

# Random Graphs Generation and Processing

Seraphim Ivanov and Roman Cherepanov

**Abstract.** In this paper, structural properties of random graphs generated by various models are discussed. In the first section, random trees and random graphs produced by different models are analysed using statistical methods. In the second section, several graph processing algorithms are applied to generated random graphs and their performance is examined.

The models employed for tree generation are Prüfer code method and random recursive tree method. Two models of random graphs generation are studied: Erdős-Rényi model with a fixed number of edges  $q$  distributed among vertices [1] and Hilbert model with a fixed probability of each edge inclusion  $\pi$  [2]. In order to evaluate graph structural differences, distributions of vertex degrees are compared and tested for conformity to binomial distribution using Pearson's  $\chi^2$  test.

The algorithms study begins with an evaluation of efficiency of Breadth-First Search (BFS) and Depth-First Search (DFS). In order to compare efficiency of the algorithms, the number of traversed vertices is measured. The impact of graph density and geodesic distance between the source and the destination vertices on efficiency of the algorithms is discussed. Then Prim and Kruskal algorithms for minimal spanning tree search are studied in terms of processed edges number depending on the density and the edges weights support of the graph. Finally, two graph colouring algorithms are compared. A sequential [3, p. 359] and a greedy [3, p. 360] colouring algorithms are applied to random graphs across a range of graph densities, and the number of used colours is measured.

## Introduction

Random graphs serve as fundamental modelling tools across network science, computational biology, and infrastructure design, where they help simulate complex systems with uncertain connectivity patterns. Beyond their direct applications, these probabilistic structures play an equally important role as test beds for graph algorithms — particularly for methods whose performance depends critically on

structural properties that are difficult to characterize theoretically. This experimental approach becomes essential when studying algorithms where traditional complexity analysis provides limited practical insight, or where exact analytical solutions are unavailable. Classical random graph models are employed to generate test instances with well-understood statistical properties, enabling systematic evaluation of how algorithmic performance depends on fundamental graph characteristics. This approach provides empirical performance insights that complement theoretical analysis while maintaining mathematical rigour in experimental design. In the first section, random trees and random graphs produced by different models are analysed using statistical methods. In the second section, several graph processing algorithms are applied to generated random graphs and their performance is examined.

The random tree models studied are random recursive trees and random Prüfer code trees. In the first model, vertices are appended to the tree sequentially, choosing the parent of each appended vertex independently and uniformly at random from among the existing vertices. Notably, vertices with low indices tend to have higher degrees. In the second model, a random sequence of integer numbers is generated, and the sequence is interpreted as the Prüfer code representation of a tree. When the number of vertices and edges is sufficiently large, model deviations induced by the requirement of connectivity can be disregarded, and the generated structure closely approximates the theoretical random graph models under consideration.

Two graphs models with fixed number of vertices  $p$  investigated are Hilbert model  $G(p, \pi)$  and Erdős–Rényi random graph model  $G(p, q)$ . In the first model, each of  $q_m = p(p-1)/2$  the edges is included independently with equal probability  $\pi$ . In the second model, exactly  $q$  edges are selected uniformly at random from the set of all possible edges.

Under the law of large numbers, the random graph models  $G(p, \pi)$  and  $G(p, q)$  become asymptotically equivalent in distribution, meaning that all reasonable structural statistics converge as  $p \rightarrow \infty$ . In particular:

- The edge probability satisfies  $q_m \xrightarrow{\text{a.s.}} \pi \binom{p}{2}$  as the graph size grows;
- The vertex degree distribution in  $G(p, q)$  converges to  $\text{Bin}(p, \pi)$ , which characterizes the degree distribution in  $G(p, \pi)$ .

This theoretical convergence is empirically validated using Pearson's  $\chi^2$ -test (Table 1). Accordingly, once  $p > 500$ , the models can be considered statistically indistinguishable.

## 1. Graph traversal

The algorithms study begins with an evaluation of efficiency of Breadth-First Search (BFS) and Depth-First Search (DFS). In order to compare efficiency of the algorithms, the number of traversed vertices is measured on random graphs with  $p = 5000$  vertices and various densities. Additionally, the geodesic distance

Random graphs generation and processing

$p$	$d$	$\chi^2$	$\chi_{max}^2$	is accepted
200	0.3	75.74	72.15	False
200	0.5	130.94	79.08	False
200	0.7	75.84	72.15	False
300	0.3	54.86	88.25	True
300	0.5	98.34	93.95	False
300	0.7	78.93	88.25	True
500	0.3	92.34	114.27	True
500	0.5	108.96	116.51	True
500	0.7	91.51	110.90	True
1000	0.3	104.10	159.81	True
1000	0.5	149.33	165.32	True
1000	0.7	115.82	156.51	True

TABLE 1. Pearson's  $\chi^2$ -test,  $\alpha = 0.05$

$d$	$p$	avg BFS	avg DFS	avg $r$
0.0005	5000	2500.5	2499.8	11.26
0.001	5000	2498.9	2498.4	5.54
0.002	5000	2498.6	2503.1	3.95
0.004	5000	2500.9	2505.3	3.13
0.008	5000	2500.3	2499.7	2.71
0.016	5000	2502.3	2504.2	2.26
0.032	5000	2499.9	2504.5	1.97
0.064	5000	2496.8	2504.1	1.94
0.128	5000	2501.6	2499.1	1.87
0.256	5000	2505.4	2499.6	1.75
0.512	5000	2502.0	2500.6	1.49

TABLE 2. BFS vs DFS comparison

between the source and the destination vertices is calculated. The Monte Carlo simulation includes generation of  $g = 1000$  random graphs in Erdős-Rényi model and  $s = 500$  random searches on each graph. The number of edges  $q$  varies as  $h \cdot 0.001 \cdot q_m$ , where  $h \in \{2^{-1}, \dots, 2^9\}$ . The results of the simulation indicate that, on average, BFS and DFS traverse approximately  $p/2$  vertices (Table 2):

However, for a fixed density  $d$ , with distance  $r$  increase DFS achieves higher efficiency compared to BFS. Similarly, with fixed distance  $r$  and increasing density  $d$ , DFS also performs more efficiently. On pictures Figure 1a, Figure 1b log of traversed vertices, ratio BFS/DFS is presented. Green colour of bars indicates positive value of the ratio and higher efficiency of DFS traversal:

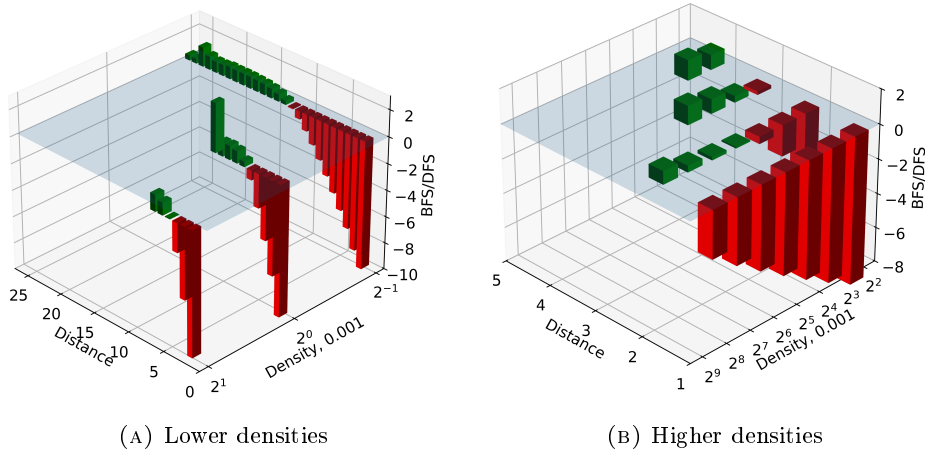


FIGURE 1. Efficiency of BFS and DFS

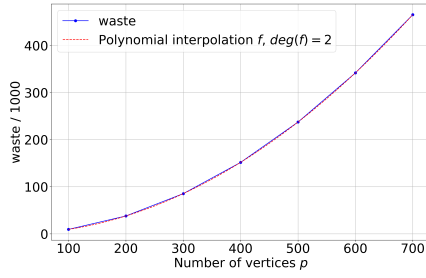
Considering the results of the experiment, with information on graph density and a heuristic for estimation of distance between the source and the target vertices, one of the algorithms is likely to perform more efficiently.

## 2. Spanning trees

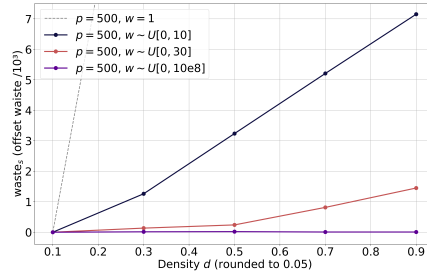
It is well known that Prim’s algorithm demonstrates greater efficiency than Kruskal’s algorithm on denser graphs. This difference stems from Kruskal’s requirement to sort edge weights, resulting in  $\mathcal{O}(q \log q)$  average complexity (using quicksort), compared to Prim’s  $\mathcal{O}(q) = \mathcal{O}(p^2)$  complexity (Figure 2a).

This established relationship motivates investigation of a different performance aspect. The analysis of minimum spanning tree algorithms is extended beyond standard graph statistics to include parameters of edge weight distributions. While Prim’s algorithm explicitly examines edges adjacent to the growing tree, Kruskal’s algorithm processes edges from a pre-sorted weight list. This structural difference suggests Kruskal’s performance depends on both the support size and variance of the edge weight distribution. Monte Carlo simulations on Gilbert model graphs  $G(p, \pi)$  confirm this hypothesis (Figure 2b). To evaluate the time complexity of the algorithms, a counter named *waste* was introduced to track the number of passes through critical sections of the algorithm. An increased *waste* value reflects a higher operational cost needed to solve the problem. Kruskal’s algorithm achieves peak efficiency when edge weights are unique — either drawn from continuous distributions (with zero collision probability) or discrete uniform distributions where the support size substantially exceeds the number of edges  $q$ . The algorithm’s efficiency decreases with either reduced support size or lower

## Random graphs generation and processing



(A) Complexity of Prim's as  $waste(p)$



(B) Complexity of Kruskal's as  $waste(d)$

FIGURE 2. Complexity of Prim's and Kruskal's algorithms

variance, eventually degrading to exhaustive search behaviour (making sorting irrelevant) for constant weights. Meanwhile, Prim's algorithm shows no comparable dependence on weight distribution characteristics in examined models.

The investigated property of the algorithm implies that, if a sorted list of the graph edges is present and used for another task, Kruskal's algorithm may perform more efficiently, provided that edge weights distribution has large enough support.

### 3. Graph colouring

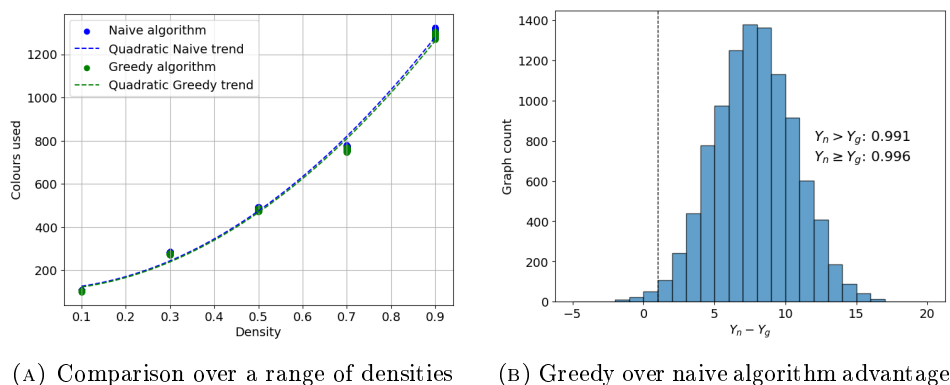
The colouring algorithms compared are naive sequential colouring and greedy colouring. For each algorithm, a Monte Carlo simulation was run across a range of densities  $\{0.1, 0.3, \dots, 0.9\}$  using Hilbert model. For each density, 20 000 random graphs with  $p = 5000$  vertices were generated, and the colouring algorithms were applied to them. The number of colours needed for both algorithms was measured. The distribution of the number of colours used indicates that the greedy algorithm tends to use fewer colours on average (Figure 3a). For  $\pi = 0.5$ , differences in the number of colours used were calculated. The experiment shows greedy colouring achieves better or equal results in 99% cases (Figure 3b).

### 4. Conclusion

In this paper, random graphs properties were studied. The experiments investigate conditions affecting graph processing algorithms' performance.

For BFS and DFS traversal algorithms, it is shown that graph density and geodesic distance between the source and the destination vertices have a significant impact on number of traversed vertices, but on average the algorithms have the same efficiency.

For Kruskal's spanning tree search algorithm, a dependency between the number of processed edges and edge weights distribution support is indicated. Prim's algorithm is shown to process all edges regardless of the weight distribution.



(A) Comparison over a range of densities (B) Greedy over naive algorithm advantage

FIGURE 3. Greedy and naive colouring algorithms comparison

Greedy colouring algorithm is demonstrated to be more efficient in terms of used colours number than naive algorithm.

## References

- [1] P. Erdős, A. Rényi, *On random graphs*, *Publicationes Mathematicae*, 1958, vol. 6, <https://doi.org/10.5486/PMD.1959.6.3-4.12>, №3-4, p. 290-297.
- [2] E. N. Gilbert, *Random Graphs*, *The Annals of Mathematical Statistics*, 1959, vol. 30, <https://doi.org/10.1214/aoms/1177706098>, № 4, p. 1141 - 1144.
- [3] F. A. Novilov, *Discrete mathematics for programmers*, 2009, ISBN: 978-5-91180-759-7, <https://stugum.wordpress.com/wp-content/uploads/2014/03/novikov.pdf>, 384 p.

Seraphim Ivanov  
 Faculty of Computer Science and Technology  
 Saint Petersburg Electrotechnical University  
 Saint Petersburg, Russia  
 e-mail: s.k.ivanov314@gmail.com

Roman Cherepanov  
 Faculty of Computer Science and Technology  
 Saint Petersburg Electrotechnical University  
 Saint Petersburg, Russia  
 e-mail: romacherepanov2002@gmail.com