm 1 is only able to find the lower set D. In Algorithm 2, the introduced randomness makes it possible to achieve the global optimum solution for this kind of problem.

As an example, the two algorithms are run on a bipartite graph with two sets of vertices where each set has 200 and 898 vertices. The results with different parameter γ in Algorithm 2 comparing with the greedy approach are shown in Figure 2. The red horizontal line represents the global optimum result which is 200 and the green line indicates the solution size from deterministic greedy approach which is 898 in numerical value. The boxplots come from 10 independent repeated runs.

As analyzed in the previous section, when the value of γ is very small, the algorithm performs like a uniformly random algorithm. In the figure, when $\gamma \leq 1$, the solution size from Algorithm 2 decreases as γ gets larger. From the figure, it is clear that with a γ in the range of 20 to 1000, the randomized greedy algorithm is able to find the global optimum solution. When a very large γ is given, the algorithm tends to perform like the deterministic greedy algorithm which only be able to get a solution of size 898.

This experiment shows that with a reasonable randomness, the greedy algorithm is able to escape the local optima.

3.3.2 Benchmarks and real world graphs. These two algorithms are then tested on some more general test cases including the DIMACS benchmarks, BHOSLIB problems and small real world graphs. The experimental results are collected from 20 independent repeated runs each case. The minimum and average solution size from the 20 runs of Algorithm 2 are reported.

Table 1 shows the experimental results from the deterministic greedy algorithm and the greedy algorithm with randomized selection scheme on some selected vertex cover problem. The γ values are selected in the range of $[1, 500]$. The results from the randomized greedy approach that are better than those from the deterministic approach are in bold and the best solutions are colored in blue.

The DIMACS benchmarks and BHOSLIB problems are widely used to test vertex cover algorithms. These test cases are designed to be hard test cases. The deterministic greedy approach can obtain nearly optimum solutions for most cases. For the brock family, the randomized approach outperforms the deterministic approach in 5 out of 6 graphs when $\gamma \leq 20$. The randomized algorithm has a better average performance in most test cases when γ increases.

The brock graphs are designed with the aim to defeat greedy heuristics by including low degree nodes into the global optimum vertex cover. The deterministic greedy approach tends to select the node with higher degree during the process, which is easily trapped in the local optimum. The randomized approach improves that.

The real world graphs are usually large but sparse graphs. The randomized greedy algorithm shows the same performance or weakly outperforms the deterministic approach in most graphs with $\gamma = 10$. Increasing γ improves the performance of Algorithm 2 in most cases.

4 DOMINATING SET PROBLEM

Another example problem studied in this paper is the problem of finding a dominating set of a given graph. The dominating sets problem has many real world applications in different areas, especially in computer and mobile networks [3, 15]. It has been proven that finding a minimal dominating set is NP-hard [8, 9]. In this research, we focus on the approximated solutions.

Definition 4.1 (Dominating Set Problem). Given an undirected graph $G = (V, E)$, a dominating set D is a subset $D \subseteq V$ where for all $v \in V$, either $v \in D$ or a neighbor u of v is in D.

One of the approximated algorithms for finding dominating set is the greedy algorithm. The algorithm starts with an empty set and adds nodes until a dominating set is achieved. The node to be included is decided based on the number of uncovered nodes in the neighbourhood.

Definition 4.2. In the Dominating Set problem, We call nodes in set D black, nodes that have neighbor in D gray and all uncovered nodes white. Assume $W(v)$ represents the set of white nodes among the direct neighbors of v. $w(v) = |W(v)|$ is called the span of v.

4.1 Greedy Algorithms

The simple greedy algorithm starts with an empty set and covers one of the nodes with the maximum span in each iteration until a dominating set is found. The span of each node is updated every

Instance name	DGA	Randomized greedy																	
		$\gamma = 1$		$\gamma = 5$		$\gamma = 10$		$\gamma = 20$		$\gamma = 30$		$\gamma = 50$		$\gamma = 100$		$\gamma = 200$		$\gamma = 500$	
		min	mean	min	mean	min	mean	min	mean	min	mean	min	mean	min	mean	min	mean	min	mean
brock200_2	192	196	196.9	192	193.8	192	193.2	**191**	192.4	**190**	**191.9**	**191**	192.1	**190**	**192.0**	**190**	193.0	**190**	**191.9**
brock200_4	188	193	194.7	189	190.6	**187**	188.4	**186**	**187.7**	**186**	**187.5**	**186**	**187.9**	**186**	**187.9**	**187**	188.3	**186**	**187.9**
brock400_2	379	389	392.7	384	386.1	380	383.3	379	381.8	380	382.0	**376**	381.0	379	381.3	**377**	381.0	379	380.7
brock400_4	380	390	392.4	383	386.2	380	383.0	**379**	381.7	**379**	381.2	**378**	380.1	**378**	380.0	**379**	380.3	**378**	380.5
brock800_2	788	793	794.5	**787**	789.4	**786**	**787.4**	**783**	**785.6**	**784**	**785.9**	**782**	**785.2**	**783**	**785.1**	**782**	**785.1**	**783**	**785.3**
brock800_4	787	793	794.8	789	790.2	**786**	787.5	**785**	**786.5**	**784**	**786.2**	**784**	**785.6**	**784**	**786.0**	**783**	**786.1**	**784**	**785.6**
C125.9	93	103	107.3	93	96.4	**92**	94.3	**91**	**92.9**	**92**	93.3	**92**	**92.8**	**92**	93.1	**92**	93.3	**92**	**92.9**
C500.9	453	475	480.8	457	463.6	453	459.1	454	457.7	454	457.4	453	457.3	**452**	456.8	**452**	456.6	**451**	456.5
frb45-21-1	913	928	930.6	917	920.6	914	917.6	914	916.2	**912**	915.6	913	915.1	913	914.6	**912**	914.8	**912**	914.9
frb45-21-2	916	928	930.6	916	919.5	**914**	917.9	**914**	916.9	**915**	916.4	**915**	916.7	**915**	916.6	**915**	916.9	**914**	916.5
frb45-21-3	917	926	929.7	918	920.3	**916**	918.5	**914**	917.1	**916**	917.0	**914**	917.2	**916**	917.4	**915**	917.3	**914**	917.1
frb45-21-4	913	927	929.9	916	918.9	**912**	915.1	**912**	915.1	913	914.6	**912**	914.2	**912**	913.6	**911**	913.8	**912**	913.4
frb45-21-5	917	927	931.3	919	921.2	**915**	917.9	**916**	917.5	**914**	**916.8**	**914**	**916.9**	**915**	**916.6**	**915**	**916.6**	**914**	**916.6**
soc-wiki-vote	411	489	503.4	415	419.9	411	415.3	412	414.0	**410**	414.1	411	414.2	**410**	414.2	411	413.0	411	414.1
web-polblogs	246	292	307.7	248	250.4	**245**	247.9	246	246.8	246	247.0	**245**	246.8	246	246.8	246	246.9	246	247.3
bio-celegans	255	288	298.9	257	262.3	**254**	258.8	256	258.8	256	258.8	255	258.3	255	258.1	255	258.4	256	258.3
bio-dmela	2667	3424	3476.3	2707	2719.6	**2661**	2673.4	**2662**	2667.6	**2663**	2668.0	**2662**	2668.2	**2662**	2667.0	**2662**	2668.0	**2664**	2668.9
soc-douban	8719	15371	15760.6	8737	8750.5	**8717**	8722.6	**8716**	8719.6	**8715**	8719.2	**8716**	**8718.8**	**8715**	**8718.2**	**8717**	**8718.6**	**8717**	**8718.5**
soc-epinions	9860	12111	12203.5	9966	9993.1	9865	9877.7	**9855**	9867.4	**9849**	9865.5	**9858**	9866.4	**9853**	9865.7	**9854**	9864.9	**9848**	9864.9

Table 1: The experimental results comparing the performance of deterministic greedy algorithm and the one with randomization on vertex cover problem. The results come from 20 independent repeated runs. The minimum vertex cover size and mean size are reported in the left and right column under each parameter setting. The minimum solutions for each problem are colored in blue.

Algorithm 3: Randomized greedy algorithm for the Dominating Set Problem

```
S := ∅;
repeat
    Choose a node u ∈ {x | w(x) = max_{u∈V}{w(u)}};
    S := S ∪ {u}
until ∄ white nodes;
Return S;
```

Algorithm 4: Randomized greedy algorithm for the Dominating Set Problem

```
S := ∅;
repeat
    Choose a node u in G according to probability
      p(u) = w(v)^γ / Σ_{u∈V} w(u)^γ;
    S := S ∪ {u}
until ∄ white nodes;
Return S;
```

iteration. The pseudo code is included as Algorithm 3. The deterministic greedy algorithm has a worst-case approximation ratio of $1 + \ln n$. Furthermore, it is NP-hard to approximate the minimum dominating set problem with a ratio of $c \cdot \ln n$, where $c < 1$ is some constant [13].

In order to study the power of randomization, a greedy algorithm with a parameter related probability of selecting white node is introduced. As shown in Algorithm 4, each white node is given a certain selection probability based on its span. The parameter γ is similar to the randomized greedy algorithm for vertex cover which controls the randomness. One white node is selected to be included to the solution set in one iteration until all nodes are covered.

With a proper γ value, the randomized greedy algorithm introduces randomness in the selection process and increases the possibility of escaping the local optima where the deterministic greedy approach may get stuck.

4.2 Experimental Results

The randomized greedy algorithm for dominating set problem with different parameter settings is tested on a range of unweighted undirected graphs, including DIMACS benchmarks and some middle to large-size real world graphs. Each instance is tested with different parameter settings for 10 repeated runs in order to gather statistics.

Both Algorithm 3 and 4 are implemented in Java and compiled with JDK 8. The hardware used for generating the experimental results is the same with the previous section.

Table 2 and 3 show the comparison of results from Algorithm 3 and 4. The parameter γ for the randomized approach is chosen from $\{1, 10, 20, 30, 50, 100, 200, 500, 1000\}$.

The first single column in the two tables reports the minimum dominating set size found by the deterministic greedy algorithm(DGA). Same as the vertex cover problem, the minimum results and average results from the randomized greedy algorithm are reported for each parameter γ in the tables. The results obtained by the randomized greedy approach that are better than those from the corresponding deterministic greedy approach are highlighted in bold. In this study, we do not focus on whether the algorithm can achieve the global optimum solutions or not. Therefore, the global optimum solution sizes are not reported.

The two algorithms are firstly tested on the well-known DIMACS benchmarks. There exist many different families in the DIMACS benchmarks. Initially, they are designed for the clique problem and

Instance name	DGA	Randomized greedy approach															
		$\gamma = 1$		$\gamma = 10$		$\gamma = 20$		$\gamma = 50$		$\gamma = 100$		$\gamma = 200$		$\gamma = 500$		$\gamma = 1000$	
		min	mean	min	mean	min	mean	min	mean	min	mean	min	mean	min	mean	min	mean
C125.9	15	22	25.3	**14**	15.8	**14**	**14.9**	**14**	15.0	**14**	**14.7**	**13**	**14.3**	**14**	15.0	**14**	15.1
C250.9	18	29	31.8	**17**	18.4	**17**	**17.2**	**16**	**16.9**	**16**	**17.2**	**16**	**17.1**	**16**	**17.2**	**16**	**17.4**
C500.9	21	35	37.3	22	23.9	21	21.8	**20**	21.0	**20**	21	21	21.2	**20**	21.1	21	21.1
C1000.9	25	40	43.8	28	29.4	25	26.1	**24**	25.1	**24**	**24.6**	**24**	**24.9**	**24**	25.0	**24**	**24.6**
C2000.5	7	9	10.6	8	8.9	7	7.9	7	7.1	7	7.0	7	7.0	7	7.0	7	7.0
C2000.9	29	47	50.9	34	35.3	31	31.3	29	29.3	**28**	**28.9**	**28**	**28.7**	**28**	**28.6**	**28**	**28.1**
C4000.5	8	10	11.6	9	9.8	9	9.0	8	8.0	8	8.0	**7**	**7.9**	8	8.0	8	8.0
brock200_2	4	6	7.6	5	5.4	4	4.9	4	4.3	4	4.2	4	4.0	4	4.0	4	4.0
brock200_4	6	9	11.0	7	7.7	6	6.9	**5**	6.3	6	6.6	6	6.6	6	6.9	6	6.7
brock400_2	10	15	17.6	11	11.5	10	10.1	**9**	**9.8**	**9**	**9.6**	**9**	**9.9**	**9**	**9.7**	**9**	**9.7**
brock400_4	10	15	16.9	11	11.7	10	10.2	**9**	**9.9**	**9**	**9.7**	**9**	**9.9**	**9**	**9.5**	**9**	**9.7**
brock800_2	8	12	13.4	10	10.2	9	9	8	8.5	8	8.0	8	8.2	8	8.3	8	8.0
brock800_4	8	12	13.7	10	10.4	8	9	8	8.5	8	8.2	8	8.2	8	8.1	8	8.0
keller4	6	8	9.0	6	6.6	6	6.5	**5**	6.0	6	6.0	**5**	6.1	**5**	**5.7**	**5**	6.0
keller5	11	15	17.0	12	12.5	11	11.4	**10**	11.1	11	11.1	11	11.3	**10**	**10.9**	**10**	11.2
keller6	18	26	28.7	19	20.3	19	19.0	**17**	**17.9**	18	18.0	**17**	**17.9**	**17**	18.0	**17**	**17.8**

Table 2: Experimental results from testing Algorithm 3 and 4 on dominating set problem from DIMACS benchmarks. The minimum and average sizes of the dominating set found in the 10 independent runs are shown in the left and right columns under each parameter setting. The minimum solutions for each problem are colored in blue.

intensionally to be hard clique instances with different characteristics. Later on, they are widely used to test other graph problems.

Most test cases in DIMACS benchmarks are small graphs with node number in the range of 5 to 1000. The deterministic greedy approach is already able to reach a small dominating set. The randomized greedy algorithm perform similarly as its deterministic counterpart. Therefore, we only report part of the benchmark set and eliminate the cases where both approaches can achieve the same good results.

The graph families reported in this paper are as follows:

- *C family*: the random graph generated with given number of nodes and an edge probability.
- *brock family*: the randomly generated graph with cliques designed to be hidden among relatively low-degree nodes.
- *keller family*: graphs generated based on Keller's conjecture on tilings using hypercubes.

From the statistics obtained from the experiments on sample DIMACS benchmarks, with $\gamma >= 10$, the randomized greedy approach achieves better solutions than the deterministic greedy approach in 2 out of 16 test cases. With γ no less than 50, the randomized greedy approach improves the solutions from the greedy approach in 10 out of 16 test cases. With $\gamma = 50$, the randomized greedy algorithm outperforms the deterministic approach in 10 out of 16 test cases and achieves the same minimum dominating sets in the other test cases. Compared to the results from deterministic greedy dominating set algorithm, the randomized approach with a larger parameter γ increases the possibility to further optimizing the solution even for the nearly optimal one.

The minimum dominating sets found in the ten repeated runs of randomized greedy algorithm are never worse than the results from the deterministic greedy algorithm. From the experimental results, with $\gamma \geq 100$, repeated randomized greedy approach is very likely to improve the solution found by its deterministic counterpart. In the case where no improvement is spotted, the randomized greedy approach maintains the minimum size of dominating set found by the deterministic approach. Increasing the γ value from 100, the approach with randomized selection scheme can further improve the size of the minimum dominating set in 4 out of the 16 test cases.

The improvement by introducing the randomization to the greedy approach is more obvious in the real world test cases. These graphs are extracted from realistic data gathered from social networks. Table 3 shows the results gathered from experiments on the social network graphs. The graphs in the social network have number of vertices and edges in the range of 60 to 600 000 and 150 to 4 000 000. Due to the characteristics of the social network, the graphs in this family are likely to include many clusters.

When $\gamma = 10$, the randomized greedy approach starts to produce comparable results with its deterministic counterpart. The randomized algorithm finds smaller or same size dominating set in 4 graphs which are of relatively small size. With γ increases, the quality of the results increases.

With $\gamma = 50$, the randomized greedy approach is able to obtain better minimum and average results in 12 and 9 out of the 13 test cases, respectively. The minimum dominating sets found by the randomized greedy approach are smaller by more than 10 nodes than the solutions found by the deterministic greedy approach in 6 test cases. Among them, there are 3 test cases where the improvement is more than 50 nodes.

For the challenging instance soc-flickr which has 513 969 nodes and 3 190 452 edges, the randomized greedy approach significantly outperforms the deterministic greedy algorithm. The minimum dominating set found by the randomized approach is 125 nodes smaller than the results from deterministic approach. Similar observation is found in the case of soc-gowalla graph which is another

Instance name	DGA	Randomized greedy approach															
		$\gamma = 10$		$\gamma = 20$		$\gamma = 30$		$\gamma = 50$		$\gamma = 100$		$\gamma = 200$		$\gamma = 500$		$\gamma = 1000$	
		min	mean	min	mean	min	mean	min	mean	min	mean	min	mean	min	mean	min	mean
soc-BlogCatalog	4899	4900	4903.7	**4898**	4899.6	**4897**	4899.4	**4896**	4900.7	**4897**	4899.8	**4897**	4899.7	**4898**	4900.8	**4898**	4900.2
soc-FourSquare	61345	61390	61405.8	**61319**	**61342.9**	**61319**	**61339.1**	**61308**	**61337.8**	**61318**	**61336.3**	**61308**	**61334.1**	**61323**	**61339.2**	**61318**	**61333.7**
soc-LiveMocha	1489	1491	1497.6	**1481**	**1487.8**	**1481**	**1485.4**	**1481**	**1487.2**	**1477**	**1484.8**	**1482**	**1487.1**	**1482**	**1486.5**	**1480**	**1485.7**
soc-brightkite	13087	13112	13123.9	**13072**	**13082.1**	**13068**	**13078.2**	**13070**	**13077.0**	**13068**	**13078.6**	**13070**	**13078.3**	**13072**	**13078.1**	**13072**	**13078.1**
soc-buzznet	134	**131**	134.3	**132**	134.0	**130**	**131.4**	**131**	**132.5**	**129**	**132.5**	**130**	**132**	**131**	**132.5**	**130**	**132.4**
soc-delicious	56059	56148	56175.2	**56015**	**56033.3**	**55994**	**56020.8**	**56002**	**56007.6**	**55989**	**56005.2**	**56008**	**56014.1**	**55999**	**56007.2**	**55994**	**56009.3**
soc-digg	66698	66744	66772.5	66744	66772.5	66734	66757.6	66741	66756.9	66722	66750.6	66739	66756.5	66706	66752.4	66724	66758
soc-dolphins	16	16	17.1	**15**	16.3	16	16.6	**15**	16.2	**15**	16.1	16	16.5	**15**	**15.6**	**15**	16.1
soc-douban	8367	8367	8369.0	**8366**	**8366.9**	**8366**	**8366.6**	**8365**	**8366.5**	**8366**	**8366.3**	**8365**	**8366.4**	**8366**	**8366.8**	**8366**	**8366.5**
soc-epinions	6504	6513	6519.9	**6486**	**6495.2**	**6492**	**6497.5**	**6485**	**6494.1**	**6485**	**6492.1**	**6485**	**6493.3**	**6482**	**6491.7**	**6489**	**6495**
soc-flickr	98786	98991	99020.1	**98740**	**98766.6**	**98661**	**98697.6**	**98681**	**98700.4**	**98696**	**98707.0**	**98675**	**98697.5**	**98677**	**98697.5**	**98696**	**98708.2**
soc-gowalla	42581	42752	42784.3	**42526**	**42554.2**	**42517**	**42541.6**	**42508**	**42525.9**	**42510**	**42528.9**	**42503**	**42526.0**	**42511**	**42529.9**	**42499**	**42532.7**
soc-wiki-Vote	216	**213**	216.6	**215**	216.4	**214**	216.0	**214**	216.0	**215**	216.3	**214**	217.3	**214**	216.4	**215**	216.3

Table 3: Comparison between deterministic greedy algorithm and the algorithm proposed in the paper with different parameter setting. The test cases are extracted from some real world graphs. The column with name DGA stores the solution size found by the deterministic greedy approach. The minimum solutions for each problem are colored in blue.

large graph. For large graphs, the randomized approach is much powerful in finding smaller dominating set.

In most of the cases, the choice of parameter γ does not affect the performance too much when it gets greater than 20. In 9 different test cases,the parameter γ with which randomized greedy approach finds the minimum solution sets fall into the range of 20 to 100. The performance of the randomized selection process is not sensitive to the different gamma value in that range.

The simple deterministic greedy approach is easy to get stuck on some local optima, especially in solving the graphs of large scale and sparse graphs. Our proposed algorithm which introduces the randomness to the selection scheme can improve the performance of simple greedy algorithm in these graphs. By simple repeated runs, the randomized greedy approach can obtain better results in many different test cases.

5 CONCLUSIONS

Greedy algorithms is one of the popular solutions to many optimization problems. As an easy algorithmic paradigm, it makes locally optimal choice at each step with the hope of achieving the global optimum in reasonable time. However, the drawback is obvious that it is easy to get stuck in local optimal solution. In this study, we examine the idea of introducing randomization into the selection stage of the greedy approach. We conduct theoretical and experimental analysis about how randomization can improve the deterministic greedy approach in two well-know NP-hard combinatorial optimization covering problems, which are minimum vertex cover and minimum dominating set problem. The results from analysis show that the randomized greedy approach improves the original deterministic greedy approach in most cases. The randomization is beneficial in getting away from the local optimum which traps the deterministic greedy algorithm. The proposed randomization mechanism should be able to be applied to other greedy algorithms for covering problems.

ACKNOWLEDGMENT

This work has been supported through Australian Research Council (ARC) grants DP140103400 and DP160102401.

REFERENCES

[1] Shaowei Cai. 2015. Balance Between Complexity and Quality: Local Search for Minimum Vertex Cover in Massive Graphs. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15)*. AAAI Press, 747–753. http://dl.acm.org/citation.cfm?id=2832249.2832353

[2] Shaowei Cai, Kaile Su, Chuan Luo, and Abdul Sattar. 2013. NuMVC: An Efficient Local Search Algorithm for Minimum Vertex Cover. *J. Artif. Int. Res.* 46, 1 (Jan. 2013), 687–716. http://dl.acm.org/citation.cfm?id=2512538.2512555

[3] Yuanzhu Peter Chen and Arthur L. Liestman. 2002. Approximating Minimum Size Weakly-connected Dominating Sets for Clustering Mobile Ad Hoc Networks. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking &Amp; Computing (MobiHoc '02)*. ACM, New York, NY, USA, 165–172. https://doi.org/10.1145/513800.513821

[4] Edsger W. Dijkstra. 1971. Hierarchical Ordering of Sequential Processes. *Acta Inf.* 1 (1971), 115–138. https://doi.org/10.1007/BF00289519

[5] Marco Dorigo and Thomas Stützle. 2004. *Ant colony optimization*. MIT Press.

[6] A. E. Eiben and James E. Smith. 2015. *Introduction to Evolutionary Computing*. Springer. https://doi.org/10.1007/978-3-662-44874-8

[7] Thomas A. Feo and Mauricio G. C. Resende. 1995. Greedy Randomized Adaptive Search Procedures. *J. Global Optimization* 6, 2 (1995), 109–133. https://doi.org/10.1007/BF01096763

[8] Michael R. Garey and David S. Johnson. 1990. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

[9] T.W. Haynes, S. Hedetniemi, and P. Slater. 1998. *Fundamentals of Domination in Graphs*. Taylor & Francis.

[10] David J. Johnson and Michael A. Trick (Eds.). 1996. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993*. American Mathematical Society, Boston, MA, USA.

[11] Daniel J. Lehmann and Michael O. Rabin. 1981. On the Advantages of Free Choice: A Symmetric and Fully Distributed Solution to the Dining Philosophers Problem. In *Eighth Annual ACM Symposium on Principles of Programming Languages (POPL)*. 133–138. https://doi.org/10.1145/567532.567547

[12] Michael Luby, Dana Randall, and Alistair Sinclair. 2001. Markov Chain Algorithms for Planar Lattice Structures. *SIAM J. Comput.* 31, 1 (2001), 167–192. https://doi.org/10.1137/S0097539799360355

[13] Ran Raz and Shmuel Safra. 1997. A Sub-Constant Error-Probability Low-Degree Test, and a Sub-Constant Error-Probability PCP Characterization of NP. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, Frank Thomson Leighton and Peter W. Shor (Eds.). ACM, 475–484. https://doi.org/10.1145/258533.258641

[14] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*. 4292–4293. http://networkrepository.com

[15] Jie Wu. 2002. Extended Dominating-Set-Based Routing in Ad Hoc Wireless Networks with Unidirectional Links. *IEEE Trans. Parallel Distrib. Syst.* 13, 9 (Sept. 2002), 866–881. https://doi.org/10.1109/TPDS.2002.1036062

[16] Ke Xu, Frédéric Boussemart, Fred Hemery, and Christophe Lecoutre. 2005. A Simple Model to Generate Hard Satisfiable Instances. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*. 337–342.