

Speeding up greedy maximum independent set approximation algorithms in sparse random graphs empirically via global-local equivalence

Sue Sue

Department of Computer Science and Software Engineering
California Polytechnic State University
San Luis Obispo, CA, USA
susue@calpoly.edu

Luke Fanguna

Department of Computer Science and Software Engineering
California Polytechnic State University
San Luis Obispo, CA, USA
lukefanguna@gmail.com

Daniel Frishberg

Department of Computer Science and Software Engineering
California Polytechnic State University
San Luis Obispo, CA, USA
dfrishbe@calpoly.edu

Abstract—The maximum (weighted or unweighted) independent set problem is a classical NP-hard combinatorial optimization problem with a variety of motivating applications and well-studied heuristics. Erdős-Rényi random graphs and Barabási-Albert random graphs are two models for generating graphs; these models are motivated by elegant mathematical properties and similarities to real-world networks. In Erdős-Rényi graphs, the approximation factor of a greedy, linear-time algorithm for finding a $1/2$ -optimal independent set is known. However, the analogue of this algorithm in weighted Erdős-Rényi graphs with n vertices and m edges runs in time $\Theta(m + n \log n)$.

In this paper, we give evidence that a heuristic known as the *nearest-neighbor chain* algorithm—which runs in linear time on sparse graphs—may find independent sets of weight asymptotically equal to those of the best polynomial-time algorithms for these random graphs.

Index Terms—greedy algorithms, random graphs

I. INTRODUCTION

A. Maximum Independent Set

An *independent set* is a set of vertices in a graph where no two vertices are adjacent. In the *maximum independent set* problem, given a graph $G = (V, E)$, the goal is to find an independent set in G that has maximum cardinality. The *maximum weighted independent set* problem is, given a graph $G = (V, E)$, where every vertex has a positive numerical *weight*, to find the independent set with maximum total weight. This is a classical combinatorial optimization problem.

The maximum independent set problem is *NP hard*, meaning that it belongs to a class of problems that are unlikely to admit a polynomial-time exact algorithm. This problem is closely

related to the *maximum clique* problem. Finding large independent sets is important in distributed algorithms [Gha22]. The maximum independent set problem remains NP hard even in graphs of bounded degree [BK99]. It is thought that given any $\epsilon > 0$, no polynomial-time algorithm can even find an approximation to the problem in the worst case that is within a factor $\frac{1}{|V|^{1-\epsilon}}$ of optimal. [Hås99; Zuc06]

Instead, efforts have focused either on heuristics for the general problem (e.g. [San+23]) or on special cases. One special case is that of random graphs. In a certain class of natural random graphs known as the *Erdős-Rényi* model, a natural greedy algorithm attains an expected $1/2$ -factor approximation to the problem, and it is believed that no polynomial-time algorithm (satisfying certain constraints) can do better. On the other hand, in general graphs, a simple greedy algorithm achieves a worst-case $\frac{3}{\Delta+2}$ -factor approximation [HR97], where Δ is the degree of the graph (the maximum degree of a vertex).

B. Greedy algorithms for maximum (weighted) independent set

We will be working with undirected graphs. In a graph $G = (V, E)$, given a vertex $v \in V$, let the *neighborhood* of v be the set

$$N(v) := \{u \in V \mid (u, v) \in E\}$$

Let $\delta(v) := |N(v)|$.

We describe a well-studied, straightforward greedy algorithm for the maximum independent set problem; we will call this algorithm *Naïve Greedy* (Algorithm 1). A straightforward implementation runs in time $\Theta(n + m)$ where n, m are respectively the number of vertices and edges in the graph. This algorithm attains a worst-case $\frac{1}{\Delta+1}$ -factor approximation. The algorithm can be modified for vertex-weighted graphs, preserving the guarantee, by selecting the vertex of highest

This work was supported in part through Summer Undergraduate Research Program (SURP) funding by the Department of Computer Science and Software Engineering at California Polytechnic State University, San Luis Obispo.

available weight in line 3—increasing the running time to $\Theta(m + n \log n)$. We call this *Naïve Weighted Greedy* (NWG).

Algorithm 1 NaïveGreedy

Input: $G = (V, E)$

Output: An independent set in G .

```

1:  $S \leftarrow \emptyset$ 
2: while  $V \neq \emptyset$  do
3:    $v \leftarrow$  any vertex in  $V$ 
4:    $S \leftarrow S \cup \{v\}$ 
5:   for  $u \in N(v)$  do
6:      $V \leftarrow V \setminus \{u\}$ 
7:   end for
8: end while
9: return  $S$ 

```

However, the simple example of a star graph reveals that the algorithm may make what appear to be poor choices by choosing high-degree vertices. The standard solution to this problem is to modify the algorithm as follows: choose the vertex that maximizes the score function

$$\arg \max_{v \in V} \{\text{score}(v)\} := \arg \max_{v \in V} \left\{ \frac{w(v)}{\delta(v) + 1} \right\}$$

where w, δ are defined with respect to the remaining graph $G[V]$ at the current step of the algorithm. (Here we assume G is a vertex-weighted graph and $w : V \rightarrow \mathbb{R}^+$ is the weight function.) The score function balances choosing vertices of large weight (since the goal is to choose a set of vertices with large weight) with choosing vertices of low degree (since choosing a high-degree vertex precludes choosing its many neighbors). One can modify line 3 to use this score function. However, calculating the score function naïvely increases the running time of the algorithm to $\Theta(n^2 + m)$. One can speed this up substantially using a priority queue: a binary heap gives a running time of $\Theta((n + m) \log n)$.

In an unweighted graph, one modifies the score function so that the numerator is 1 for all vertices. In unweighted graphs, the modified algorithm attains a $\frac{3}{\Delta+2}$ -factor approximation in the worst case. (Better guarantees are available at the cost of significantly worse constant factors in the running time.) In the weighted case (and therefore also in the unweighted case), [GGW10] gave a randomized $1 - \varepsilon$ -factor approximation algorithm for arbitrary $\varepsilon > 0$ for the problem in bounded-degree graphs, under certain assumptions about the distribution of the weights. However, the running time is $O\left(\frac{n}{\varepsilon^\Delta}\right)$; thus the dependence on the degree is exponential.

C. The Nearest-Neighbor Chain Algorithm

The *nearest-neighbor chain* (NNC) is a stack-based data structure first developed for clustering algorithms in computational biology. [Mam+19] applied this data structure to a number of problems in computational geometry, including the *multi-fragment heuristic* for the *Euclidean traveling salesperson problem*. In many cases, a greedy algorithm makes what can be considered *global* choices: when choosing the next

element to add to a solution, the algorithm calculates some score function or other criterion by checking every element in the problem instance. [Mam+19] showed that one may instead make *local* choices without sacrificing the quality of the solution found by the algorithm. They showed further that under certain conditions, the local algorithm finds exactly the same solution as the global algorithm—and runs more quickly due to the local nature of the choices.

One may also apply the NNC technique to combinatorial algorithms; however, the authors did not identify cases where a speedup was readily available. Furthermore, in the case of the maximum independent set problem (weighted and unweighted), the relevant locality property fails: the local algorithm may return a different solution from the “global” greedy algorithm.

In the unweighted version of the problem, the greedy score heuristic described in section I-B does not improve in expectation, in Erdős-Rényi graphs, on the (faster) naïve greedy algorithm. However, in the weighted case, obtaining a worst-case $\Theta(1/\Delta)$ guarantee with the naïve greedy algorithm requires first sorting the vertices in the graph by weight. This leads to a worst-case $\Theta(m + n \log n)$ running time. In contrast, the NNC algorithm runs in time $\Theta((n + m)\Delta)$ and can be improved to $\Theta((n + m) \log \Delta)$ if only the weights or static scores are considered. NNC is therefore a speed improvement, in bounded-degree graphs and in sparse Erdős-Rényi random graphs, over what appears to be the fastest “global” greedy algorithm for the weighted problem in Erdős-Rényi graphs.

D. Erdos-Renyi and Barabasi-Albert Random Graph Models

The maximum independent set problem is well motivated in part due to its close connection with the *maximum clique* problem. Finding cliques—collections of vertices in a graph that are all pairwise adjacent—arises naturally in the study of social networks and other real-world graphs. Therefore, in the present work, we consider graph models that approximate real-world graphs. We focus on two models of random graphs: *Erdős-Rényi* graphs and *Barabási-Albert* graphs [AB02]. The Erdős-Rényi model is a well-studied model with theoretical properties that simplify some aspects of analysis.

Many real-world networks, including social media, biological systems, and internet infrastructure, share a common “scale-free” structure where a few highly connected nodes exist alongside many nodes with only a handful of connections [AB02]. This is analogous to the way in which a few celebrities in a social network may have millions of followers while most users have only a few. The Barabási-Albert model mimics this real-world behavior through preferential attachment: new nodes are more likely to connect to already “popular” (well-connected) nodes. This contrasts with the simpler Erdős-Rényi model, where the probability that a connection exists between any pair of vertices is independent of all other connections.

Despite the general hardness of approximating the maximum independent set problem, in Erdős-Rényi graphs the naïve greedy algorithm finds a solution within a $1/2$ factor of

the size of the optimal solution with high probability [CE15; Wei21; Fri90; GM75]. It is believed that no polynomial-time algorithm (satisfying certain conditions) does better.

In weighted Erdős-Rényi graphs, the naïve greedy algorithm does not perform nontrivially unless it is modified to account for weights. This modification incurs an extra logarithmic factor due to a step in which one sorts the weights, giving a running time of $\Theta(m + n \log n)$. The nearest-neighbor chain algorithm, which we consider, improves on this time in sparse graphs—but no proof is known that the algorithm gives the same solution quality as the weighted “naïve” algorithm.

The question arises whether this result extends to Barabási-Albert graphs (a more natural model for social networks). A related question is the expected *independence number* of these graphs (the size of the largest independent set). [Kov20] made theoretical progress towards this question but left a logarithmic gap in the upper and lower bounds, in contrast to the known asymptotically tight bound for Erdős-Rényi graphs.

II. OUR RESEARCH QUESTIONS

In this work we focused on the following research questions:

Research Question 1: For the maximum weighted independent set problem, in random graphs, does the NNC implementation of the greedy algorithm with the standard score heuristic achieve the same, better, or worse approximation factor compared with the standard priority queue implementation?

Research Question 2: What is the asymptotic expected independence number of Barabási-Albert graphs?

III. PRELIMINARIES

The following section provides the essential definitions and models needed to understand our experiments. We assume familiarity with the definition of a graph, including vertices, edges, adjacency, and degree.

A. Graph models

The Erdős-Rényi model $G(n, p)$ generates random graphs by starting with $n \geq 1$ isolated vertices and independently adding each possible edge, independently and identically distributed with probability p (which may depend on n). This model generates graphs where most vertices have the same *expected* degree (although for any given graph, there may be vertices with very high degree and some with very low degree).

The Barabási-Albert model, in contrast, starts with a small initial graph, then repeatedly adds new vertices. Each new vertex connects to k existing vertices, where k is a fixed parameter. The probability of connecting a new vertex u to an existing vertex v is proportional to the current degree of v . This “rich get richer” mechanism produces graphs with a few high-degree vertices and many low-degree vertices [AB02].

B. Algorithm Variants

We study implementations of the greedy algorithm with the heuristic score function described in section I-B. The first is the priority queue (PQ) implementation. This implementation keeps all vertices in a priority queue (implemented using a

binary heap), extracting the maximum score vertex at each step and updating affected scores. The algorithm is as follows:

Algorithm 2 PQGreedy

Input: $G := (V, E)$

Output: An independent set in G

```

1:  $Q \leftarrow \text{PriorityQueue}()$ 
2:  $S \leftarrow \emptyset$ 
3: while  $V \neq \emptyset$  do
4:    $v \leftarrow Q.\text{pop}()$ 
5:    $S \leftarrow S \cup \{v\}$ 
6:   for  $u \in N(V)$  do
7:     for  $w \in N(u) \setminus \{v\}$  do
8:        $\text{score} \leftarrow$  the new score of  $w$ 
9:        $Q.\text{update\_priority}(w, \text{score})$ 
10:    end for
11:  end for
12:  remove  $v$  and its neighbors from  $Q$  and  $V$ 
13: end while
14: return  $S$ 
```

The use of a binary heap for the priority queue enables an overall $\Theta((n + m) \log n)$ runtime complexity. For the nearest-neighbor chain (NNC) variation on the algorithm, see Algorithm 3. The running time of the NNC algorithm is $\Theta((n + m)\Delta)$: each vertex has its score updated only when a neighbor is deleted from the graph. Also, each vertex v (re)calculates its score (which takes $O(\Delta)$ time) and the scores of its neighbors only when v is pushed to the stack or when a neighbor of v is popped from the stack. Taken over all vertices, these operations produce $O((n + m)\Delta)$ score recalculations; each score update takes $O(1)$ time. Although this running time is worse than that of the PQ-based algorithm in general, it is better (actually linear) in bounded-degree graphs.

IV. EXPERIMENTS

We implemented all four algorithms—Naïve Greedy (NG), Naïve Weighted Greedy (NWG), Priority Queue (PQ), and Nearest Neighbor Chain (NNC) in Erdős-Rényi and Barabási-Albert random graphs on n vertices, for $n = 1$ to 5,000, in increments of 500. For both graph models, we chose values of the probability parameter p to obtain graphs of expected degree $2/n$, $10/n$, $\log_{10} n/n$, $1/\sqrt{n}$, and $n/4$. For each value, we ran the experiment on five graph instances. Figure 1 shows results for the sparse case $10/n$. (Results were comparable for the case $2/n$.) Figure 2 shows results for the case $1/\sqrt{n}$. Figure 3 shows results for the dense case $n/4$.

A. Implementation

To generate the graphs, we used networkx [HSS07], a commonly used Python graph library. We used the networkx implementations of Erdős-Rényi and Barabási-Albert graphs. We created weighted and unweighted versions of the graphs; for the weighted versions, we generated uniformly random weights within a prescribed range. We used Python’s heapq priority queue, which supports heapify, heappush,

Algorithm 3 NNCGreedy**Input:** $G := (V, E)$ **Output:** An independent set in G

```

1: while  $V \neq \emptyset$  do
2:    $T \leftarrow \text{Stack}()$ 
3:    $v \leftarrow$  an arbitrary vertex
4:    $T.\text{push}(v)$ 
5:    $S \leftarrow \emptyset$ 
6:   while not  $T.\text{empty}()$  do
7:      $v \leftarrow T.\text{peek}()$ 
8:     if  $v \notin S$  then
9:        $T.\text{pop}()$ 
10:    else
11:      Calculate scores of  $v$  and its neighbors
12:       $w \leftarrow$  the vertex with the highest of these scores
13:      if  $w \neq v$  then
14:         $T.\text{push}(w)$ 
15:      else
16:         $T.\text{pop}()$ 
17:         $S \leftarrow S \cup \{v\}$ 
18:         $V \leftarrow V \setminus \{v\}$ 
19:        Delete all neighbors of  $v$  from  $S$ 
20:      end if
21:    end if
22:  end while
23: end while
24: return  $S$ 
25:

```

and heappop. To handle ties in the score function, we perturbed each vertex’s score. In the weighted case, we used the modified function $\text{score}(v) := \frac{w(v)}{\delta(v)+1} + \frac{w(v)}{10^7}$, breaking ties in favor of higher-weight vertices without meaningfully altering the scores. In the unweighted case, we added random noise to the score giving $\text{score}(v) := 1/(\delta(v) + 1 + \varepsilon)$, where $\varepsilon \sim \text{Uniform}(5 \times 10^{-8}, 10^{-6})$. For the NNC algorithm, we chose the initial vertex uniformly at random from V . When the stack became empty but edges remained, we chose a new starting vertex uniformly from the remaining vertices. For the naïve greedy algorithm (NG), we chose a vertex uniformly from the remaining graph at each step. For the naïve weighted greedy algorithm (NWG), we processed the vertices in decreasing weight order, breaking ties by index.

V. RESULTS

Concerning Research Question 1, NNC (with dynamic scores) obtained sets of comparable size (or weight) to PQ (with dynamic scores) and NWG (with weights only), outperforming NG. We observed this in unweighted and weighted graphs, in both models, for all edge densities. See Figures 1, 2, and 3. We did not obtain results for Research Question 2.

VI. DISCUSSION AND LIMITATIONS

We were unable to obtain a reliable comparison of running times, since we ran the experiments on a laptop. However,

in graphs of bounded degree, the theoretical time guarantee of NNC is asymptotically the same as NG and faster than PQ and NWG; thus for weighted graphs, NNC may combine the “best of both worlds,” giving a provably faster asymptotic running time (for suitable values of Δ) without sacrificing the 1/2-factor approximation of PQ and NWG.

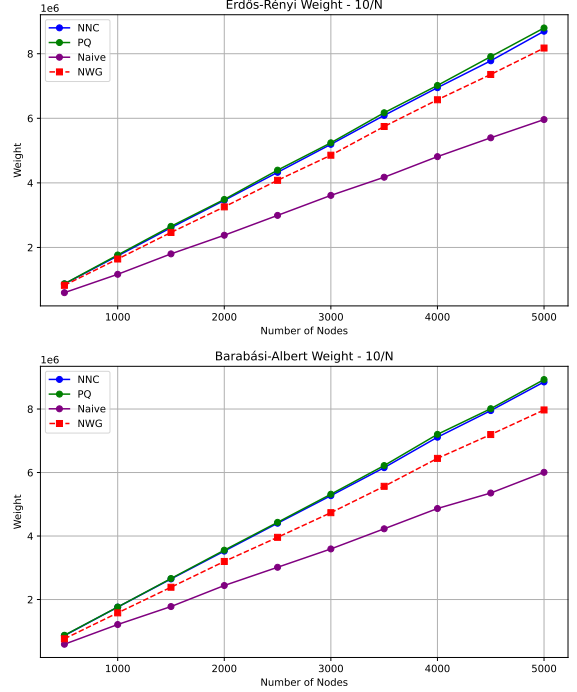


Fig. 1. In sparse Erdős-Rényi graphs, NNC, PQ, and NWG obtain higher-weight sets than NG.

Erdős-Rényi graphs with bounded (expected) average degree do not have bounded maximum degree. The worst-case improvement for NNC, therefore, is not immediate from the worst-case time in bounded-degree graphs. However, in these graphs a tighter analysis yields a high-probability linear-time bound. Nonetheless as noted, we did not experimentally verify running times. We also ran the experiments on a limited number of graphs—we chose values of n at increments of 500, and we generated only five graphs per setting of n and p . We did not perform a statistical analysis and cannot make claims about significance. Future work should run the experiments on larger instances, and on a larger number of instances.

Concerning Research Question 2, in Barabási-Albert graphs, no 1/2-factor expected guarantee is known, although looser bounds were given in [Kov20] on the independence number of these graphs. Empirical progress could be made in future work by evaluating whether the weights of the sets we find are closer to the lower or upper bound.

ACKNOWLEDGMENT

The authors acknowledge helpful conversations with Shayan Daijavad and Professor Theresa Migler.

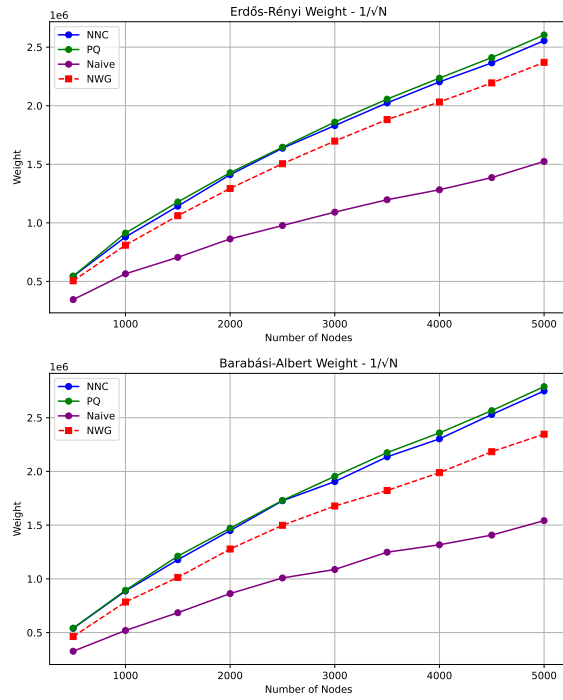


Fig. 2. In Erdős-Rényi graphs of expected average degree $\Theta(\log n)$ and $\Theta(\sqrt{n})$, NNC, PQ, and NWG obtain higher-weight sets than NG.

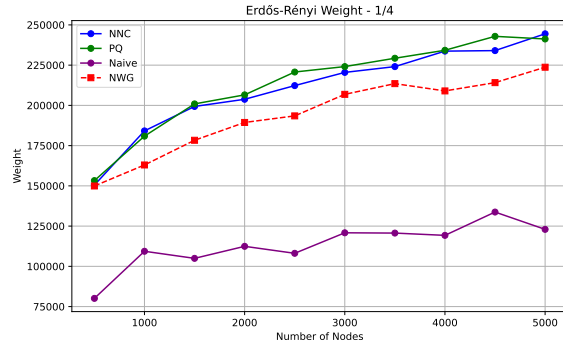


Fig. 3. In dense Erdős-Rényi graphs, NNC, PQ, and NWG obtain higher-weight sets than NG.

REFERENCES

- [AB02] Réka Albert and Albert-László Barabási. “Statistical mechanics of complex networks”. In: *Reviews of modern physics* (2002) (cit. on pp. 2, 3).
- [BK99] Piotr Berman and Marek Karpinski. “On Some Tighter Inapproximability Results (Extended Abstract)”. In: *Automata, Languages and Programming*. 1999 (cit. on p. 1).
- [CE15] Amin Coja-Oghlan and Charilaos Efthymiou. “On independent sets in random graphs”. In: *Random Structures & Algorithms* (2015) (cit. on p. 3).
- [Fri90] A.M. Frieze. “On the independence number of random graphs”. In: *Discrete Mathematics* (1990) (cit. on p. 3).
- [GGW10] David Gamarnik, David Goldberg, and Theophane Weber. “PTAS for maximum weight independent set problem with random weights in bounded degree graphs”. In: *Proceedings of the 2010 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2010 (cit. on p. 2).
- [Gha22] Mohsen Ghaffari. “Local Computation of Maximal Independent Set”. In: *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*. 2022 (cit. on p. 1).
- [GM75] G. R. Grimmett and C. J. H. McDiarmid. “On colouring random graphs”. In: *Mathematical Proceedings of the Cambridge Philosophical Society* (1975) (cit. on p. 3).
- [Hås99] Johan Håstad. “Clique is hard to approximate within $n^{1-\varepsilon}$ ”. In: *Acta Mathematica* (1999) (cit. on p. 1).
- [HR97] M. M. Halldórsson and J. Radhakrishnan. “Greed is good: Approximating independent sets in sparse and bounded-degree graphs”. In: *Algorithmica* (1997) (cit. on p. 1).
- [HSS07] Aric Hagberg, Pieter J Swart, and Daniel A Schult. *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Laboratory (LANL), 2007 (cit. on p. 3).
- [Kov20] Kirill Kovalenko. “On the independence number and the chromatic number of generalized preferential attachment models”. In: *Discrete Applied Mathematics* (2020) (cit. on pp. 3, 4).
- [Mam+19] Nil Mamano, Alon Efrat, David Eppstein, Daniel Frishberg, Michael T. Goodrich, Stephen Kobourov, Pedro Matias, and Valentin Polishchuk. “New Applications of Nearest-Neighbor Chains: Euclidean TSP and Motorcycle Graphs”. In: *30th International Symposium on Algorithms and Computation (ISAAC 2019)*. Dagstuhl, Germany, 2019 (cit. on p. 2).
- [San+23] Pablo San Segundo, Fabio Furini, David Álvarez, and Panos M. Pardalos. “CliSAT: A new exact algorithm for hard maximum clique problems”. In: *European Journal of Operational Research* (2023) (cit. on p. 1).
- [Wei21] Alexander S Wein. “Optimal low-degree hardness of maximum independent set”. In: *Mathematical Statistics and Learning* (2021) (cit. on p. 3).
- [Zuc06] David Zuckerman. “Linear degree extractors and the inapproximability of max clique and chromatic number”. In: *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing (STOC)*. 2006 (cit. on p. 1).