Approximation algorithms for maximum matchings in undirected graphs

Fanny Dufossé, Kamer Kaya, Ioannis Panagiotas, Bora Uçar

Approximation algorithms for maximum matchings in undirected graphs

Fanny Dufossé* Kamer Kaya† Ioannis Panagiotas‡ Bora Uçar§

Abstract

We propose heuristics for approximating the maximum cardinality matching on undirected graphs. Our heuristics are based on the theoretical body of a certain type of random graphs, and are made practical for real-life ones. The idea is based on judiciously selecting a subgraph of a given graph and obtaining a maximum cardinality matching on this subgraph. We show that the heuristics have an approximation guarantee of around $0.866 - \log(n)/n$ for a graph with n vertices. Experiments for verifying the theoretical results in practice are provided.

1 Introduction

A matching in a graph is a set of edges no two of which share a common vertex. We consider the maximum cardinality matching problem for general graphs; this problem asks for a matching of maximum size. There are a number of polynomial time algorithms to solve this problem exactly. The lowest worst-case time complexity of the known algorithms is $\mathcal{O}(\sqrt{n}\tau)$ for a graph with n vertices and τ edges [1, 9, 23]. There is considerable interest in simpler and faster algorithms that have some approximation guarantee, as such simpler algorithms are used as a jump-start routine by the current state-of-the-art matching algorithms [5, 19].

We propose randomized heuristics and analyze their performance. The proposed heuristics extend previous work [6] and construct a subgraph of the input graph by randomly choosing some edges. They then obtain a maximum cardinality matching in the selected subgraph and return it as an approximate matching for the input graph. The probability density function for choosing a given edge is obtained with a sparse matrix scaling method. One of the heuristics use the well-known Karp-Sipser (KS) heuristic [14] to find a maximum cardinality matching on the selected subgraph. Its expected matching cardinality provides an approximation around 0.866 under the condition that the scaling method has successfully scaled the input matrix. The analysis can also be thought of analyzing Karp-Sipser heuristic's performance on a certain model of random graphs. Two other variants obtain better results than the first one; in theory and in practice.

The organization of the paper is as follows. In Section 2, we give some background on matchings, doubly stochastic scaling methods, and a brief summary of related work. In Section 3, we propose our matching heuristics and analyze their approximation guarantee. We then give experiments in Section 4, where we observe the theoretical findings in practice. Section 5 concludes the paper and briefly discusses the future work.

*Inria, Grenoble Rhône-Alpes, fanny.dufosse@inria.fr

†Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul, Turkey, kaya@sabanciuniv.edu

‡ENS Lyon, France, ioannis.panagiotas@ens-lyon.fr

§CNRS and LIP, UMR5668 (CNRS - ENS Lyon - UCBL - Université de Lyon - INRIA), Lyon, France, bora.ucar@ens-lyon.fr

2 Notation and background

Let $G = (V, E)$ be an undirected graph, where V is the vertex set and E is the edge set. G can be represented as a symmetric, square, sparse matrix $\mathbf{A}$. Each row (and its corresponding column) of $\mathbf{A}$ corresponds to a unique vertex in V so that $a_{ij} = a_{ji} = 1$ if and only if $(v_i, v_j) \in E$. We use $\text{adj}(u)$ to denote the set of neighbors of a vertex, and extend this to a set S of vertices as $\text{adj}(S)$ in a natural way. The number of edges incident on a vertex is called its degree. A *path* in a graph is a sequence of vertices such that each consecutive vertex pair share an edge. A vertex *is reachable from* another, if there is a path between them. The *connected components* of a graph are the equivalence classes of vertices under the "is reachable from" relation. A *cycle* in a graph is a path whose start and end vertices are the same. A *simple cycle* is a cycle with no vertex repetitions. A *tree* is a connected graph with no cycles. A *spanning tree* of a connected graph G is a tree containing all its vertices.

A graph $G' = (V', E')$ is a *subgraph* of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. Let $G_k = (V, E')$ be a random subgraph of G where each vertex in V randomly chooses k of its edges, with repetition. An edge $\{u, v\}$ is included in G^k_{out} only once even if it is chosen multiple times. We call G^k_{out} as a k-out subgraph of G. A permutation of $1, \ldots, n$ in which no element stays in its original position is called *derangement*. The total number of derangements of $1, \ldots, n$ is denoted by $!n$, and is the nearest integer to $\frac{n!}{e}$ [3, pp. 40–42], where e is the base of the natural logarithm.

2.1 Maximum cardinality matching. A matching $\mathcal{M}$ in a graph $G = (V, E)$ is a subset of edges E where a vertex in V is in at most one edge in $\mathcal{M}$. Given a matching $\mathcal{M}$, a vertex v is said to be *matched* by $\mathcal{M}$ if v is in an edge of $\mathcal{M}$, otherwise v is called *unmatched*. If all the vertices are matched by $\mathcal{M}$, then $\mathcal{M}$ is said to be a *perfect matching*. The *cardinality* of a matching $\mathcal{M}$, denoted by $|\mathcal{M}|$, is the number of edges in $\mathcal{M}$. The maximum cardinality matching problem asks for a matching of maximum size. We call such a matching a *maximum matching*.

2.2 Scaling matrices to doubly stochastic form. An $n \times n$ matrix $\mathbf{A} \neq 0$ is said to have *support* if there is a perfect matching in the associated bipartite graph. An $n \times n$ matrix $\mathbf{A}$ is said to have *total support* if each edge in its bipartite graph can be put into a perfect matching. A square sparse matrix is called irreducible if its directed graph is strongly connected. A square sparse matrix $\mathbf{A}$ is called fully indecomposable if for a permutation matrix $\mathbf{Q}$, the matrix $\mathbf{B} = \mathbf{AQ}$ has a zero free diagonal and the directed graph associated with $\mathbf{B}$ is irreducible. Fully indecomposable matrices have total support; but a matrix having total support could be a block diagonal matrix, where each block is fully indecomposable. For more formal definitions of support, total support, and the fully indecomposability, see the book by Brualdi and Ryser [2, Ch. 3 and Ch. 4].

Any nonnegative matrix $\mathbf{A}$ with total support can be scaled with two (unique) positive diagonal matrices $\mathbf{D}_R$ and $\mathbf{D}_C$ such that $\mathbf{D}_R\mathbf{A}\mathbf{D}_C$ is doubly stochastic (that is, the sum of entries in any row and in any column of $\mathbf{D}_R\mathbf{A}\mathbf{D}_C$ is equal to one). If $\mathbf{A}$ has support but not total support, then $\mathbf{A}$ can be scaled to a doubly stochastic matrix but not with two positive diagonal matrices [30]—this fact is also seen in more recent treatments [16, 18, 29]).

The Sinkhorn-Knopp algorithm [30] is a well-known method for scaling matrices to doubly stochastic form. This algorithm generates a sequence of matrices (whose limit is doubly stochastic) by normalizing the columns and the rows of the sequence of matrices alternately. While the limit is symmetric, the intermediate matrices generated in the sequence are not symmetric. Two other iterative scaling algorithms [17, 18, 29]) maintain symmetry all along the way. That is why we use this latter class of algorithms.

2.3 Related work. The first polynomial time algorithm with $\mathcal{O}(n^2\tau)$ complexity for the maximum cardinality matching problem on general graphs with n vertices and τ edges is proposed by Edmonds [7]. Currently, the fastest algorithms have $\mathcal{O}(\sqrt{n}\tau)$ complexity [1, 9, 23]. The first of these algorithms is by Micali and Vazirani [23] whose clear exposition is later presented by Peterson and Loui [27] and recently, its simpler analysis is given by Vazirani [31]. Later, Blum [1] and Gabow and Tarjan [9] proposed algorithms with the same complexity.

Random graphs have been frequently analyzed in terms of their maximum matching cardinality. For bipartite graphs, Walkup [32] showed that a 1-out subgraph of a complete bipartite graph $G = (V_1, V_2, E)$ where $|V_1| = |V_2|$ asymptotically almost surely (a.a.s.) does not contain a perfect matching. He also showed that a 2-out subgraph of a complete bipartite graph a.a.s. contains a perfect matching. For complete undirected graphs, similar results are shown by Frieze [8]. Karoński and Pittel [12] improved the results of Walkup by allowing the vertices which are not selected by other vertices to select one more neighbor. They showed that this random graph a.a.s. has a perfect matching. In these graphs, the expected number of edges per vertex is $1 + e^{-1}$.

Randomized algorithms which check the existence of a perfect matching and generate perfect/maximum matchings have been proposed in the literature [4, 11, 20, 25, 26]. Lovász showed that the existence of a perfect matching can be verified in randomized $\mathcal{O}(n^\omega)$ time where $\mathcal{O}(n^\omega)$ is the time complexity of the fastest matrix multiplication algorithm available [20]. More information such as the set of allowed edges, i.e., the edges in some maximum matching, of a graph can also be found with the same randomized complexity as shown by Cheriyan [4].

To construct a maximum cardinality matching in a general, non-bipartite graph, a simple, easy to implement algorithm with $\mathcal{O}(n^{\omega+1})$ randomized complexity is proposed by Rabin and Vazirani [28]. The algorithm uses matrix inversion as a subroutine. Later, Mucha and Sankowski proposed the first algorithm for the same problem with $\mathcal{O}(n^\omega)$ randomized complexity [26]. This algorithm uses expensive sub-routines such as Gaussian elimination and equivalence classes formed based on the edges that appear in some perfect matching [21]. A simpler randomized algorithm with the same complexity is proposed by Harvey [10].

Although the above-mentioned randomized algorithms find maximum matchings, they use expensive subroutines. Our focus is on matching heuristics that have linear or near linear run time and use randomization to obtain good quality guarantees. Recent surveys of matching heuristics are given by Kaya et al. [5, Section 4] and Langguth et al. [19]. The Karp-Sipser (KS) heuristic [14] is suggested as a good initialization step for exact bipartite maximum cardinality matching algorithms [15]. The KS heuristic takes any degree-1 vertex and matches it to its unique neighbor. When there is no degree-1 vertex, it chooses an edge randomly and matches the two end-points. Upon matching a pair of vertices, KS removes them from the graph and updates the degrees of the remaining vertices. Karp-Sipser heuristic becomes an exact algorithm for graphs with components having unique cycles.

Three of us proposed [6] heuristics for bipartite graphs. One of the heuristics scales the adjacency matrix of a given bipartite graph to extract a 1-out subgraph, and applies Karp-Sipser heuristic as an exact algorithm. This heuristic is shown to deliver matchings of cardinality $0.866n$ for a bipartite graph with n vertices on each side. This number is also the expected cardinality of a maximum matching in a random 1-out subgraph of a complete bipartite graph [22]. Excluding the cost of scaling, the heuristic has $\mathcal{O}(n + \tau)$ time complexity. We empirically showed that only five scaling iterations (with linear time complexity) is sufficient to obtain such matchings in practice. This work extends the previous work from bipartite graphs to general graphs. Our algorithmic constructions are the same (the use of scaling and Karp-Sipser). The theoretical analysis involves

new results and adaptations of some of the previous ones.

3 Heuristics and their analysis

We first present the main heuristic, and then analyze its approximation guarantee. While the heuristic is a straightforward adaptation of its counterpart for bipartite graphs [6], the analysis is more complicated, because of odd cycles. The analysis shows that random 1-out subgraphs of a given graph have the maximum cardinality of a matching around $0.866 - \log(n)/n$ of the best possible—the observed performance (see Section 4) is higher. Then two variants of the main heuristic are discussed in Section 3.3 without proofs. They include the 1-out subgraphs from the first one, and hence deliver better results theoretically.

3.1 One-Out**: The main heuristic.** The heuristic shown in Algorithm 1 first scales the adjacency matrix of a given undirected graph to be doubly stochastic. Based on the values in the matrix, the heuristic then randomly marks one of the neighbors for each vertex as chosen. This defines one marked edge per vertex. Then, the subgraph of the original graph containing only the marked edges is formed, yielding an undirected graph with at most n edges, each vertex having at least one edge incident on it. The heuristic KS is run on this subgraph and finds a maximum cardinality matching on it, as the resulting 1-out graph has unicyclic components. The approximation guarantee of the proposed heuristic is analyzed for this step. One improvement to the previous work is to make sure that the matching obtained is a maximal one by running KS heuristic on the vertices that are not matched. This brings in a large improvement to the cardinality of the matching in practice.

Algorithm 1 One-Out

Input $G = (V, E)$ and its adjacency matrix $\mathbf{A}$
Output `match`$[\cdot]$: the matching
1: $\mathbf{D} \leftarrow$ SymScale($\mathbf{A}$)
2: **for** $i = 1$ **to** n **do**
3: Pick a random $j \in \mathbf{A}_{i*}$ by using the probability density function

$$\frac{s_{ik}}{\Sigma_{t \in \mathbf{A}_{i*}} s_{it}}, \text{for all } k \in \mathbf{A}_{i*}$$

where $s_{ik} = \mathbf{d}[i] \times \mathbf{d}[k]$ is the corresponding entry in the scaled matrix $\mathbf{S} = \mathbf{DAD}$.
4: Mark that i chooses j
5: Construct a graph $G^1_{out} = (V, E)$, where

$$V = \{1, \ldots, n\}$$
$$E = \{(i, j) : i \text{ chose } j \text{ or } j \text{ chose } i\}$$

6: `match` $\leftarrow$KarpSipser$_{\text{One-Out}}(G^1_{out})$
7: Make `match` a maximal matching

Let us classify the edges incident on a vertex v_i as in-edges (from those neighbors that have chosen i at Line 4) and an out-edge (to a neighbor chosen by i at Line 4. Dufossé et al. [6, Lemma 3] show that during any execution of Karp-Sipser, it suffices to consider the vertices whose in-edges unavailable but out-edges are available as degree-1 vertices.

The analysis traces an execution of KS on the subgraph G^1_{out}. Let A_1 be the set of vertices not

chosen by any other vertex at Line 4 of Algorithm 1. These vertices have in-degree zero and out degree one, and hence can be processed by KS. Let B_1 be the set of vertices chosen by the vertices in A_1. The vertices in B_1 can be perfectly matched with vertices in A_1; leaving some A_1 vertices not matched and creating some new in-degree-0 vertices. We can proceed to define A_2 to be the vertices that have in degree-0 in $V \setminus (A_1 \cup B_1)$, and define B_2 as those chosen by A_2, and so on so forth. Formally, let B_0 be an empty set, and define A_i to be the set of vertices with in-degree 0 in $V \setminus B_{i-1}$, and B_i be the vertices chosen by those in A_i, for $i \geq 1$. Notice that $A_i \subseteq A_{i+1}$ and $B_i \subseteq B_{i+1}$. The Karp-Sipser heuristic can process A_1, then $A_2 \setminus A_1$, and so on, until the remaining graph has cycles only. The sets A_i and B_i and their cardinality are at the core of our analysis. We first present some facts about these sets and their cardinality, and describe an implementation of KS instrumented to highlight them.

LEMMA 3.1. *With the definitions above,* $A_i \cap B_i = \emptyset$.

Proof. We prove this by induction. For $i = 1$ it clearly holds. Assume that it holds for all $i < \ell$. Suppose there exists a vertex $u \in A_\ell \cap B_\ell$. Because $A_{\ell-1} \cap B_{\ell-1} = \emptyset$, u must necessarily belong to both $A_\ell \setminus A_{\ell-1}$ and $B_\ell \setminus B_{\ell-1}$. For u to be in $B_\ell \setminus B_{\ell-1}$, there must exist at least one vertex $v \in A_\ell \setminus A_{\ell-1}$ such that v chooses u. However the condition for $u \in A_\ell$ is that no vertex in $V \cap (A_{\ell-1} \cup B_{\ell-1})$ has selected it. This is a contradiction and the intersection $A_\ell \cap B_\ell$ should be empty. □

COROLLARY 3.1. $A_i \cap B_j = \emptyset$ *for* $i \leq j$.

Proof. Assume $A_i \cap B_j \neq \emptyset$. Since $A_i \subseteq A_j$ we have a contradiction as $A_j \cap B_j = \emptyset$ by Lemma 3.1.□

Thanks to Lemma 3.1 and Corollary 3.1, the sets A_i and B_i are disjoint, and they form a bipartite subgraph of G^1_{out}, for all $i = 1, \ldots, \ell$.

The version of Karp-Sipser heuristic for 1-out graphs is shown in Algorithm 2. The degree-1 vertices are kept in a first-in first-out priority queue Q. The queue is first initialized with A_1, and a # is used to mark the end of A_1. Then, all vertices in A_1 are matched to some other vertices, defining B_1. When we remove two matched vertices from the graph G^1_{out} at Lines 29 and 40, we update the degrees of their remaining neighbors, and append the vertices which have degrees of 1 to the queue. During Phase-1 of Karp-Sipser, we also maintain the set of A_i and B_i vertices, while storing only the last one. A_ℓ and B_ℓ are returned along with the number ℓ of levels, which is computed thanks to the use of the marker #.

Apart from the scaling step, the proposed heuristic in Algorithm 1 has linear worst-case time complexity of $O(n + \tau)$. The scaling step, if applied until convergence, can take more time than that. We do do not suggest running it until convergence; for all practical purposes 5 or 10 iterations of the basic method [18] or even less of the Newton iterations [17] seem sufficient (see experiments). Therefore, the practical run time of the algorithm is linear.

3.2 ONE-OUT**: Analysis.** Let a_i and b_i be two random variables representing the cardinalities of A_i and B_i, respectively, in an execution of KS on a random 1-out graph. Then, the KS algorithm matches b_ℓ edges in the first phase, and leaves $a_\ell - b_\ell$ vertices unmatched. What remains after the first phase is a set of cycles. In the bipartite graph case [6], all vertices in those cycles are matchable and hence the cardinality of the matching was measured by $n - a_\ell + b_\ell$. Since we can possibly have odd cycles after the first phase, we cannot match all remaining vertices in the general case of undirected graphs. Let c be a random variable representing the number of odd cycles after the

Algorithm 2 KarpSipser$_{\text{One-Out}}$

Input $G^1_{out} = (V, E)$:
Output `match`$[\cdot]$: the mates of vertices
Output ℓ: the number of levels in the first phase
Output A: the set of degree-1 vertices in the first phase
Output B: the set of vertices matched to A vertices

`match`$[u] \leftarrow$ NIL for all u
$Q \leftarrow \{v : \deg(v) = 1\}$ ▶ degree-1 or in-degree 0
if $Q = \emptyset$ **then**
  $\ell \leftarrow 0$ ▶ no vertex in level 1
else
  $\ell \leftarrow 1$
Enqueue$(Q, \#)$ ▶ marks the end of the first level
Phase-1$\leftarrow$ ongoing
$A \leftarrow B \leftarrow \emptyset$
while true do
  while $Q \neq \emptyset$ **do**
    $u \leftarrow$ Dequeue(Q)
    if $u = \#$ **and** $Q = \emptyset$ **then**
      break the while-Q-loop
    else if $u = \#$ **then**
      $\ell \leftarrow \ell + 1$
      Enqueue$(Q, \#)$ ▶ new level formed
      skip to the next while-Q-iteration
    if `match`$[u] \neq$ NIL **then**
      skip to the next while-Q-iteration
    for $v \in \text{adj}(u)$ **do**
      if `match`$[v] =$ NIL **then**
        `match`$[u] \leftarrow v$
        `match`$[v] \leftarrow u$
        if Phase-1 = ongoing **then**
          $A \leftarrow A \cup \{u\}$
          $B \leftarrow B \cup \{v\}$
        $N \leftarrow \text{adj}(v)$
        $G^1_{out} \leftarrow G^1_{out} \setminus \{u, v\}$
        Enqueue(Q, w) for $w \in N$, $\deg(w) = 1$
        break the for-v-loop
    if Phase-1 = ongoing **and** `match`$[u] =$ NIL **then**
      $A \leftarrow A \cup \{u\}$ ▶ u cannot be matched
  Phase-1 $\leftarrow$ done
  if $E \neq \emptyset$ **then**
    pick a random edge (u, v)
    `match`$[u] \leftarrow v$
    `match`$[v] \leftarrow u$
    $N \leftarrow \text{adj}(\{u, v\})$
    $G^1_{out} \leftarrow G^1_{out} \setminus \{u, v\}$
    Enqueue(Q, w) for $w \in N$, $\deg(w) = 1$
  else
    break the while-true loop

first phase of Karp-Sipser. Then we have the following (obvious) lemma about the approximation guarantee of Algorithm 1.

LEMMA 3.2. *At the end of execution, the number of unmatched vertices is* $a_\ell - b_\ell + c$. *Hence, Algorithm 1 matches at least* $n - (a_\ell - b_\ell + c)$ *vertices.*

We need to quantify $a_\ell - b_\ell$ and c in Lemma 3.2. In Section 3.2.1, we obtain an upper bound on $a_\ell - b_\ell$. Then in Section 3.2.2, we obtain a pessimistic upper bound on c, which we improve with a more detailed analysis in Section 3.2.3. While the bound for $a_\ell - b_\ell$ holds for any graph with total support presented to Algorithm 1, the bounds for c are shown for random 1-out graphs of complete graphs. By plugging the bounds for these quantities, we obtain the following theorem on the approximation guarantee of Algorithm 1.

THEOREM 3.1. *Algorithm 1 obtains a matching with cardinality at least* $0.866 - \frac{\lceil 1.04 \log(0.336n) \rceil}{n}$ *in expectation, when the input is a complete graph.*

The theorem can also be interpreted more theoretically as a random 1-out graph has a maximum cardinality of a matching at least $0.866 - \frac{\lceil 1.04 \log(0.336n) \rceil}{n}$, in expectation. The bound is in close vicinity of 0.866, which was the proved bound for the bipartite case [6]. We note that in deriving this bound we assumed a random 1-out graph (as opposed to a random 1-out subgraph of a given graph) only at a single step. We leave the extension to this latter case as future work and present experiments suggesting that the bound is also achieved for this case. In Section 4, we empirically show that the same bound also holds for graphs whose corresponding matrices do not have total support.

3.2.1 An upper bound on $a_\ell - b_\ell$. In order to measure $a_\ell - b_\ell$, we adapt a proof from earlier work [6], which was inspired by Karp and Sipser's analysis of the first phase of their heuristic [14]. Let $\alpha_j^{(k)} = \Pr(v_j \in A_k)$ be the probability of a vertex v_j to belong to A_k, and similarly $\beta_j^{(k)} = \Pr(v_j \in B_k)$ be the probability of a vertex v_j to belong to B_k.

LEMMA 3.3. *It holds that*

$$\beta_j^{(k)} \geq 1 - e^{-\sum_i s_{ij} a_i^{(k)}} \tag{3.1}$$

$$\alpha_j^{(k)} \leq e^{1-\sum_i s_{ij} \beta_i^{(k-1)}} . \tag{3.2}$$

The proof of the lemma can be derived by following the bipartite case [6, Lemmas 6 and 7]. That is why, we give a high level sketch.

Proof. (Sketch) By Lemma 3.1 and its corollary, A_ℓ and B_ℓ define a bipartition. Therefore, the equations remain the same. The only technicality in the proof is to make sure that the vertices in A_k and those in B_k are from the same set V and to take the bipartition into account. □

Thanks to Lemma 3.3, we can now bound $a_\ell - b_\ell$.

LEMMA 3.4. $a_\ell - b_\ell \leq (2\Omega - 1)n$, *where* $\Omega \approx 0.567$ *equals to* $W(1)$ *of Lambert's W function.*

Proof. In expectation, $a_\ell = \sum_{i=1}^n \alpha_i^{(\ell)}$ and $b_\ell = \sum_{i=1}^n \beta_i^{(\ell)}$. The expected difference is

$$a_\ell - b_\ell = \sum_{i=1}^{n} \alpha_i^{(\ell)} - \beta_i^{(\ell)} .$$

From Lemma 3.3 and another result of Dufossé et al. [6, Lemma 8], we have $\alpha_i^{(\ell)} - \beta_i^{(\ell)} \leq (2\Omega - 1)$ and hence

$$a_\ell - b_\ell \leq \sum_{i=1}^{n} (2 \cdot 0.567 - 1) \ \leq 0.134 \cdot n \ . \qquad \square$$

3.2.2 An upper bound on c. We now investigate c, the number of odd cycles that remain on a G_{out}^1 graph after the first phase of Karp-Sipser.

LEMMA 3.5. *We have* $c \leq \frac{n - a_\ell - b_\ell}{3}$.

Proof. After the first phase, we have removed A_ℓ and B_ℓ from V. Therefore, the number of vertices remaining for the second phase is $n - a_\ell - b_\ell$. The shortest odd cycle is of length 3 and so we can derive the result by assuming that all vertices belong to such cycles. $\square$

We need a lower bound on $a_\ell + b_\ell$ to bound c in Lemma 3.5. We start by bounding a_ℓ first.

LEMMA 3.6. *For random 1-out graphs with* $n \geq 30$ *vertices,* $a_\ell \geq 0.361n$.

Proof. As $A_i \subseteq A_{i+1}$, we have $\alpha_i^{(\ell)} \geq \alpha_i^{(1)}$, hence $a_\ell \geq a_1$. The probability of a vertex u_i to be in A_1 is $\alpha_i^{(1)} = \left(1 - \frac{1}{n-1}\right)^{n-1}$, since u_i should not be chosen by any other vertex. For $n \geq 30$, we have $\frac{1}{e} \geq \left(1 - \frac{1}{n-1}\right)^{n-1} \geq 0.361$. This concludes the proof by the linearity of expectation. $\square$

We use the result of the above lemma to bound b_ℓ.

LEMMA 3.7. *For random 1-out graphs with* $n \geq 30$ *vertices,* $b_\ell \geq 0.303n$.

Proof. We start again with $b_\ell \geq b_1$, as $B_i \subseteq B_{i+1}$. By (3.1),

$$b_\ell \geq \sum_j 1 - e^{-\sum_i s_{ij} \alpha_i^{(1)}} \ ,$$

and using Lemma 3.6,

$$b_\ell \geq \sum_j 1 - e^{-0.361} \ ,$$

for $n \geq 30$. This simplifies to the stated result. $\square$

The proofs used in obtaining the bounds in Corollary 3.2 needed to assume a random 1-out graph only for lower bounding $\alpha_i^{(1)}$. This quantity can also be bound in graphs with uniform vertex degrees. We sum the two lower bounds from Lemmas 3.6 and 3.7 in a corollary.

COROLLARY 3.2. $a_\ell + b_\ell \geq 0.664n$ *for random 1-out graphs with* $n \geq 30$ *vertices.*

At this point, we have a bound on the number of matched vertices using Lemmas 3.2, 3.4, 3.5, and Corollary 3.2 as $n - a_\ell + b_\ell - \frac{n - a_\ell - b_\ell}{3} \geq 0.754n$.

3.2.3 An improved bound on c. Consider the family of graphs in which all vertices have degree two. As graphs from this family have disjoint cycles; we call them cyclic graphs. Let C_M denote a random graph from this family with M vertices. There are $!M$ cyclic graphs with M vertices, as derangements create cycles. The $n - a_\ell - b_\ell$ vertices remaining after the first phase of Karp-Sipser form a $C_{n-a_\ell-b_\ell}$ by the principal of deferred decisions [24, p. 9]. This is so, as the edge chosen by a vertex u is incident on a remaining vertex, and u is chosen by exactly one of the remaining vertices. We will find an upper bound on the expected number of odd cycles in a $C_{n-a_\ell-b_\ell}$ and improve Lemma 3.5.

For a vertex $u_i \in C_M$, let h_i be the length of the cycle containing it, and define a variable $Y_i = \frac{1}{h_i}$ if h_i is odd and $Y_i = 0$ otherwise. Then, $\sum_{u_i \in C_M} Y_i$ is the number of odd cycles in C_M. To see why, consider an odd cycle of length k and observe that each of its vertices contributes $\frac{1}{k}$ for counting the cycle. We now measure the expected value of Y_i for a vertex u_i.

LEMMA 3.8. *In a random cyclic graph with M vertices, $\Pr(h_i = k) \leq \frac{1.04}{M}$ for any vertex u_i when $k < M - 2$, and $\Pr(h_i = k) \leq \frac{1.365}{M}$ for $k = M - 2$.*

Proof. We have the formula whose explanation follows:

$$\Pr(h_i = k) = \frac{\binom{M-1}{k-1} \cdot (k-1)! \cdot !(M-k)}{!M}\,.$$

We select a set of $k - 1$ vertices which will form a cycle of length k with u_i. There are $\binom{M-1}{k-1}$ different sets, and there are $(k-1)!$ possible cycles with the selected set. The remaining $M - k$ vertices must also lie in cycles; there are $!(M-k)$ ways to achieve this. Furthermore, with M vertices there are $!M$ ways to create a union of disjoint cycles.

Let us manipulate the formula as

$$\begin{aligned}\Pr(h_i = k) &= \frac{\binom{M-1}{k-1} \cdot (k-1)! \cdot !(M-k)}{!M} \\ &= \frac{(M-1)! \cdot !(M-k)}{!M \cdot (M-k)!} \\ &= \frac{1}{M} \cdot \frac{M! \cdot !(M-k)}{!M \cdot (M-k)!}\,.\end{aligned}$$

For $M \geq 5$, we have $\frac{M!}{!M} \leq 2.73$. If $M - k \geq 4$, $\frac{(M-k)!}{!(M-k)} \geq 2.64$, and $\frac{M!}{!M}\frac{(M-k)!}{!(M-k)} \leq \frac{2.73}{2.64} \leq 1.04$. When $M - k = 3$, we have $\frac{3!}{!3} = 3$, and $\frac{M!}{!M}\frac{(M-k)!}{!(M-k)} \leq \frac{2.73}{3} \leq 1 \leq 1.04$. When $M - k = 2$, we have $\frac{2!}{!2} = 2$, and the upper bound is $2.73/2 = 1.365$. □

LEMMA 3.9. *In a random cyclic graph with M vertices, $\mathbb{E}[Y_i] \leq \frac{1.04}{M} \cdot \log(M) + \frac{0.1625}{M}$ for a vertex u_i.*

Proof.

$$\begin{aligned}
\mathbb{E}[Y_i] &= \sum_{k=3,odd}^{M} \Pr(h_i = k) \cdot \frac{1}{k} \\
&\leq \sum_{k=1}^{M} \Pr(h_i = k) \cdot \frac{1}{k} \\
&\text{(by splitting } 1.365/M \text{ for } k = M-2\text{)} \\
&\leq \left(\sum_{k=1}^{M} \frac{1.04}{M} \cdot \frac{1}{k} \right) + \frac{(1.365 - 1.04)}{M} \cdot \frac{1}{M-2} \\
&\leq \frac{1.04}{M} \cdot \sum_{k=1}^{M} \frac{1}{k} + \frac{0.1625}{M} \\
&\leq \frac{1.04}{M} \cdot \log(M) + \frac{0.1625}{M} \,. \qquad \square
\end{aligned}$$

LEMMA 3.10. *The number of odd cycles in C_M is less than or equal to $\lceil 1.04 \log(M) \rceil$.*

Proof. By the linearity of expectation, we find this formula by summing $\mathbb{E}[Y_i]$ over all u_i in C_M.□

We plug in the number of vertices remaining after the first phase of Karp-Sipser into the formula and obtain the following corollary.

COROLLARY 3.3. *The expected number of cycles remaining after the first phase of Karp-Sipser in random 1-out graphs is less than or equal to $\lceil 1.04 \log(n - a_\ell - b_\ell) \rceil$, which is also less than or equal to $\lceil 1.04 \log(0.336n) \rceil$.*

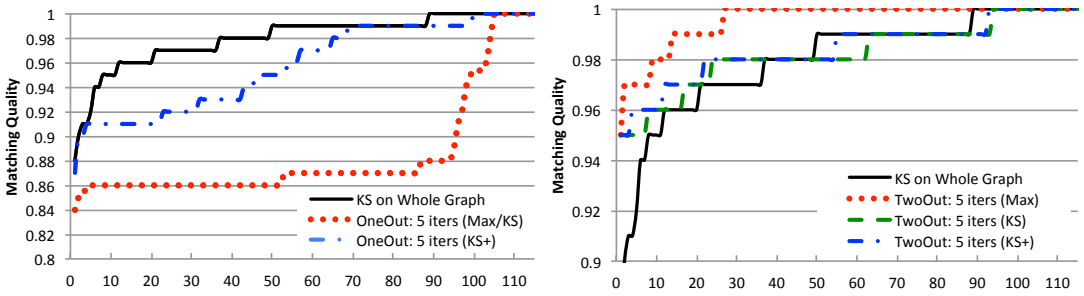
3.3 Variants. Here we summarize two related theoretical random bipartite graph models that we adapt to the undirected case using similar algorithms. The presentation will be brief and without proofs; we will present experiments in Section 4.1.

The random $1 + e^{-1}$ undirected graph model (see the bipartite version [12] summarized in Section 2.3) lets each vertex choose a random neighbor. Then, the vertices that have not been chosen select one more neighbor. The maximum cardinality of a matching in the subgraph consisting of the identified edges can be computed as an approximate matching in the original graph. As this heuristic delivers perfect matchings with high probability in the uniform bipartite case, we expect perfect or near perfect matchings in the general case.

A model richer in edges is the random 2-out graph model. In this model, each vertex chooses two neighbors. There are two enticing characteristics of this model in the uniform bipartite case. First, the probability of having a perfect matching goes to one with the increasing number of vertices [32]. Second, there is a special algorithm for computing the maximum cardinality matchings in these (bipartite) graphs [13] with high probability, in linear time in expectation.

4 Experiments

To understand the effectiveness and efficiency of the proposed heuristics in practice, we report the matching quality and various statistics that appear in our analyses in Section 3. For the experiments, we used three graph datasets: (1) the first set is generated with matrices from SuiteSparse Matrix



(a) 1-out graphs (b) 2-out graphs

Figure 1: The matching qualities sorted in increasing order for KS on the original graph, S5-KS and S5-KS+ on 1-out and 2-out graphs. The figures also contain the quality of the maximum cardinality matchings in these graphs. The experiments are performed on the 115 graphs in the first dataset.

Collection (`https://sparse.tamu.edu/`). We investigated all $n \times n$ matrices from the collection with $50000 \leq n \leq 100000$. For a matrix from this set, we removed the diagonal entries, symmetrized the pattern of the resulting matrix, and discarded a matrix if it has empty rows. There were 115 matrices at the end which we used as the adjacency matrix. (2) The graphs in the second dataset are synthetically generated to make KS deviate from the optimal matching as much as possible. This dataset contains five graphs with different hardness levels for KS. (3) The third set contains five large, real-life matrices from the SuiteSparse collection for measuring the run time efficiency of the proposed heuristics.

4.1 A comprehensive evaluation. We use MATLAB to measure the matching qualities based on the first two datasets. For each matrix, five runs are performed with each randomized matching heuristic and the average is reported. One, five and ten iterations are performed to evaluate the impact of the scaling method.

Table 1 summarizes the quality of the matchings for all the experiments on the first dataset. The matching quality is measured as the ratio of the matching cardinality to the maximum cardinality matching in the original graph. The table presents statistics for matching qualities of KS performed on the original graphs (first row), 1-out graphs (the second set of rows), Karoński-Pittel-like $(1 + e^{-1})$-out graphs (the third set of rows), and 2-out graphs (the last set of rows).

For the U-Max rows, we construct k-out graphs by using uniform probabilities while selecting the neighbors as proposed in the literature [8, 12]. We compute the cardinality of the maximum matchings in these k-out graphs to the maximum matching cardinality on the original graphs and report the statistics. The rows St-Max report the same statistics for the k-out graphs constructed by using probabilities with $t \in \{1, 5, 10\}$ scaling iterations. These statistics serve as upper bounds on the matching qualities of the proposed St-KS heuristics which execute KS on the k-out graphs obtained with t scaling iterations. Since St-KS heuristics use only a subgraph, the matchings they obtain are not maximal with respect to the original edge set. The proposed St-KS+ heuristics exploit this fact and apply another KS phase on the subgraph containing only the unmatched vertices to improve the quality of the matchings. The table does not contain St-Max rows for 1-out graphs since KS is an optimal algorithm for these subgraphs.

	Alg.	Min	Max	Avg.	GMean	Med.	StDev
	KS	0.880	1.000	0.980	0.980	0.988	0.022
1-out	U-Max	0.168	1.000	0.846	0.837	0.858	0.091
	S1-KS	0.479	1.000	0.869	0.866	0.863	0.059
	S5-KS	0.839	1.000	0.885	0.884	0.865	0.044
	S10-KS	0.858	1.000	0.889	0.888	0.866	0.045
	S1-KS+	0.836	1.000	0.951	0.950	0.953	0.043
	S5-KS+	0.865	1.000	0.958	0.957	0.968	0.036
	S10-KS+	0.888	1.000	0.961	0.961	0.971	0.033
$(1+e^{-1})$-out	U-Max	0.251	1.000	0.952	0.945	0.967	0.081
	S1-Max	0.642	1.000	0.967	0.966	0.980	0.042
	S5-Max	0.918	1.000	0.977	0.977	0.985	0.020
	S10-Max	0.934	1.000	0.980	0.979	0.985	0.018
	S1-KS	0.642	1.000	0.963	0.962	0.972	0.041
	S5-KS	0.918	1.000	0.972	0.972	0.976	0.020
	S10-KS	0.934	1.000	0.975	0.975	0.977	0.018
	S1-KS+	0.857	1.000	0.972	0.972	0.979	0.025
	S5-KS+	0.925	1.000	0.978	0.978	0.984	0.018
	S10-KS+	0.939	1.000	0.980	0.980	0.985	0.016
2-out	U-Max	0.254	1.000	0.972	0.966	0.996	0.079
	S1-Max	0.652	1.000	0.987	0.986	0.999	0.036
	S5-Max	0.952	1.000	0.995	0.995	0.999	0.009
	S10-Max	0.968	1.000	0.996	0.996	1.000	0.007
	S1-KS	0.651	1.000	0.974	0.974	0.981	0.035
	S5-KS	0.945	1.000	0.982	0.982	0.984	0.013
	S10-KS	0.947	1.000	0.984	0.983	0.984	0.012
	S1-KS+	0.860	1.000	0.980	0.979	0.984	0.020
	S5-KS+	0.950	1.000	0.984	0.984	0.987	0.012
	S10-KS+	0.952	1.000	0.986	0.986	0.987	0.011

Table 1: For each matrix in the first dataset and each proposed heuristic, five runs are performed the statistics are performed over the mean of these results; the minimum, maximum, arithmetic and geometric averages, median and standard deviation are reported.

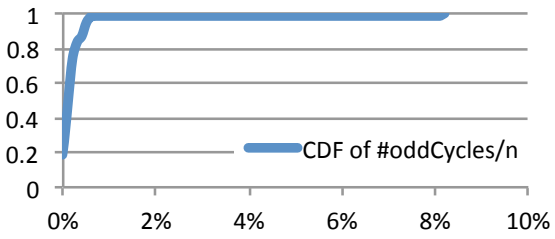


Figure 2: The cumulative distribution of the ratio of number of odd cycles remaining after the first phase of KS in One-Out to the number of vertices in the graph.

As Table 1 shows, more scaling iterations increase the maximum matching cardinalities on k-out graphs. Although this is much more clear when the worst-case graphs are considered, it can also be observed for arithmetic and geometric means. Since U-Max is the no scaling case, the impact of the first scaling iteration (S1-KS vs U-Max) is significant. On the other hand, the difference on the matching quality for S5-KS and S10-KS is minor. Hence, five scaling iterations are deemed sufficient for the proposed heuristics in practice.

As the theory suggests, the heuristics St-KS perform well for $(1 + e^{-1})$-out and 2-out graphs. With $t \in \{5, 10\}$, their quality is almost on par with KS on the original graph, and even better for 2-out graphs. In addition, applying KS on the subgraph of unmatched vertices to obtain a maximal matching does not increase the matching quality much. Since this subgraph is small, the overhead of this extra work will not be significant. Furthermore, this extra step significantly improves the matching quality for 1-out graphs which a.a.s. do not have a perfect matching.

To better understand the practical performance of the proposed heuristics and the impact of the additional KS execution, we profile their performance by sorting their matching qualities in increasing order for all 115 matrices. Figure 1a plots these profiles on 1-out and 2-out graphs for the heuristics with five scaling iterations. As the first figure shows, five iterations are sufficient to obtain 0.86 matching quality except 1.7% of the 1-out experiments. The figure also shows that the maximum matching cardinality in a random 1-out graph is worse than what KS can obtain on the original graph. This is why although S5-KS finds maximum matchings on 1-out graphs, its performance is still worse than KS. The additional KS in S5-KS+ closes almost all of this gap and makes the matching qualities close to those of KS. On the contrary, for 2-out graphs generated with five scaling iterations, the maximum matching cardinality is more than the cardinality of the matchings found by KS. There is still a gap between the best possible (red line) and what KS can find (blue line) on 2-out graphs. We believe that this gap can be targeted by specialized, efficient exact matching algorithms for 2-out graphs.

There are 30 rank-deficient matrices without total support among the 115 matrices in the first dataset. We observed that even for 1-out graphs, the worst-case quality for S5-KS is 0.86 and the average is 0.93. Hence, the proposed approach also works well for rank-deficient matrices/graphs in practice.

Since the number c of odd cycles at the end of the first phase of KS is a performance measure, we investigate it. For each matrix, we compute the ratio c/n. We then plot the cumulative distribution of these values in Fig. 2. For all the experiments except one, this ratio is less than 1%. For the extreme case, the ratio increases to 8%. In that particular case, the matching found in the 1-out

(a) $h = 2$ (b) $h = 512$

Figure 3: Bad matrices for KS with $h = 2$ and $h = 512$.

subgraph is maximum for the input graph (i.e., the number of odd components is also large in the original graph).

4.2 Bad matrices for KS**.** Let $\mathbf{A}$ be an $n \times n$ symmetric matrix, V_1 be the set of $\mathbf{A}$'s first $n/2$ vertices, and V_2 be the set of $\mathbf{A}$'s last $n/2$ vertices. As Figure 3 shows, $\mathbf{A}$ has a full $V_1 \times V_1$ block, i.e., a clique, and an empty $V_2 \times V_2$ block. The last $h \ll n$ vertices of V_1 are connected to all the vertices in the corresponding graph. The block $V_1 \times V_2$ has a nonzero diagonal, hence the corresponding graph has a perfect matching.

On such a matrix with $h = 0$, the KS heuristic consumes the whole graph during Phase 1 and finds a maximum cardinality matching. When $h > 1$, Phase 1 immediately ends, since there is no degree-one vertex. In Phase 2, the first edge (nonzero) consumed by KS is selected from a uniform distribution over the nonzeros whose corresponding rows and columns are still unmatched. Since the block $V_1 \times V_1$ forms a clique, it is more likely that the nonzero will be chosen from this block. Thus, a vertex in V_1 will be matched with another vertex from V_1, which is a bad decision since the block $V_2 \times V_2$ is completely empty. Hence, we expect a decrease on the performance of KS as h increases. On the other hand the probability that the proposed heuristics chooses an edge from that block goes to zero, as those entries cannot be in a perfect matching.

Table 2 shows that the quality of KS drops to 0.63 as h increases. In comparison, One-Out heuristic with five scaling iterations maintains a good quality for small h values. However, a better scaling with more iterations (10 and 20 for $h = 128$ and $h = 512$, respectively) is required to guarantee the desired matching quality—see the scaling error in the table.

4.3 Experiments on large-scale graphs. This experiments are performed on a machine running 64-bit CentOS 6.5, has 30 cores each of which is an Intel Xeon CPU E74870 v2 core operating at 2.30 GHz. To choose five large-scale matrices from SuiteSparse, we first sorted the pattern-symmetric matrices in the collection in decreasing order of their nonzero count. We then chose the five largest matrices from different families to increase the variety of experiments. The details of these matrices are given in Table 3. This table also contains the run times and matching qualities of the original KS, and the proposed One-Out and Two-Out heuristics. The proposed heuristics have five scaling iterations and also apply KS at the end for ensuring maximality.

The run time of the proposed heuristics are analyzed in four stages in Table 3. The *Scale* stage scales the matrix with five iterations, the *k-out* stage chooses the neighbors and constructs the k-out subgraph, the *KS* stage applies KS on the k-out subgraph, and *KS+* is the stage for the additional Karp-Sipser at the end. The total time with these four stages are also shown. The

		One-Out					
	KS	5 iters.		10 iters.		20 iters.	
h	quality	error	quality	error	quality	error	quality
2	0.96	7.54	0.99	0.68	0.99	0.22	1.00
8	0.78	8.52	0.97	0.78	0.99	0.23	0.99
32	0.68	6.65	0.81	1.09	0.99	0.26	0.99
128	0.63	3.32	0.53	1.89	0.90	0.33	0.98
512	0.63	1.24	0.55	1.17	0.59	0.61	0.86

Table 2: Results for the bad matrices with $n = 5000$ and different h values. One-Out is executed with 5, 10, and 20 scaling iterations and scaling errors are also reported. Averages of five are reported for each cell.

					OneOut-S5-KS+						TwoOut-S5-KS+					
			KS			Execution time (seconds)						Execution time (seconds)				
Matrix	$\|V\|$	$\|E\|$	*Quality*	*time*	*Quality*	*Scale*	*1-out*	*KS*	*KS+*	Total	*Quality*	*Scale*	*2-out*	*KS*	*KS+*	Total
`cage15`	5.2	94.0	1.00	5.82	0.93	0.67	0.85	0.65	0.05	2.21	0.99	0.67	1.48	1.78	0.01	3.94
`dielFilterV3real`	1.1	88.2	0.99	3.36	0.98	0.52	0.36	0.07	0.01	0.95	0.99	0.52	0.62	0.16	0.00	1.30
`hollywood-2009`	1.1	112.8	0.93	4.18	0.91	1.12	0.45	0.06	0.02	1.65	0.95	1.12	0.76	0.13	0.01	2.01
`nlpkkt240`	28.0	746.5	0.98	52.95	0.93	4.44	4.99	3.96	0.27	13.66	0.98	4.44	9.43	10.56	0.11	24.54
`rgg_n_2_24_s0`	16.8	265.1	0.98	19.49	0.93	2.33	2.33	2.08	0.17	6.91	0.98	2.33	4.01	6.70	0.11	13.15

Table 3: Summary of the results with five large-scale matrices for original KS and the proposed One-Out and Two-Out heuristics which generate maximal matchings with extra KS and five scaling iterations.

quality results are consistent with the experimental results obtained on the first dataset. As the table shows, the proposed heuristics are faster than the original KS on the input graph. For 1-out graphs, the proposed approach is 2.5–3.9 faster than KS on the original graph. The speedups are in between 1.5–2.6 for 2-out graphs with five iterations.

5 Conclusion

We proposed heuristics for approximating the maximum cardinality matchings on general (undirected) graphs. The heuristics scale the adjacency matrix of a given graph with doubly stochastic scaling algorithms, and then selects a subgraph on which a maximum cardinality matching is obtained. We showed that the obtained matching approximates the maximum cardinality matching on the original graph; both in theory and in practical experiments. Our theoretical analysis can be perceived as an analysis of the well-known Karp-Sipser heuristic on the random 1-out graph model. Our experiments suggest that one of the heuristics has performance on par with Karp-Sipser whilst being faster and more reliable.

A more rigorous treatment and elaboration of the variants described in Section 3.3 seem worthwhile. Although Karp-Sipser heuristic work well for these graphs, we wonder if there are exact, linear time algorithms for $(1 + e^{-1})$-out and 2-out graphs. We also plan to work on the parallel implementations of these algorithms.

References

[1] N. Blum. A new approach to maximum matching in general graphs. In *ICALP '90*, pages 586–597, London, UK, UK, 1990. Springer-Verlag.

[2] R. A. Brualdi and H. J. Ryser. *Combinatorial Matrix Theory*, volume 39. Cambridge University Press, 1991.

[3] P. J. Cameron. Notes on combinatorics. `https://cameroncounts.files.wordpress.com/2013/11/comb.pdf`, Last checked Nov. 2017.

[4] J. Cheriyan. Randomized $\tilde{O}(M(|V|))$ algorithms for problems in matching theory. *SIAM J. Comput.*, 26(6):1635–1655, December 1997.

[5] I. S. Duff, K. Kaya, and B. Uçar. Design, implementation, and analysis of maximum transversal algorithms. *ACM T. Math. Software*, 38(2):13:1–13:31, 2011.

[6] F. Dufossé, K. Kaya, and B. Uçar. Two approximation algorithms for bipartite matching on multicore architectures. *J. Parallel Distr. Com.*, 85:62–78, 2015.

[7] J. Edmonds. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, 1965.

[8] A. M. Frieze. Maximum matchings in a class of random graphs. *J. Comb. Theory B*, 40(2):196 – 212, 1986.

[9] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for network problems. *SIAM J. Comput.*, 18(5):1013–1036, 1989.

[10] N. J. A. Harvey. Algebraic algorithms for matching and matroid problems. *SIAM J. Comput.*, 39(2):679–702, 2009.

[11] O. H. Ibarra and S. Moran. Deterministic and probabilistic algorithms for maximum bipartite matching via fast matrix multiplication. *Infor. Process. Lett.*, pages 12–15, 1981.

[12] M. Karoński and B. Pittel. Existence of a perfect matching in a random $(1+e^{-1})$–out bipartite graph. *J. Comb. Theory B*, 88(1):1–16, 2003.

[13] R. M. Karp, A. H. G. Rinnooy Kan, and R. V. Vohra. Average case analysis of a heuristic for the assignment problem. *Math. Oper. Res.*, 19:513–522, 1994.

[14] R. M. Karp and M. Sipser. Maximum matching in sparse random graphs. In *FOCS'81*, pages 364–375, Nashville, TN, USA, 1981.

[15] K. Kaya, J. Langguth, F. Manne, and B. Uçar. Push-relabel based algorithms for the maximum transversal problem. *Comput. Oper. Res.*, 40(5):1266–1275, 2013.

[16] P. A. Knight. The Sinkhorn–Knopp algorithm: Convergence and applications. *SIAM J. Matrix Anal. A.*, 30(1):261–275, 2008.

[17] P. A. Knight and D. Ruiz. A fast algorithm for matrix balancing. *IMA J. Numer. Anal.*, 33(3):1029–1047, 2013.

[18] P. A. Knight, D. Ruiz, and B. Uçar. A symmetry preserving algorithm for matrix scaling. *SIAM J. Matrix Anal. A.*, 35(3):931–955, 2014.

[19] J. Langguth, F. Manne, and P. Sanders. Heuristic initialization for bipartite matching problems. *J. Exp. Algorithmics*, 15:1.1–1.22, 2010.

[20] L. Lovász. On determinants, matchings, and random algorithms. In *FCT*, pages 565–574, 1979.

[21] L. Lovasz and M. D Plummer. *Matching theory*. Elsevier Science Publishers, Netherlands, 1986.

[22] A. Meir and J. W. Moon. The expected node-independence number of random trees. *Indagationes Mathematicae (Proceedings)*, 76(4):335–341, 1973.

[23] S. Micali and V. V. Vazirani. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *FOCS'80*, pages 17–27, 1980.

[24] M. Mitzenmacher and E. Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[25] M. Mucha and P. Sankowski. Maximum matchings in planar graphs via Gaussian elimination. In S. Albers and T. Radzik, editors, *ESA 2004*, pages 532–543, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[26] M. Mucha and P. Sankowski. Maximum matchings via Gaussian elimination. In *FOCS '04*, pages 248–255, Washington, DC, USA, 2004. IEEE Computer Society.

[27] P. A. Peterson and M. C. Loui. The general maximum matching algorithm of Micali and Vazirani. *Algorithmica*, 3(1):511–533, Nov 1988.

[28] M. O. Rabin and V. V. Vazirani. Maximum matchings in general graphs through randomization. *J. Algorithm.*, 10(4):557–567, December 1989.

[29] D. Ruiz. A scaling algorithm to equilibrate both row and column norms in matrices. Technical Report TR-2001-034, RAL, 2001.

[30] R. Sinkhorn and P. Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific J. Math.*, 21:343–348, 1967.

[31] V. V. Vazirani. An improved definition of blossoms and a simpler proof of the MV matching algorithm. *CoRR*, abs/1210.4594, 2012.

[32] D. W. Walkup. Matchings in random regular bipartite digraphs. *Discrete Math.*, 31(1):59–64, 1980.