Scalable Benchmark Graph Generation for the Maximum Cardinality Matching and Distance-1 Minimum Coloring Problems

Anthony Fabius[1,2], Ujwal Pandey[1,3], Dong Lin[1,4], Yash Kaul[1,5], and George M. Slota[1,6]

[1] Rensselaer Polytechnic Institute, Troy NY
[2] `fabiua@rpi.edu`, [3] `pandeu@rpi.edu` [4] `lind6@rpi.edu`, [5] `kauly@rpi.edu`, [6] `slotag@rpi.edu`

Abstract. The use of random graphs for algorithm benchmarking has found considerable use in recent years. The parametric generation of random benchmark graphs has been developed to analyze solution quality for community detection algorithms and for general scalability studies of parallel graph algorithms, among other applications. This paper presents general frameworks for the generation of benchmark graphs for two widely utilized problems within scientific computing applications: distance-1 coloring and maximum matching. We present methods to systematically generate useful instances of benchmarks for these problems, prove their ground truth solutions to be correct, and demonstrate their usage on real algorithm implementations in an experimental study.

1 Introduction

Random graphs is one of the more widely-studied topics within network science. Historically, random graph models have served as a theoretical basis to understand observed empirical properties of real-world networks (Erdös and Renyí [1959], Aiello et al. [2000], Bollobás [1980]). Within the past couple decades, random graph models have also been used to serve as null models to serve as a basis for explicitly measuring such emergent real-world properties (Dormann et al. [2009], Váša and Mišić [2022], Farine [2017], Perry and Wolfe [2012]). Generating random graphs has likewise emerged as a useful tool for studies of more complex phenomena, where theoretical analysis becomes unwieldy, such as with the analysis of subgraphs in biological networks (Milo et al. [2002]).

Concurrently, random graphs have found a usage in benchmarking studies, where the performance of some algorithm is analyzed across a range of random graphs with slight variations in the generation parameters. Possibly, the most well-known example is the Lancichinetti-Fortunato-Radicchi (LFR) benchmark (Lancichinetti et al. [2008]), which will generate a randomly configured graph with communities of certain sizes and relative densities (via the ratio of inter- vs. intra-community edges). Such a benchmark allows the systematic analysis of community detection algorithms across varying degree distribution skews, graph scales, and community definitions. Many other benchmark graph generators for the community detection problem have more recently emerged, generally targeting improvements upon LFR in speed, scale, or ability to control parameters (Hamann et al. [2018], Slota et al. [2019], Slota and Garbus [2020]).

This work proposes similar-style generators for the graph problems of maximum cardinality matching and distance-1 minimum graph coloring. We will use the shorthand *matching* and *coloring* to refer to these problems throughout this article. While matching and coloring have existed as long-standing graph theoretic problems, there has been a renewed interest in algorithms targeting them. This is primarily the result of matching and coloring having found widespread usage in a variety of scientific computing applications, from usage within parallelization frameworks (Allwright et al. [1995]), graph partitioning (Gilbert et al. [2021], Slota and Brissette [2024]), linear algebraic solvers (Gebremedhin et al. [2013], Duff and Koster [2001]), and a plethora of others (Belongie et al. [2002], Garey et al. [1976], Mandulak et al. [2025], among many more).

While maximum matchings can be computed in polynomial time, the scale of modern networks and scientific meshes often necessitates approximate solutions (Mandulak et al. [2024]). Likewise, minimum vertex coloring is famously NP-complete, hence no fast exact method likely exists. As such, when a new algorithm is proposed for these problems, it is typical to compare its results on certain inputs in a pairwise fashion to other existing approximation methods (Bogle et al. [2022]). A comparison against a known *ground truth solution* (borrowing terminology from the LFR benchmark and related work) is commonly absent. However, the knowledge of such a solution, combined with a systematic way to generate varying graphs having that solution, would enable studies that might produce considerable novel insight. E.g., do existing coloring algorithms show systematic poor performance when the chromatic number of some network is considerably smaller than the maximum degree when the degree distribution is particularly skewed? Or, how does the solve time behavior for maximum matching change when we increase graph diameter or increase problem scale overall?

Our Contributions: We present generalized scalable frameworks to enable to construction of benchmark graphs for the maximum cardinality matching and minimum graph coloring problems. Our approaches are customizable in several ways, including modifiable maximum matching cardinalities and chromatic numbers, allowing varying degree distributions, and a customizable edge generation process that can utilize any existing arbitrary random graph generator with minimal modification. We implement specific variants of our benchmark graph generators and perform systematic studies on existing well-known algorithms for matching and coloring to demonstrate their efficacy in practice.

2 Background

Graph Definitions: We define an undirected graph $G = (V, E)$, where V is the vertex set and E is the edge set, comprised of edges $e = (u, v) : e \in E$ with $u, v \in V$. Within the descriptions of our methods, we also use the adjacency matrix A

of graph G. A is a square and symmetric $|V| \times |V|$ nonnegative integer matrix, where a nonzero at $a_{i,j}$ indicates an edge from vertex i to vertex j. For multigraphs, the value of $a_{i,j}$ indicates the number of edges between i and j.

Maximum Cardinality Matching: A *matching* on G is a set of edges $M \subseteq E$ such that every vertex has at most a single edge in M incident on it. A vertex that has an incident matched edge is called *saturated*. A *maximum cardinality matching* is a matching on M where the largest possible number of vertices are saturated. The *maximum bipartite matching* problem is usually considered for bipartite graphs separately than *general graph matching* on any graph. Exact polynomial time algorithms for computing maximum matchings exist (e.g., Edmond's Blossom Algorithm, Edmonds [1965]), which typically use M-augmenting paths. Given a matching M on G, an *M-augmenting path* is a *path* starting on an unsaturated vertex in G, that traverses unmatched and then matched edges in alternating order, ending on another unsaturated vertex. By swapping matched-unmatched edges along this path, one can increase the size of the existing match by one. The famous result of *Berge* (Berge [1957]) states that a match M on G is maximum if and only if there exists no M-augmenting path.

Minimum Graph Coloring: The *graph coloring* problem assigns *colors* to every vertex $v \in V$ of some graph G. In practice, these are generally nonnegative integer values in $0 \dots C_{max}$. A *proper coloring* assigns colors such that $\forall e = (u, v) \in E : C(u) \neq C(v)$, where $C(v)$ gives the assigned color of v. The *chromatic number* of G, often given as $\chi(G)$, is the minimum number of colors necessary to properly color G. The computational problem of minimum graph coloring describes algorithms that attempt to produce a proper coloring of some G with the fewest possible number of colors. Generally, such algorithms are evaluated on how few colors result as well as time-to-solution. Most algorithms greedily color vertices *in some order* by assigning to each vertex the *lowest* color currently absent from its neighborhood.

Random Graphs: As part of our frameworks, we can consider any arbitrary random graph model, as long as it has a bounded vertex set. However, within our descriptions, we will focus on the related *Configuration Model* (Bollobás [1980]) and *Chung-Lu Model* (Aiello et al. [2000]), as well as *Random Geometric Graphs* (Penrose [2003]). Configuration Model (CM) and Chung-Lu (CL) generation results in a graph with some degree distribution, which the outputs of CM graphs match exactly but CL graphs do so only in expectation. A random geometric graph (RGG) places $|V|$ vertices in some metric space and creates edges between a pair of vertices if their Euclidean distance is below some threshold. Scalable implementations of these graphs have found widespread usage for benchmarking algorithms targeting graphs with skewed degree distributions (CM and CL) as well as clustered or mesh-like graphs with some geometric dependence (RGG).

3 Our Benchmark Model Generation Frameworks

Within this section, we will define and prove correctness of our benchmark models for matching and coloring. Both of our benchmark models incorporate the same basic generation concept. We first build a ground truth solution for each problem using a given vertex set via the construction of what we term as an *inner graph*, before overlaying much more complex topology via a random set of edges that comprise an *outer graph*. We note that the outer graph construction can be done independent of the inner graph using any random graph process. This might be useful to analyze an algorithm's performance across varying degree distribution and solution qualities, or analyze an algorithm's performance on a graph of a specified structure. We will provide explicit implementation details that utilize CM, CL, and RGG edge generation.

3.1 Maximum Cardinality Matching for General Graphs

Our benchmark model uses the well-known theorem from Berge. By constructing an inner graph with an given vertex set and maximum match M with a given cardinality, we can further construct a partitioning of vertices and an overlaying of edges such that we can guarantee no M-augmenting paths exists.

Vertex Set Partitioning: Given $|V|$ and some ratio r, defined as the proportion of saturated vertices in V, we partition V into three disjoint sets $V \rightarrow \{X, Y, Z\}$. We define and constrain cardinalities of these sets as:

$$|X| = |Y|,\ |X| + |Y| = r|V|,\ |Z| = (1-r)|V|$$

Within these sets, X and Y will be fully saturated and Z is entirely unsaturated by at least one possible maximum match. We will additionally define the following for use in the next subsection: $i = |X| = |Y|,\ k = |Z|$.

Inner Graph Construction: We construct the inner graph solely by creating all vertices and the edges for a ground truth maximum match. Note that this match may not necessarily be unique, though we will later show that it will always be maximum. Our ground truth match is created by adding a matched edge between each $x \in X$ to some $y \in Y$ in a bijection. As all $z \in Z$ are unmatched, no edges are added to these vertices in this stage.

Outer Graph Construction: We will define our outer graph in terms of an adjacency matrix A, given in a block structure in Equation 1 in terms of sub-adjacency matrices comprising the edges connecting solely within and solely between our sets X, Y, Z.

$$A = \begin{pmatrix} A_{i\times i} & A_{i\times i} & A_{i\times k} \\ A_{i\times i} & A_{i\times i} & A_{i\times k} \\ A_{k\times i} & A_{k\times i} & A_{k\times k} \end{pmatrix} = \begin{pmatrix} \mathbf{0} & \mathbf{1} + E_{XY} & \mathbf{0} \\ \mathbf{1} + E_{YX} & E_{YY} & E_{YZ} \\ \mathbf{0} & E_{ZY} & \mathbf{0} \end{pmatrix} \qquad (1)$$

We first order vertices as $X = \{x_1, x_2, \ldots, x_i\}$ and $Y = \{y_1, y_2, \ldots, y_i\}$, where $M = \{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i)\}$. Then, if we consider the block structure given in the middle portion of Equation 1, the adjacency matrix for M within the $A_{i\times i}$ blocks for edges between X and Y is simply the identity matrix. We do not need to consider any particular ordering for Z.

We construct our outer graph via the addition of an arbitrary set of edges in between X and Y represented as E_{XY} or equivalently E_{YX}, a set of edges in between vertices in Y as E_{YY}, and finally another set of edges in between Y and Z with E_{YZ} or E_{ZY}. These sets of edges can be generated using any arbitrary random graph process. After a short correctness proof, we will describe how one might use a process based on a degree distribution, as with CM and CL graphs, as well as a process based on RGGs.

Proposition 1. *The above definition of the adjacency matrix A in Equation 1 guarantees the construction of a graph with a maximum match exactly equal to $|M| = \frac{r|V|}{2} = |X| = |Y|$.*

Proof: Per Berge, a larger match than the one defined via a bijection from X to Y would require the existence of an M-augmenting path. By its definition, an M-augmenting path must start and finish at unsaturated vertices. We consider our ground truth match M and adjacency matrix A, as defined above. We have $|M| = |X| = |Y|$. We now attempt to find some M-augmenting path.

Any M-augmenting path must start and end at distinct vertices in Z, as all of X and Y are saturated by M. An M-augmenting path also alternatives between edges not in M and edges in M. Per the definition of A, we note that edges from all vertices in Z only have endpoints in Y, and these edges are all guaranteed to not be in M. We will attempt to construct an M-augmenting path. Consider traversing from some $z \in Z$ to $y \in Y$ along edge $e = (z, y)$. As y is saturated with edge $f = (y, x)$ with $x \in X$, our hypothetical M-augmenting path can be currently considered as $P_M = \{e, f\}$.

Case 1: x has no other edges. We are done, as there are no additional edges to add to P_M, and since x is saturated then P_M is not a proper M-augmenting path and M is a maximum match.

Case 2: There exists an edge $g = (x, y')$ to some vertex $y' \in Y : y \neq y'$. g can not be in M and y' must be saturated. Hence, there is guaranteed an additional edge $h = (y', x')$ with $h \in M$ and $x' \in X : x' \neq x$, and we now have $P_M = \{e, f, g, h\}$. At this point, we must consider these two possible cases again and we can observe the obvious inductive logic which results the conclusion:

Any path from an unsaturated vertex in Z and follows alternating edges not in M then in M will always end on a saturated vertex in X. Hence, no M-augmenting path exists and therefore M is maximum per Berge. □

Benchmark Graphs with a Given Degree Distribution: To construct some G that matches an input degree sequence $D = \{d_1, d_2, \ldots d_{|V|}\}$, we first must attempt to assign degrees to vertices in X, Y, Z. When considering the outer graph, we need to decrement all degrees of vertices in X and Y by exactly 1 to account for their inner graph matched edge. We then have the following defined constraint for assigning degrees to vertices. Note that this is in addition to the constraints given above for cardinalities of X, Y, and Z relative to r and $|V|$.

$$\sum_{x \in X} d(x) - 1 + \sum_{z \in Z} d(z) \leq \sum_{y \in Y} d(y) - 1 \qquad (2)$$

In general, a solution is observed to exist when D follows a realistic power-law degree distribution and the ratio r is not set to be too small. For our implementation, we employ a greedy approach, where we assign the largest degrees to Y then to X or Z in alternating order. In our experiments, we observe that for reasonable selections of r and many graph distributions, defined by a power-law exponent and vertex set cardinality, this simple greedy approach works. Generating the sets of edges for the outer graph given the assigned degrees is then trivially done via any arbitrary CM or CL edge generator. However, we note that an in-depth analysis of the limits of generation with these specific models is beyond the scope of this current work.

Benchmark Graphs with Higher Diameters and Clustering: We can instead use a random geometric process to construct our benchmark graph. First, the ground truth solution is generated. For each (x, y) matched edge, we randomly place x and y co-located within our metric space with pairwise distance lower than the edge generation threshold. All of z vertices are independently randomly placed. Edges are otherwise generated as normal using the distance threshold, as long as the edge also fits our defined outer graph construction.

Maximum Bipartite Matching: Our general model is also directly adaptable to be used for the maximum bipartite matching problem. We split each of X, Y, Z into $\{X_1, X_2\}$, $\{Y_1, Y_2\}$, and $\{Z_1, Z_2\}$. Matched edges are defined between $\{X_1, Y_1\}$ and $\{X_2, Y_2\}$, with random edges overlaid between $\{X_1, Z_1\}$, $\{X_1, Y_2\}$, and $\{Y_2, Z_2\}$. A correctness proof is omitted for space, but it follows the same logic as the one of the general case. An M-augmenting path must begin at either C_1 or C_2, but it is not possible to reach C_1 or C_2 while following alternating unmatched and matched edges.

3.2 Minimum Graph Coloring for General Graphs

Our approach for generating minimum graph coloring benchmark graphs, similar to our framework for constructing matching benchmark graphs, utilizes a known ground truth solution to overlay a more complex graph topology. Our method ensures that the chromatic number of the final, more complex graph remains unchanged from its initial, simpler form. The following description is relatively broad and straightforward, but we will soon present an explicit implementation that generates suitably *interesting* benchmark marks for our proposed use-cases.

Inner Graph Construction: The foundation of our benchmark graph is the inner graph, which is constructed to have a known chromatic number, $\chi(G) = k$. Any graph with a known chromatic number and proper coloring could serve as the inner graph. While a Mycielskian graph or a simple clique such are well-known graphs, we will describe a process for generating considerably more complex inner graph topology. The purpose of this inner graph is to provide a known *ground truth* solution, establishing a chromatic number for the final graph.

Outer Graph Construction: Our outer graph construction adds additional vertices and a random set of edges among the entire vertex set, without increasing the chromatic number of the graph. This approach allows us to generate a wide range of graphs with varying structures while guaranteeing a known ground truth for the coloring. To generate an outer graph, we simply can append new vertices to the vertex set and applying some k-coloring to them, while considering the known explicit k-coloring of the inner graph. We then add our random edges pairwise between vertices of differing colors using some random edge generative process. This process could allow for recoloring of vertices and rewiring of edges in the outer graph to achieve some degree distribution, color distribution, or other target constraint.

Proposition 2. *Merging a k-colored inner graph G_i with an outer graph G_o using the described procedure produces a k-chromatic final graph G_f.*

Proof: Our proof is relatively trivial. Per the definition of the outer graph construction, we assume that we have a k-chromatic inner graph. As G_i is a subgraph of G_f and it requires k colors to properly color, then G_f subsequently requires at least k colors. When combining G_o with G_i, we initially label the new vertices of G_o with at most k colors and only add edges $e = (u, v)$ such that $C(u) \neq C(v)$. Any recoloring or rewiring during the outer graph process adheres to these basic constraints. Hence, we maintain an explicit proper k-coloring for the final G_f and do not increase nor decrease the chromatic number relative to our inner graph. Hence, G_f is k-chromatic. □

A Nontrivial Inner Graph: For the implementation of our coloring benchmark, we utilize the Hajós construction operation to build a k-critical inner graph, which is a minimal graph with chromatic number k.We select the Hajós construction specifically because it can be adapted to create nontrivial topology up to an arbitrary size and k value, given some limitations. A Hajós construction takes two undirected graphs G and H and performs the following operation. Let $(x, y) \in E(G)$ and $(v, w) \in E(H)$, with vertices x and v having the same color and w and y having different colors. Join the graphs by performing a contraction on x and y, removing the edges $\{(x, y), (v, w)\}$, and adding a new edge (w, y), thus producing a new graph F. A fundamental property of the Hajós construction is that it preserves the maximum chromatic number between the inputs, meaning $\chi(F) = \max(\chi(G), \chi(H))$.

Our explicit construction of the k-critical inner graph begins with a K_k clique graph G. We then create F by selecting $H \cong G$ or $H \cong F'$, where F' is some output of a k-chromatic Hajós construction. We apply this process iteratively and at random, maintaining a known proper coloring, until we reach some maximal size, which could be determined by a target degree distribution or some other input parameter. Randomness is introduced through edge and vertex selections between G and H as well as the selection of H. Note that this process can create a k-chromatic graph with a maximum clique size arbitrarily smaller than k, an interesting property of the Mycielskian graphs that are currently used for benchmarking purposes; however, the Mycielskian construction is considerably more ordered and creates degree distributions and graph topologies very uncharacteristic or real networks.

Benchmark Graphs with a Given Degree Distribution: To construct some k-chromatic G that matches an input degree sequence $D = d_1, d_2, \ldots, d_{|V|} : d_1 \leq d_2 \leq \ldots d_{|V|}$, we first assign a subset of the degree sequence to the inner graph and the remainder to the outer graph. Similar to our matching benchmark, our approach assigns the larger degrees to the inner graph to enable its generation possible without vertices violating constraints of the degree sequence. Since we use K_k cliques as part of the Hajós construction process, our decision on where to partition D is based on the index of the lowest degree that is at least k. However, we can modify this index based on whether we wish to maximize or minimize the size of the inner graph or achieve some target ratio. Note that our inner graph construction process is adaptable to either.

For the remaining edges, we use CM or CL edge generations, given the outer graphs degrees and the *remainder* inner graph degrees after the Hajós construction process. We follow our general outer graph construction process, where we assign tentative colors before using these random models to generate edges. For achieving an exact degree sequence with a CM process, if a proposed edge would create a coloring conflict the edge is discarded, and the stubs are reshuffled to ensure the exact degree sequence is realized. For the CL model, edge probabilities are only calculated and examined pairwise between vertices of differing colors.

4 Experimental Demonstration

Experimental Setup: We implement our benchmark generation methods in Python. We specifically implement methods to match degree distributions for the general matching and coloring problems, as well as random geometric methods for matching. We perform a benchmarking study using an exact matching algorithm from LEMON (Dezső et al. [2011]). For coloring, we run variants from the ColPack library (Gebremedhin et al. [2013]), testing four of its greedy coloring algorithm ordering variants: Smallest Last, Incidence Degree, Largest First, and Dynamic Largest First. We examine time-to-solution for matching, coloring solution quality, as well as the general parametric limits of our implementations. We plan to publicly release our code upon publication of this article.

4.1 Maximum Cardinality Matching

We generate a large number of benchmark graphs for matching with $|V|$ varying from 500 to 10M vertices and vertex saturation ratio $r = \{0.2, 0.3, \ldots, 1.0\}$. We utilize power-law degree distributions with exponent $\gamma = \{2.0, 2.2, \ldots, 3.0\}$ with our CM implementation. For our RGG implementation, we place vertices in a 1.0×1.0 2D Euclidean space in a uniform random fashion and use a distance cutoff $d = \{0.05, 0.10, \ldots 0.50\}$. We evaluate the accuracy of our generator given parameter inputs as well as the performance of LEMON's `MaxMatching()` function in terms of time-to-solution.

Benchmark Generator Accuracy: We specifically consider a power-law exponent γ between 2 and 3, as it is representative of many real networks (Eikmeier and Gleich [2017]). Given our choices of power-law exponent, we note that the constraint given in Equation 2 is satisfied across all tested $|V|$ when $r \geq 0.4$. Analysis of our generated graphs also validates this, as our outputs match the input distribution exactly with an accurate ground truth maximum match cardinality. While a systematic study on the empirical r for real graphs has not been performed, theoretical studies of random graphs have suggested that the ratio value approaches 1.0 when the average degree of the graph is greater than approximately 4 (Zhou and Ou-Yang [2003]). Hence, our degree distribution-based generator is able to accurately model the properties of many real networks, while still allowing exploration of algorithm performance beyond that space. We observe larger diameters and higher clustering in our geometric generator, as expected, even though we limit edge generation per Equation 1.

Benchmark Study Observations: We are able to use our systematically-generated benchmark graphs to make a number of observations on the performance of LEMON's maximum cardinality matching algorithm. We report an experiment where we measured output graph diameter for our geometric benchmark and time-to-solution for LEMON by holding $r = 0.9$, $|V| = 100\text{k}$, and varying $d = 0.05 \ldots 0.50$. We observe a strong correlation between graph diameter and solve time, peaking at $d = 0.15$. This correlation was stronger than that between graph density (number of edges) and solve time, a rather surprising but not entirely unexpected result – exact matching algorithms tend to iteratively find M-augmenting paths, the length of which can have a dependence on graph diameter. Using our Configuration Model benchmark, we report another experiment where we hold $\gamma = 2.0$, $r = 0.8$, and vary $|V| = 500 \ldots 10\text{M}$ and observe LEMON's solve time. We note that as input scale increases, algorithm performance asymptotically decreases relative to the worst-case complexity.

4.2 Minimum Graph Coloring

We generate benchmark graphs for minimum graph coloring with $|V|$ varying from 1K to 100K vertices and chromatic number $k = \{3, 4, \ldots 8\}$. We use power-law degree distributions with exponent γ from 2.0 to 3.0 with our CM and CL implementations. We evaluate the performance ColPack's greedy coloring orderings with its distance-1 coloring algorithm by measuring the number of colors used relative to our generator's ground truth chromatic number.

Benchmark Generator Accuracy: Our generator's accuracy was validated by confirming that the generated graphs conform to their input parameters, producing instances with a guaranteed ground truth chromatic number, k. We achieve this through our implementation of CM and CL edge generation. Our CM methods matches the input degree sequence exactly, while our CL method matches in expectation, minus minor deviations from the target degree sequence due to coloring constraints. This deviation decreases as k increases. Despite this, the adherence to our coloring constraints ensures the graph remains properly colored, thereby demonstrating the accuracy of the ground truth solution.

Benchmark Study Observations: We used our generator to make a number of observations on the performance of ColPack's coloring algorithms. In one experiment, we alter the skew of the graph degree distributions by varying γ while holding $|V| = 100\text{K}, k = 5$. We observe that for highly skewed degree distribution graphs ($\gamma = 2.0$), the Incidence Degree ordering consistently performs better than other orderings. As the graph's degree distribution become more uniform ($\gamma = 3.0$), the performance is more consistent across all orderings, with Dynamic Largest First producing slightly better results. Interestingly, when $\gamma \approx 2.5$, we note that all orderings perform notably worse than for larger or smaller γ values. This is possibly the result of a interesting interplay between average degree, degree distribution tail weight, and their impact on ordering quality.

Following this, our next experiment varied the vertex set size, $|V|$ as we hold $k = 5, \gamma = \{2.0, 3.0\}$. We observed that as the graphs scale in size, the performance of the orderings remain consistent with what was observed above, regardless of whether the degree distribution is closer to $\gamma = 2.0$ or $\gamma = 3.0$. However, as $|V|$ increases, results deviated further from the ground truth on a relative basis. We report a final observation as we vary k while holding $\gamma = \{2.0, 3.0\}, |V| = 100\text{K}$. We note that the colorings perform consistently again as above but remain far from optimal. For $\gamma = 2.0$, we observe colorings between $[10, 25]$, while surprisingly the less skewed $\gamma = 3.0$ graphs resulted in colorings between $[25, 28]$. We also observed that solutions for lower k values were relatively closer to optimal than those for higher k values. Overall, these findings suggests that our generator provides a unique tool for a deeper study of where the pitfalls of different coloring orderings may be.

5 Conclusions and Future Work

This manuscript has presented two generalized frameworks for the systematic generation of random benchmark graphs for the graph matching and graph coloring computational problems. We described implementations and demonstrated their usefulness by benchmarking widely used matching and coloring codes, while making observations with the reported results. Our methods will be useful for algorithm designers, to identify potential drawbacks or benefits of novel algorithmic approaches to the coloring and matching problems.

Current Limitations and Future Work: As this is the first such effort towards generalized benchmarking graphs for coloring and matching, our approaches are limited to the baseline and most well-known problem variants. Other related formulations of these problems, such as distance-2 coloring, partial distance-2 coloring, star coloring, bipartite matching, weighted matching, minimum cost matching, and a plethora of others was prohibitive to include in the scope of our current effort. Exploring all of these problems for possible benchmark graph generation is an excellent avenue for future work. Weighted graph matching in particular, for its use in graph partitioning and general graph coarsening, as well as partial distance-2 coloring, for its use in algorithmic differentiation, are particularly promising future targets for our efforts.

6 Acknowledgments

This work is supported by the National Science Foundation under Grant No. 2047821.

Bibliography

William Aiello, Fan Chung, and Linyuan Lu. A random graph model for massive graphs. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 171–180, 2000.

JR Allwright, R Bordawekar, PD Coddington, K Dincer, and CL Martin. A comparison of parallel graph coloring algorithms. *SCCS-666*, pages 1–19, 1995.

Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE transactions on pattern analysis and machine intelligence*, 24(4):509–522, 2002.

Claude Berge. Two theorems in graph theory. *Proceedings of the National Academy of Sciences*, 43(9):842–844, 1957.

Ian Bogle, George M Slota, Erik G Boman, Karen D Devine, and Sivasankaran Rajamanickam. Parallel graph coloring algorithms for distributed GPU environments. *Parallel Computing*, 110:102896, 2022.

Béla Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European Journal of Combinatorics*, 1(4):311–316, 1980.

Balázs Dezső, Alpár Jüttner, and Péter Kovács. Lemon - an open source c++ graph template library. In *WGT@ETAPS*, 2011. URL https://api.semanticscholar.org/CorpusID:44883675.

Carsten F Dormann, Jochen Fründ, Nico Blüthgen, and Bernd Gruber. Indices, graphs and null models: analyzing bipartite ecological networks. 2009.

Iain S Duff and Jacko Koster. On algorithms for permuting large entries to the diagonal of a sparse matrix. *SIAM Journal on Matrix Analysis and Applications*, 22(4):973–996, 2001.

Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17: 449–467, 1965.

Nicole Eikmeier and David F Gleich. Revisiting power-law distributions in spectra of real world networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 817–826, 2017.

P Erdös and A Renyí. On random graphs i. *Publ. math. debrecen*, 6(290-297): 18, 1959.

Damien R Farine. A guide to null models for animal social network analysis. *Methods in Ecology and Evolution*, 8(10):1309–1320, 2017.

Michael Garey, David Johnson, and Hing So. An application of graph coloring to printed circuit testing. *IEEE Transactions on circuits and systems*, 23(10): 591–599, 1976.

Assefaw H Gebremedhin, Duc Nguyen, Md Mostofa Ali Patwary, and Alex Pothen. Colpack: Software for graph coloring and related problems in scientific computing. *ACM Transactions on Mathematical Software (TOMS)*, 40 (1):1–31, 2013.

Michael S Gilbert, Seher Acer, Erik G Boman, Kamesh Madduri, and Sivasankaran Rajamanickam. Performance-portable graph coarsening for efficient multilevel graph analysis. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 213–222. IEEE, 2021.

Michael Hamann, Ulrich Meyer, Manuel Penschuck, Hung Tran, and Dorothea Wagner. I/o-efficient generation of massive graphs following the lfr benchmark, 2018.

Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 78(4):046110, 2008.

Michael Mandulak, Sayan Ghosh, SM Ferdous, Mahantesh Halappanvar, and George Slota. Efficient weighted graph matching on GPUs. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2024.

Michael Mandulak, S M Ferdous, Sayan Ghosh, Mahantesh Halappanavar, and George Slota. ApproxJoin: Approximate matching for efficient verification in fuzzy set similarity join, 2025. URL https://arxiv.org/abs/2507.18891.

Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

Mathew Penrose. *Random geometric graphs*, volume 5. OUP Oxford, 2003.

Patrick O Perry and Patrick J Wolfe. Null models for network data. *arXiv preprint arXiv:1201.5871*, 2012.

George M Slota and Christopher Brissette. Constant-memory graph coarsening. In *2024 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–7. IEEE, 2024.

George M Slota and Jack Garbus. A parallel LFR-like benchmark for evaluating community detection algorithms. In *IPDPS Workshops*, pages 1112–1115, 2020.

George M Slota, Jonathan W Berry, Simon D Hammond, Stephen L Olivier, Cynthia A Phillips, and Sivasankaran Rajamanickam. Scalable generation of graphs for benchmarking HPC community-detection algorithms. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–14, 2019.

František Váša and Bratislav Mišić. Null models in network neuroscience. *Nature Reviews Neuroscience*, 23(8):493–504, 2022.

Haijun Zhou and Zhong-can Ou-Yang. Maximum matching on random graphs. *arXiv preprint cond-mat/0309348*, 2003.