2012 IEEE 53rd Annual Symposium on Foundations of Computer Science

Randomized Greedy Algorithms for the Maximum Matching Problem with New Analysis

Matthias Poloczek
Institute of Computer Science
Goethe University
Frankfurt, Germany
Email: matthias@thi.cs.uni-frankfurt.de

Mario Szegedy
Department of Computer Science
Rutgers University
Piscataway, USA
Email: szegedy@cs.rutgers.edu

***Abstract*—It is a long-standing problem to lower bound the performance of randomized greedy algorithms for maximum matching. Aronson, Dyer, Frieze and Suen [1] studied the modified randomized greedy (MRG) algorithm and proved that it approximates the maximum matching within a factor of at least $\frac{1}{2}+1/400,000$. They use heavy combinatorial methods in their analysis. We introduce a new technique we call *Contrast Analysis*, and show a $\frac{1}{2}+1/256$ performance lower bound for the MRG algorithm. The technique seems to be useful not only for the MRG, but also for other related algorithms.**

***Keywords*-matching; randomized; greedy; approximation;**

I. Introduction

This presentation follows the long tradition of analyzing (random) greedy algorithms for the maximum cardinality matching problem. Researchers have been concerned with their performance on worst case instances as well as on random graph instances. We have results on both, but our hardest results are for worst case instances. Some of the literature we survey below deal with the weighted case. Our results will exclusively deal with the un-weighted case, and so do our references unless we explicitly say otherwise.

A classical result by Korte and Hausmann [2], dating back to 1978, gives that the greedy algorithm that always picks a new edge (disjoint of the ones selected so far) with the largest weight, achieves an approximation ratio of at least $\frac{1}{2}$. Moreover, they show that for every graph (besides some trivial exceptions) for the worst weight function the greedy algorithm cannot beat $\frac{1}{2}$. By picking the edges at random instead (in the un-weighted case), the approximation improves only to $\frac{1}{2}+o(1)$ on worst-case inputs, but a significant benefit is achieved for sparse graphs and trees [3]. [4] deals with the weighted case where the probability of picking an edge is proportional to its weight, and gives improved bounds for some graph classes.

Karp and Sipser [5] in 1981 invented an algorithm that matches an arbitrary vertex of degree one, if such a vertex exists, and picks a random edge otherwise. Aronson, Frieze and Pittel [6] prove that this small modification yields a matching that is within $n^{1/5+o(1)}$ of maximum size on random graph instances with edge probability c/n.

Random cubic graph instances are studied in [7]. They proved that the MinGreedy algorithm of [8], which picks a vertex of minimal degree randomly and matches it to a random neighbor, leaves at most $o(n)$ vertices unmatched, on expectation.

Despite the success on random graph instances, the $\frac{1}{2}$–barrier for worst case instances remained unchallenged, until Aronson, Dyer, Frieze and Suen [1] made a breakthrough by managing to analyze a modification of the randomized greedy algorithm first proposed in [8]. They call this algorithm MRG (Modified Randomized Greedy). This variant picks each next edge by first selecting a random node (with non-zero degree) of the remaining graph and then a random neighbor of it. (We keep deleting nodes that become endpoints of edges in M, together with their incident edges.) They proved that this algorithm has an approximation factor of at least $\frac{1}{2}+1/400,000$ for all graphs. While this tiny advance over $\frac{1}{2}$ was very important theoretically, experiments suggest, that the worst case performance of MRG could be as large as $\frac{2}{3}$. There are also theoretical evidences for this: A sequence of results [8], [9], [10] give, that on random graphs with edge probability c/n the MRG algorithm outputs a matching of expected size $\left(\frac{1}{2}-\frac{\log(2-e^{-c})}{2c}\right)n$. Our first theorem improves on [1]:

Theorem 1. *The approximation ratio of the* MRG *algorithm for any graph G is at least $\frac{1}{2}+\frac{1}{256}$ (on expectation, over all random choices of the algorithm).*

Our proof takes use of a new technique we call *Contrast Analysis*, at the heart of which we have a *Contrast Lemma*.

Many researchers have also looked at the case when $G=(V=L\cup R,E)$ is bipartite. In an interesting model of [11] the algorithm has control over the order of the nodes on the L side, while the ordering on the R side is

Matthias Poloczek's research was partially supported by DFG SCHN 503/1-5.

Mario Szegedy's research was partially supported by NSF grant CCF-0832787.

0272-5428/12 $26.00 © 2012 IEEE
DOI 10.1109/FOCS.2012.20

adversarial. The RANKING algorithm of Karp, Vazirani and Vazirani picks a random permutation π on L and matches the adversarially arriving vertex from R to a free neighbor in L of minimal rank in π, thereby achieving an approximation ratio of $1 - 1/e \approx 0.632$ [12], [13]. If the order on the R-side is also random, the same algorithm performs better; [14] gives a performance guarantee of 0.696 (independently [15] shows a bound of 0.653).

Jukna and Schnitger recently started to investigate the non-bipartite graph version of RANKING. We call this version 'RANKING' as well, and from now on under this term we exclusively mean the general version. This algorithm is likely to have a significantly better worst case performance than MRG (perhaps as large as 0.727), but paradoxically, showing a greater-than-$1/2$ approximation ratio has proven to be an even harder challenge than before. The difference between MRG and RANKING is that in the latter we once and for all pick a random permutation, π, of the vertices, and when we select a "random" neighbor to the new node (new nodes are simply selected in the order of π) we again use π to pick its neighbor (the first available node in π, i.e. with the lowest rank). The lower bound method of Aronson, Dyer, Frieze and Suen breaks down, because RANKING uses only a fraction of the randomness that MRG does, which makes it harder to apply any independence argument. Very recently we have learned that Pushkar Tripathi, independently from us, has invented and studied RANKING, and in his thesis [16] establishes (referring to joint work [17]):

Theorem 2. *The approximation ratio of the* RANKING *algorithm for any graph G is at least 0.56 (on expectation, over all random choices of the algorithm).*

We hope, that with our new technique we will be able to improve on this bound. What we can give now is a Contrast Lemma for RANKING.

For the special case of bipartite graphs, however, the better guarantee of 0.696, obtained in the Online Bipartite Matching Model [14], carries over to (our version of) RANKING, as well as the upper bound 0.727 of [15]. Independently from us, this has also been observed by Tripathi [16]. We give the proof in the full version of our paper. The RANKING algorithm, just as MRG, can be run in linear time (see Sect. VI). We have also studied the performance of RANKING on random instances (see Sect. V):

Theorem 3. *For random graphs, where each edge is selected with probability $p = \omega\left(\frac{\log n}{n}\right)$, algorithm* RANKING *matches almost all nodes of the input graph with high probability.*

This result is an analogue of the results of [8], [9], [10] for RANKING.

A. Randomized Greedy Algorithms

As customary, n will denote the number of vertices of a graph instance G. Many known randomized greedy algorithms for the maximum cardinality matching problem follow a scheme described in this section. For a node v let $N(v)$ denote its set of neighbors of v. Moreover, a vertex is called *free* with respect to a matching M if it is not an endpoint of any edge in M.

Definition 1 (Schedule)**.** *A* schedule σ *specifies* $n+1$ *permutations* π_O, $\pi_{v_1}, \ldots, \pi_{v_n}$ *on* $V(G)$.

A randomized greedy algorithm A first chooses a schedule randomly according to a distribution on S_n^{n+1} (here S_n is the set of all permutations on $V(G)$). The distribution depends on the type of greedy algorithm we consider, and may depend on G (in our cases it will not). After this random choice the algorithm proceeds *deterministically*: In round l, for $l = 1, 2, \ldots, n$, the algorithm selects node $v = \pi_O(l)$ and matches v to the first free neighbor according to π_v. However, if v is already matched or if it has no free neighbor, A skips to the next round. At the end of round n the algorithm returns the produced matching. We shall investigate the following two types of greedy algorithms:

- The MRG algorithm chooses π_O, $\pi_{v_1}, \ldots, \pi_{v_n}$ uniformly at random from S_n^{n+1}.
- The RANKING algorithm fixes all n+1 permutations of the schedule to the same permutation π that is drawn uniformly at random from S_n.

II. KEY IDEAS

A. Proof Outline

It is known [1] and nearly trivial, that it is enough to concentrate on graphs that have a perfect matching (see also Sect. II-D). We define the *approximation ratio* (or *performance*) of a randomized greedy matching algorithm A on a graph G as:

$$r = \frac{\mathbb{E}\left[\text{size of the matching the algorithm } A \text{ gives on } G\right]}{\text{the size of the maximum matching in } G}.$$

We say that a node v of G is *covered* by a specific run of algorithm A, if during that run A adds an edge to its output-matching that is incident to v. The first thing to be noticed is that running *any* schedule σ on input graph G results in covering at least one endpoint of every edge $e = (u, v) \in E(G)$. For this reason any greedy algorithm A that follows the scheme in Sect. I-A has an approximation ratio of at least $\frac{1}{2}$ on any graph G. To show a approximation ratio better than $\frac{1}{2}$ for a randomized greedy algorithm, it is sufficient to prove, that for a constant fraction of the edges of the optimal matching both endpoints are covered with non-negligible probability. Unlike [1], which takes a global view of the constructed matching and then examines paths of length three formed in the union of the constructed

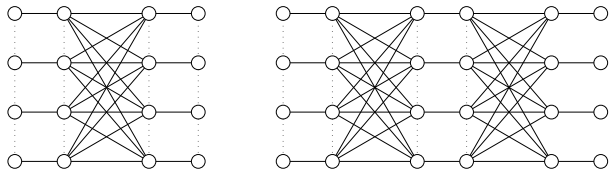

Figure 1. The bomb–graph B_4 (l) and the double bomb–graph D_4 (r)

Algorithm	Lamp	KVV	B_{900}	D_{900}
MRG	0.806	0.785	0.670	0.697
RANKING	0.797	0.911	0.751	0.738

Table I
EXPERIMENTALLY DETERMINED APPROXIMATION RATIOS

and the optimum matching, we are going to give a simple condition, which guarantees that both endpoints of a given edge e of G (without any assumption on e) are covered with positive constant (in terms of n) probability. We show that this probability is going to be $o(1)$ only if both endpoints of e die very early (within the first $o(n)$ rounds) in the matching process (on expectation). A node by definition "dies" if the algorithm covers it or if the node becomes isolated (because all of its neighbors get covered, and so itself loses the chance of getting covered later). Indeed, if one endpoint of an edge does not die shortly after the other does, then throughout all those rounds when the second node is still alive, it might be chosen as an endpoint. And if it becomes covered, then both endpoints will be covered. In the above discussion "short" and "long" means $o(n)$ and $\Omega(n)$ rounds, respectively, and we always talk about a "typical" run.

What will be harder to rule out is the possibility that both endpoints of e survive for $\Omega(n)$ rounds, but then shortly after the first endpoint is covered, the other one becomes isolated (all is understood on expectation). To show that this is impossible we need to prove that when a node u becomes covered, it will not suddenly change the perspectives of v (its pair) getting covered (unless u was covered with the (u, v) edge, which is, however, good for us). The problem boils down to an interesting clear-cut problem, which is to "contrast" (i.e. relate) the coverage probabilities of a node v by some algorithm A for G and for $G \backslash u$, respectively (for any $u \neq v \in V(G)$). We establish the desired relation, when A is the MRG algorithm, and when A is the RANKING algorithm.

B. Hard Instances

While MRG has been studied extensively, RANKING (for general graphs) is a recent invention of Jukna and Schnitger (also, independently in [16], [17]), inspired by the ideas in [11].

The motivation was to supersede the performance of MRG on hard graphs. A classical hard graph for greedy randomized matching algorithms is the so called "bomb–graph". In the bipartite adaption of [3] B_n consists of a complete bipartite graph on $2n$ vertices, the *core*. Moreover, each node of the core is connected to an additional node, its *antenna* (cp. Fig. 1). The antenna edges form the only perfect matching of the graph.

The (uniform) random edge heuristic will almost exclusively pick edges from the core, reaching an approximation ratio $\frac{1}{2} + o(1)$. MRG has a chance of at least $\frac{1}{2}$ to select an antenna and thereby collect an edge that belongs to the perfect matching. The benefit is substantial as the approximation ratio on B_n converges to $\frac{2}{3}$ (for $n \to \infty$).

What Jukna and Schnitger have observed is that RANKING performs better on B_n than MRG, as bad decisions of collecting edges inside the core are now compensated for. In case of MRG core vertices that have low rank according to π_O are likely to be selected as first endpoints and pick (uniformly at random) a core vertex as mate. In case of RANKING the reduced randomness plays a crucial role: Since both endpoints of a matching edge are chosen according to the same permutation π, the second endpoint is likely to have a low rank according to π as well, and hence two dangerous vertices are eliminated at once. As a consequence, the probability of antenna nodes to be matched is increased and the approximation ratio is boosted to 0.75. For a long time 0.75 seemed to be the worst approximation ratio of RANKING, until the double bomb–graph (first proposed for another reason in [15]; cp. Fig. 1) was found. On this graph RANKING has an approximation ratio of at most 0.727 when n tends to infinity, the current worst number for RANKING.

Table I contains some experimental results (all approximation ratios are averaged over 1 million repetitions). The "Lamp graph" consists of two triangles connected via a path with two internal vertices. The "KVV graph" is the well-known counter-example of [11]. For this bipartite graph $G = (L \cup R, E)$ we set $|L| = |R| = 450$; then the i-th node of R is adjacent to the vertices $i, i+1, \ldots, 450$ in L (for $i = 1, 2, \ldots, 450$). To our table we have also added B_{900} (Bomb) and D_{900} (Double Bomb).

C. Basic Equations for Covering Probabilities

This section is the beginning of our rigorous mathematical discussion. G will be our input graph, that we assume, has a perfect matching (see next section for justification). First we establish some trivial, but important equations. For every schedule of Definition 1, it holds, that by the time we finish running it on G, every node $v \in V(G)$ will end up either selected as a first endpoint of an edge of the final matching, or picked as a second endpoint (picked as a neighbor of some first endpoint selected in the same round) or it becomes isolated in the end. (The unmatched nodes will form an

independent set in the end.) Whether we talk about MRG or RANKING, the set of all schedules forms the basic set of a probability space (only the probability measure depends on the algorithm). For this probability space we define:

$s(v)$	the probability that the node gets selected as the first endpoint of a matching edge.
$k(v)$	the probability that the node gets selected as the second endpoint of a matching edge.
$i(v)$	the probability that the node becomes isolated (un-matched) in the end.

Clearly,

$$s(v) + k(v) + i(v) = 1 \quad \text{(for every } v\text{)} \tag{1}$$

$$\sum_v s(v) = \sum_v k(v) \tag{2}$$

Also, let M be the matching the algorithm creates, and let $\mathbb{E}$ denote expectation. Then

$$\sum_v s(v) = \mathbb{E}[|M|]. \tag{3}$$

For an edge $e = (v, w)$ let $b(e)$ be the probability that both endpoints of e are matched in the end (maybe or may not be with each other). Since at least one of the endpoints of e must be matched in the end, we have that

$$s(v) + k(v) + s(w) + k(w) = 1 + b(e). \tag{4}$$

If M^* is a perfect matching of the input graph G with $n/2$ edges, Equations (2)-(4) give:

$$\mathbb{E}[|M|] = n/4 + \sum_{e \in M^*} b(e)/2.$$

We would like to show that $\sum_{e \in M^*} b(e)/2 = \Omega(n)$ with a hopefully large constant.

Let $e = (v, w)$ be an arbitrary edge of G. We shall lower bound $b(e)$ (the probability that both endpoints of e are covered) in terms of $s(v)$ for both MRG and RANKING (separately) *without any additional assumption on* e. For MRG the desired relation between $s(v)$ and $b((v, w))$ is stated by Lemma 6, and it is used in Sect. III-B to finish the proof. (Nothing else is used in the tiny Sect. III-B, except trivial monotonicity.)

D. Basic Lemmas

The lemmas of this technical section are essentially known to the randomized matching community. The proofs therefore can be skipped, but for the sake of completeness, and for getting the exact formulations for our more general randomized greedy algorithm framework we include them. Our goal here is to establish certain monotonicity and independence (more precisely, disjointness) properties: for instance it is shown that in the analysis we may assume that our input graph G has a perfect matching. In the sequel, let A denote a randomized greedy algorithm (as defined in Sect. I-A) that runs schedule σ.

Lemma 1 (Locality Lemma)**.** *Let M be the matching obtained by running schedule σ under A, and let $X \subseteq V(G)$ be such that the $(X, V(G) \setminus X)$ partition does not have any edge of M across the parts.*

Let N be the subset of edges of M induced on X and $\sigma|_X$ be the schedule induced on X, i.e. all vertices not in X are omitted in all permutations. Then, if we run A on $G|_X$ (the induced subgraph of G on X), with the schedule $\sigma|_X$, it returns N (i.e. the algorithm works obliviously to the environment of X).

Proof: Imagine running the algorithm on the entire $V(G)$, and also on X, with the schedules σ and $\sigma|_X$. Run the two algorithms simultaneously in such a way, that whenever the first algorithm selects a node v not belonging to X as a first endpoint, we switch the second algorithm temporarily off. The condition of the lemma guarantees that in this case we never select a neighbor in X.

On the other hand, if a node inside X is selected according to π_O, the algorithm run on G will never pick a neighbor (if it picks any) outside X. Thus, whenever we arrive at processing a node $v \in X$, the two algorithms pick the first available neighbor (according to π_v) inside X, which by induction will be always the same for both algorithms. ■

Corollary 1. *Let σ be a fixed schedule on the nodes of G, on which we run A. Let M be the resulting matching. Then $V(G) \setminus V(M)$ is an independent set in G and hence the cardinality of M is at least half of a maximum matching.*

Next we prove the property of "vertex monotonicity" that has been shown for the MRG algorithm in [1].

Lemma 2 (Vertex Monotonicity)**.** *Let M be the matching obtained by running schedule σ, and let C be the set of nodes that M covers. Assume, we insert a new node v in the graph and connect it to some nodes in $V(G)$. Update σ by inserting v in each of the permutations $\pi_O, \pi_{v_1}, \ldots, \pi_{v_n}$ of σ arbitrarily (but keeping the relative order of all nodes) and extending the schedule by a new permutation π_v on $V(G) \cup \{v\}$ (while keeping the remaining edges of M).*

Then, when we run the algorithm, we get a new matching M'. Let $C' = V(M')$. Then either $C' \supseteq C$ or $|C \setminus C'| = 1$ and $v \in C'$.

Proof: We claim that M' is either a superset of M, or results from it, by switching edges along a single alternating path that starts at v. This of course, immediately implies the lemma.

In order to show the claim, consider the connected components of $M \cup M'$. We are done, if we manage to prove that except perhaps the component that contains v all components are single edges. Indeed, consider a connected component X of $M \cup M'$ that does not contain v. If we

apply Lemma 1 on X for both G and $G+v$, then we get that $N = M|_X$ and $N' = M'|_X$ must be both identical with the matching that we get, when we run the algorithm on X (here we use that v is not included in X). Thus $N = N'$, and since $N \cup N'$ by our assumption was connected, they are both the same single edge. ■

Corollary 2. *Regarding the approximation ratio of a randomized greedy algorithm A we may assume that the graph has a perfect matching.*

Proof: We fix any maximum matching M^* in a graph and assume that M^* leaves some nodes uncovered. By Lemma 2, removing these nodes does not increase the size of the matching computed by A. Hence, the approximation ratio is not improved if we restrict ourselves to graphs containing a perfect matching. ■

Let $x_1, \ldots, x_k \in V(G)$ and define $G' := G \setminus \{x_1, \ldots, x_k\}$. The final matching obtained by running A on G and G' resp. is denoted by M and M' resp.

Lemma 3. *The symmetric difference of the matchings obtained by running a given schedule σ on G and G', respectively, contains k alternating paths $P_1, \ldots, P_k$ starting at nodes $x_1, \ldots x_k$, respectively. (We define a path as an ordered sequence of nodes.) It might occur that $P_i = x_i$ or that $P_i = P_j$. In the latter case P_i starts at node x_i and ends at x_j. The same conclusion holds in the more general setting, where we run only the first l rounds of σ on G and G', respectively.*

Proof: We proceed by induction on l (the number of rounds run so far). Initially $P_i = x_i$ for all i, which reflects the situation before the first round. When we go into the l^{th} round, the algorithm selects $\pi_O(l)$, i.e. the vertex of rank l in permutation π_O, as first endpoint. If $\pi_O(l)$ is an endpoint of some P_i, then either the run on G or the run on G' cannot collect an edge at this round. This is because the endpoint of P_i is either matched in G or in G' or it is x_i. Thus the edge that does get collected in this round for either G or G' (if at all) will belong to the symmetric difference, and thus augments the alternating path that had $\pi_O(l)$ as endpoint. Note that the other endpoint of the new edge cannot be a middle point of any path, nor the starting point, unless the latter is an empty path. It is possible, however, that the new edge joins two paths. Assume now that $\pi_O(l)$ is not an endpoint of some P_i. Then when picking the second endpoint, we might get two different new matching edges for G and G', but this can happen only, when at least one of these endpoints coincides with the endpoint of some path P_i. In the latter case P_i extends by $\pi_O(l)$ and possibly by another point (which can be an endpoint for another paths, thus joining the two paths). ■

E. Contrast Analysis and Contrast Lemmas

After several failed attempts to duplicate the lower bound theorem of [1] for RANKING, we felt that we were missing statements that tie the $s(v)$ and $k(v)$ values of different graphs together. First we found such a lemma for the MRG, and later, with a much greater effort for RANKING. We call these 'Contrast Lemmas' as they allow us to compare the s and k values of related graphs with each other. The lemma for MRG gives a little more, and as a result, we get sharper lower bounds for MRG.

Why do Contrast Lemmas help us in our lower bound proofs? As the algorithm proceeds, these lemmas allow us to monitor the development of the expected future coverage of each node, and allow us to contrast different stages of the algorithm. Importantly, they state that a sudden change will not happen in the expected covering probabilities of a node, if the algorithm matches a separate node (unless through their connecting edge). Recall our heuristic reasoning from Sect. II-A that we want to prove that if a node dies (after surviving long), its mate (w.r.t. to the optimal matching M^*) will not die immediately by isolation. Indeed, it cannot, because as its mate, that node is also a long survivor, so its s value (before the death of the mate) is not (close to) zero. Then the Contrast Lemma guarantees that its s value cannot change suddenly to (or close to) zero, when the mate dies.

Lemma 4 (Contrast Lemma for the MRG algorithm)**.** *Let A be the* MRG *algorithm, and $u, x_1, \ldots, x_k \in V(G)$. The probability that u covered in $G \setminus \{x_1, \ldots, x_k\} =: G'$, under running A, is at least*

$$\frac{1}{k+1} \cdot \Bigg[\text{Prob}(u \textit{ is covered in } G \textit{ (under } A\textit{)}) - \sum_{j=1}^{k} \text{Prob}(u \textit{ is matched to } x_j \textit{ in } G \textit{ (under } A\textit{)}) \Bigg].$$

The proof can be found in Sect. III-C. A similar lemma is stated for RANKING in Sect. IV.

III. PUTTING THEOREM 1 TOGETHER

A. Estimating b from s

Towards our crucial Lemma 6 we show:

Lemma 5. *Both for the* RANKING *algorithm and for the the* MRG *algorithm the following holds: Let $e = (v, w)$ be an edge of G. Then $b(e) + s(w) + k(w) \geq s(v)$. (In fact, for* MRG *an even a stronger statement holds: $b(e) + s(w) \geq s(v)$.)*

Proof: First we prove this for the RANKING algorithm. Let $\Gamma(b(e))$ denote the set of schedules for which both v and w are covered, and define $\Gamma(s(v))$, $\Gamma(s(w))$ and $\Gamma(k(w))$ analogously. It is clearly sufficient to establish

an injective mapping from $\Gamma(s(v))$ to the multiset $\mathcal{M} := \Gamma(b(e)) \cup \Gamma(s(w)) \cup \Gamma(k(w))$ for RANKING (if a schedule is contained in two or three of the sets $\Gamma(b(e))$, $\Gamma(s(w))$, $\Gamma(k(w))$, it appears twice (resp. three times) in $\mathcal{M}$).

Recall that in case of RANKING any schedule with non-zero probability is defined by a single permutation, π, and with a little abuse of the notation we call the associated schedule also π. Let $\pi \in \Gamma(s(v))$. If w is an endpoint in the final matching produced by π, then $\pi \in \Gamma(b(e))$ holds and $\pi \in \Gamma(s(v))$ is mapped to that occurrence of π in $\Gamma(b(e)) \subseteq \mathcal{M}$. Otherwise v must precede w in π, since with RANKING the situation can never occur, that the lower ranked endpoint of an edge is not covered at all, while the higher ranked endpoint is selected as a first endpoint of a matching edge (since in the said situation the higher ranked endpoint must be free at the time the lower ranked endpoint is considered, and by default, it would match with it). Let π' be the permutation we obtain by swapping v and w in π (thus, lowering the rank of w), and let us focus on the round when RANKING (now run with π') considers w: If w has already been picked by a vertex in the prefix, π' contributes to $\Gamma(k(w))$. Otherwise it must contribute to $\Gamma(s(w))$. To see this, observe that no matching decision about v was altered by moving v to higher ranks (when processing the nodes up-to its old rank), so at the moment when w is considered by π', v is still free, and provides a default match. The mapping we define now goes from $\pi \in \Gamma(s(v))$ to π', which is either contained in $\Gamma(k(w))$ or in $\Gamma(s(w))$, and is charged to that set.

To conclude the argument, assume that $\pi \in \mathcal{M}$ is an image under our mapping and observe that its origin can be determined uniquely: If π is charged to $\Gamma(b(e))$, then π itself is the origin. Otherwise, the origin is obtained by swapping v and w.

Now let us switch to the MRG algorithm. We are proving the stronger statement: $b(e) + s(w) \geq s(v)$. Set $\mathcal{M} = \Gamma(b(e)) \cup \Gamma(s(w))$. Assume $\sigma = \pi_O, \pi_{v_1}, \ldots, \pi_{v_n} \in \Gamma(s(v))$, i.e. for schedule σ vertex v is selected as a first endpoint in some round l. If w has already been matched, σ contributes not only to $s(v)$, but also to $b(e)$ and we map $\sigma \in \Gamma(s(v))$ to $\sigma \in \Gamma(b(e))$.

Now assume that w is still free at the beginning of round l. Let us denote by σ' the schedule that one obtains by swapping v and w in π_O, while keeping the remaining part of σ the same. If we run MRG with schedule σ', then w will be selected as a first endpoint of a matching edge in round l, because v must be free at that time. In this second situation we map $\sigma \in \Gamma(s(v))$ to $\sigma' \in \Gamma(s(w))$. Similarly to the proof for RANKING we can now observe that the destination of our map reveals the origin: For $\sigma' \in \Gamma(b(e))$ the origin is σ' itself, and in the case, when $\sigma' \in \Gamma(s(w))$, the origin is obtained by swapping v and w in π_O of σ'. ■

The next lemma will use all our lemmas so far, crucially the Contrast Lemma, to make a simple relation between $b((v,w))$ and $s(v)$ (for any edge (v,w) of G) in the case of MRG.

Lemma 6. *For any edge $e = (v,w)$ in G we have $b(e) \geq s(v)^2/8$.*

We prove the statement by induction on the number of the edges of the graph G. If $E(G)$ is just the (v,w) edge, the statement clearly holds. Now consider an arbitrary graph G with edge (v,w) and let $\beta := s(w) + k(w)$. If $\beta \leq \frac{7}{8}s(v)$ then by Lemma 5 we have $b(e) \geq s(v) - \beta$, which gives $b(e) \geq s(v)/8 \geq s(v)^2/8$. We need one more caveat. Let s_e be the probability that any time during the algorithm the edge $e = (v,w)$ is selected from any side. If $s_e \geq s(v)/8$, we are again done outright.

Now we are ready for the inductive step, where we can assume that $\beta > \frac{7}{8}s(v)$ and $s_e < s(v)/8$. We subdivide the event space created by the randomness of the algorithm into disjoint events. Let $\mathcal{E}_{x,y}$ be the event that in the first round (x,y) is selected from the x side. (Remark: by definition, if $(x,y) \in E(G)$, then $(y,x) \in E(G)$.) Define also: $\mathcal{E}_x = \cup_y$ is a neighbor of x $\mathcal{E}_{x,y}$. Then

$$\mathcal{E}_v;\ \{\mathcal{E}_{x,y}\}_{x,y}, \quad \text{where } (x,y) \in E(G) \text{ and } x \neq v$$

is obviously a complete system of disjoint events. Define

q	:=	$\frac{1}{n} = Prob(\mathcal{E}_v)$
$q_{x,y}$	:=	$Prob(\mathcal{E}_{x,y})$
$s_{x,y}$	:=	the probability that v is selected (in any round) from the v side, conditioned on $\mathcal{E}_{x,y}$
$b_{x,y}$	:=	the probability that both v and w are covered (in some rounds), conditioned on $\mathcal{E}_{x,y}$
b	:=	the probability that w is covered (in some round), conditioned on $\mathcal{E}_v$

Now we have

$$1 = q + \sum_{x \neq v \text{ and } (x,y) \in E(G)} q_{x,y} \tag{5}$$

$$s(v) = q + \sum_{x \neq v \text{ and } (x,y) \in E(G)} q_{x,y} s_{x,y}$$

$$b(e) = qb + \sum_{x \neq v \text{ and } (x,y) \in E(G)} q_{x,y} b_{x,y}$$

Lemma 7. *For all $(x,y) \in E(G)$, $x \neq v$ we have $b_{x,y} \geq s_{x,y}^2/8$.*

Proof: For (x,y) we separate four different cases. If $\{x,y\} \cap \{v,w\} = \emptyset$, then we are done by induction, since the $b_{x,y}$ and $s_{x,y}$ values are respectively the $b(e)$ and $s(v)$ values of G', where G' is the graph that arises after the first step of the MRG algorithm, i.e. when after (x,y) is selected.

If $x = w$ and $y \neq v$, we are done, since when $x = w$, w is guaranteed to be covered, so $b_{x,y} \geq s_{x,y}$, (Since $b_{x,y}$ allows for v to be covered from the non-v side, while $s_{x,y}$ does not, equation does not necessarily hold). If $y = w$ (but $x \neq v$) we are done, since $b_{x,y} \geq s_{x,y}$ (the same reason as previously). Finally, if $y = v$ we are done, since $s_{x,y} = 0$, because v is matched from the "wrong" side. We have considered all cases, since $x = v$ is not permitted by assumption. ∎

Lemma 8. *If $\beta > \frac{7}{8}s(v)$ and $s_e < s(v)/8$, then $b \geq s(v)/4$.*

Proof: If the degree d of v in G is one, then $b = 1$, and the statement is trivial. Otherwise let $w, x_1, \ldots, x_{d-1}$ be the d neighbors of v. We apply the Contrast Lemma (Lemma 4) to every G, G_i pair ($1 \leq i \leq d-1$), where $G_i = G \setminus \{v, x_i\}$. For $1 \leq i \leq d-1$ let p_i be zero if x_i is not a neighbor of w, otherwise we define p_i as the probability that any time during the algorithm the edge (w, x_i) is selected *from any side*. Recall, that s_e is the probability that any time during the algorithm the edge $e = (v, w)$ is selected from any side. Then

$$\beta \geq s_e + p_1 + \ldots + p_{d-1}$$

Lemma 4 gives that the probability that w is matched in G_i is at least $(\beta - s_e - p_i)/3$. Thus, the conditional probability that w is covered, after v is selected as a first endpoint of a matching edge in the first round, is lower bounded by

$$\frac{1}{d} + \frac{\beta - s_e - p_1}{3d} + \ldots + \frac{\beta - s_e - p_{d-1}}{3d} \tag{6}$$

Here the first term corresponds to the case, where w is picked as the partner of v, and the i^{th} subsequent term lower bounds the probability that x_i is picked as a neighbor of v, and w is covered at a later point. The expression in Equation (6) can be further expressed as

$$\frac{1}{d} + \frac{1}{3d}\left(\beta(d-1) - s_e(d-1) - \sum_{i=1}^{d-1} p_i\right) \geq$$
$$\frac{1}{d} + \frac{d-2}{3d}(\beta - s_e) \geq \frac{1}{d} + \frac{d-2}{4d}s(v) \geq \frac{s(v)}{4}$$

∎

We are now ready to do the recurrence:

$$\begin{aligned} b(e) &= qb + \sum_{x \neq v \text{ and } (x,y) \in E(G)} q_{x,y} b_{x,y} \\ &\geq qb + \frac{1}{8} \sum_{x \neq v \text{ and } (x,y) \in E(G)} q_{x,y} s_{x,y}^2 \\ &\geq qb + \frac{1}{8} \frac{\left(\sum_{x \neq v \text{ and } (x,y) \in E(G)} q_{x,y} s_{x,y}\right)^2}{\sum_{x \neq v \text{ and } (x,y) \in E(G)} q_{x,y}} \\ &\geq \frac{1}{n}\frac{s(v)}{4} + \frac{1}{8}\frac{\left(s(v) - \frac{1}{n}\right)^2}{1 - \frac{1}{n}} \end{aligned} \tag{7}$$

The first inequality follows from Lemma 7, the second from Jensen's inequality. The estimate of qb comes from Lemma 8. We have also used Equation (5), and the fact, that

$$\sum_{x \neq v \text{ and } (x,y) \in E(G)} q_{x,y} s_{x,y} = s(v) - \frac{1}{n}.$$

Estimating (7) further:

$$\frac{1}{n}\frac{s(v)}{4} + \frac{1}{8}\frac{(s(v) - \frac{1}{n})^2}{1 - \frac{1}{n}} \geq \frac{1}{n}\frac{s(v)}{4} + \frac{1}{8}\left(s(v) - \frac{1}{n}\right)^2$$
$$= \frac{1}{8}s(v)^2 + \frac{1}{8n^2} \geq \frac{s(v)^2}{8}$$

as needed.

B. The MRG *algorithm achieves at least a* $(1/2 + 1/256)$ *approximation*

Denote by M^* an arbitrary maximum matching of G and let $M^* = \{(v_i, w_i)\}_{1 \leq i \leq |M^*|}$ such that $s(v_i) \geq s(w_i)$. Recall that we may assume that M^* is a perfect matching because of the monotonicity property given by Lemma 2. Clearly,

$$\sum_i (s(v_i) + s(w_i)) = \sum_i (k(v_i) + k(w_i))$$

holds, because every edge selected by the algorithm has a first and a second endpoint. Moreover, since the algorithm obtains a maximal matching, we have

$$\sum_i (s(v_i) + s(w_i) + k(v_i) + k(w_i)) \geq |M^*|$$

and hence $\sum_i s(v_i) \geq |M^*|/4$.
Let $e_i = (v_i, w_i)$. Then Jensen's inequality and Lemma 6 give:

$$\sum_i b(e_i) \geq \sum_i \frac{s_i^2}{8} \geq \frac{1}{8}|M^*|\left(\frac{1}{|M^*|}\sum_i s_i\right)^2 \geq \frac{|M^*|}{128}$$

Then the size of the matching the MRG algorithm finds on expectation is at least

$$\frac{|M^*|}{2} + \frac{1}{2}\sum_i b(e_i) \geq (1/2 + 1/256) \cdot |M^*|.$$

C. The Proof of the Contrast Lemma (Lemma 4 for the MRG *algorithm)*

In order to be able to compare the running of MRG on two different graphs we recall the notion of *schedule* for MRG from Definition 1. A schedule σ is given by $n+1$ permutations on $V(G) = [n]$:

$$\sigma = \pi_O | \pi_{v_1} \ldots \pi_{v_n}$$

There are $(n!)^{n+1}$ schedules, and in case of MRG we equip the set of all schedules with the uniform distribution. It is easy to show that we can run the MRG algorithm on an

arbitrary subgraph H of G in such a way that we pick a random schedule and run it on this subgraph by skipping any vertex in π_O that is not contained in H.

To prove the claim, it is sufficient to define a max-$(k+1)$-to-one map ϕ from the set

$S_1 =$ schedules, that when run on G, node u is matched to a node in $V(G)\setminus\{x_1,\dots,x_k\}$

to the set

$S_2 =$ schedules (for G), that when run on G', node u is matched

Recall from Lemma 3, that when executing a given schedule, at the beginning of each round the symmetric difference of the matchings obtained on G and G' differ in at most k alternating paths P_i ($1 \le i \le k$). We are now ready to define $\phi(\sigma)$ for any $\sigma \in S_1$. First observe that if $u \notin P_i$ for any $1 \le i \le k$, then the node u appears either in both matchings or in none. Also, if u is an inner point of some P_i, then u appears in both matchings. This gives us the first rule:

1. Let $\sigma \in S_1$. If u is not an an endpoint of any P_i created by σ, then $\phi(\sigma) = \sigma$ ($\in S_2$).

The critical case corresponds to u being the endpoint of some P_i ($1 \le i \le k$). Now, walking backwards on P_i starting from u, let we first encounter node y (directly connected with u) and then node y'. Note, that y' exists, since $y \neq x_i$, by the definition of S_1.

2. For those $\sigma \in S_1$, where u is the endpoint of some P_i ($1 \le i \le k$) and σ chooses u as a first endpoint of a matched edge in G, define $\phi(\sigma)$ by swapping u with y' in π_O.

3. For those $\sigma \in S_1$, where u is the endpoint of some P_i ($1 \le i \le k$) and σ chooses u as a second endpoint of a matched edge in G, define $\phi(\sigma)$ by swapping u with y' in π_y.

It is easy to see that in items 2. and 3. $\phi(\sigma) \in S_2$ also holds. We are done if we show that any $\sigma' \in S_2$ can have at most $k+1$ inverse images. Trivially, σ' can have at most one inverse image from item 1. We show that σ' can have at most k inverse images from items 2. and 3 together. In fact, if σ' is such that G' chooses u as a first endpoint of an edge under σ', then for any σ with $\phi(\sigma) = \sigma'$ u is chosen for G by σ as a first endpoint of an edge, and the mapping was done under rule 2, otherwise for any σ with $\phi(\sigma) = \sigma'$ u is chosen for G by σ as a second endpoint of an edge, and the mapping was done under rule 3. Thus no σ' can come from 2. and 3. at the same time.

We can decode σ from σ' (of cases 2. or 3.) in such a such a way that we run schedule σ' until the l^{th} round, when u is chosen. Then we use the following lemma:

Lemma 9. *The P_is are the same for σ' and σ before the l^{th} round. Also, In the case, when σ' comes from 3, y can be identified as the other endpoint of the edge matched in G' under σ'.*

Both claims of the lemma are easy to see. The second claim holds, because when σ' comes from 3, for both σ and σ' in the l^{th} round y is the new vertex in π_O, and for σ and σ' the vertex u is chosen (in G, and respectively, in G') in this (i.e. in the same) round.

The gist of the proof is now, that we can identify y' as the endpoint of one of the P_is as we enter the l^{th} round. Therefore we have only k options for y'. The rest is simple: If u is chosen as a first endpoint for σ', then σ arises by swapping u and y' in π_O. Otherwise y is determined by σ', and σ arises by swapping y' in π_y. This procedure lets us recover at most k σs coming from 2. or 3.

IV. The Contrast Lemma for the Ranking Algorithm

Recall that the RANKING algorithm maintains a single random ordering π of the vertex set and chooses the next first endpoint as well as its neighbor according to π.
We show a (slightly weaker) analogue of Lemma 4 for the RANKING algorithm:

Lemma 10 (Contrast Lemma for the RANKING algorithm)**.** *Let $u, x \in V(G)$. Then the probability that u becomes an endpoint in the final matching in $G \setminus \{x\} =: G'$ is at least*

$$\frac{1}{2}\cdot\Big[\mathrm{Prob}(u \textit{ is matched in } G) - \mathrm{Prob}(u \textit{ is matched to } x \textit{ in } G)\Big].$$

In the proof of Lemma 10 we define a max-2-to-one map ϕ from the set

$S_1 =$ permutations, that when run on G, node u is matched to a node in $V(G)\setminus\{x\}$

to the set

$S_2 =$ permutations, that when run on $G' := G\setminus\{x\}$, node u is matched

Note: to run a permutation π on G', that was originally designed for G, one simply needs to leave out node x from π. We leave the the non-trivial proof that such map exists to the journal version.

V. The Ranking algorithm on Random Graphs

In the analysis of a graph algorithm it is customary to look at how it performs on random graphs. We look at the the performance of RANKING on $\mathcal{G}_{n,p}$, the Erdős-Rényi distribution of graphs on n nodes, where every edge is picked independently with probability p. We show that

Theorem 3, restated. *If $p = \omega(\frac{\log n}{n})$, then with high probability* RANKING *matches almost all nodes of $G \in_R \mathcal{G}_{n,p}$.*

It is sufficient to show:

Lemma 11. *Let $p = \omega(\frac{\log n}{n})$. Let us fix a permutation π of $[n]$, and let G range randomly in $\mathcal{G}_{n,p}$. Then with probability $1-1/n$ almost all ($n(1-o(1))$) nodes of G will be matched.*

Proof: Without loss of generality we may assume that $\pi[i] = i$ for all i. Our G now is not a fixed graph, but rather a random one, that we can generate as follows: As we process RANKING, and we reach node i, we generate all edges (i, j) of G, where $j > i$ on the fly. Once we have decided at edges from i, we can determine the edge that RANKING will pick in the i^{th} round (if it picks any) without knowing anything about the (existence of the) edges (i', j') for $i', j' > i$. Assume, node i is not yet picked in any of the previous rounds. Assume furthermore that we still have at least $L = n\varepsilon$ nodes with index greater than i that the algorithm has not matched yet with a node less than i. Conditioned on the current situation meeting the above criterions, when introducing the new edges in the i^{th} round, node i will have at least one edge to the un-matched node-set with probability at least $1-(1-p)^L \geq 1-n^{-\varepsilon pn/\log n} = 1-1/n^{\omega(1)} > 1-1/n^2$. If we have less than L un-matched nodes remaining, we abort the algorithm and return the current matching. Consider the probability that there exists a stage i that the algorithm leaves node i unmatched even though there are at least L future nodes that are not matched from earlier rounds. The union bound gives that this probability is at most $n \times \frac{1}{n^2} = 1/n$. Thus with probability at least $1-1/n$ at least $1-\varepsilon$ fraction of the nodes is matched. This is true for every fixed ε if n is large enough. ∎

In conclusion we get that the RANKING algorithm works on random graphs with efficiency approaching one for a wide range of parameter p. As a by-product we also got that a random graph with edge probability $\omega(\frac{\log n}{n})$ almost surely contains a near-perfect matching (i.e. a one with $n(0.5 - o(1))$ edges). (Of course the latter is not new and much stronger is known: Erdős and Rényi have shown that if a random graph has n vertices and $\frac{n}{2}(\ln n + \omega(1))$ edges, then G almost surely has a perfect matching [18].)

VI. An Efficient Implementation of Ranking and MRG

We describe an implementation of RANKING and MRG that runs in time $O(n+m)$. The main problems one needs to address are how to ensure that no endpoint is matched more than once and in case of the RANKING algorithm that the second endpoints are chosen according to the same random permutation.

Let us assume that the vertex identifiers are given by $[n]$. For the outer loop that selects the first endpoint a doubly linked list L_O is created that contains an element for each vertex. Moreover, we have A_O, an array of size n, where the i-th element (for $i \in [n]$) contains a pointer to the element of vertex i in L_O (or $\perp$ if the element of i is not contained in L_O anymore). Note that with an auxiliary array and the well–known Fisher-Yates shuffle algorithm we can obtain the random permutation π_O for A_O and L_O in time $O(n)$ [19].

Then we generate a pair

$$(\min\{\pi_O^{-1}(u), \pi_O^{-1}(v)\}, \max\{\pi_O^{-1}(u), \pi_O^{-1}(v)\})$$

for each edge $(u, v) \in E$ and collect all pairs in an array F (recall that $\pi_O^{-1}(w)$ denotes the rank of vertex w according to permutation π_O). F is sorted lexicographically in time $O(n + m)$; note that there are $2m$ pairs of integers that are bounded by n, the largest rank. A doubly linked adjacency list L_i is created for each vertex $i \in [n]$. In a single pass through F, for each tuple $(\pi_O^{-1}(u), \pi_O^{-1}(v))$ we append v in L_u as well as u in L_v and add mutual *cross–pointers* to both list elements; note that the vertices in each list are permuted according to π_O. This completes the construction for the RANKING algorithm. The MRG algorithm, however, requires each list L_i to be permuted randomly; the total cost of this additional step is $O(m)$.

Now we are ready to run the algorithm: As first endpoint we pick the first element of L_O, say u. If L_u is not empty, the second endpoint, say v, is the first element of L_u, otherwise we skip the round of u.

To assert that this procedure gives a valid matching, we remove u and v from L_O as well as from the adjacency lists of their neighbors, thereby maintaining the invariant that only free vertices appear in L_O and L_i. The removal is accomplished as follows: we pass once through L_u and L_v and for each vertex $w \in L_u$ (resp. $w \in L_v$) we access and remove the element of u (resp. v) in L_w in constant time, using the cross–pointers. Observe that the overall running time of this step is bounded by $O(m)$, since each edge incurs constant cost only once. Finally, u and v are deleted in L_O, before these changes are reflected in A_O accordingly.

Acknowledgment

The authors would like to thank Stasys Jukna and Georg Schnitger for pointing them to these questions. Moreover, the authors feel indebted to Ulrich Meyer and Georg Schnitger for their helpful comments.

References

[1] J. Aronson, M. E. Dyer, A. M. Frieze, and S. Suen, "Randomized greedy matching II," *Random Struct. Algorithms*, vol. 6, no. 1, pp. 55–74, 1995.

[2] B. Korte and D. Hausmann, "An analysis of the greedy algorithm for independence systems," *Annals of Discrete Mathematics*, vol. 2, pp. 65–74, 1978.

[3] M. E. Dyer and A. M. Frieze, "Randomized greedy matching," *Random Struct. Algorithms*, vol. 2, no. 1, pp. 29–46, 1991.

[4] Z. Miller and D. Pritikin, "On randomized greedy matchings," *Random Struct. Algorithms*, vol. 10, no. 3, pp. 353–383, 1997.

[5] R. M. Karp and M. Sipser, "Maximum matchings in sparse random graphs," in *FOCS*, 1981, pp. 364–375.

[6] J. Aronson, A. M. Frieze, and B. Pittel, "Maximum matchings in sparse random graphs: Karp-sipser revisited," *Random Struct. Algorithms*, vol. 12, no. 2, pp. 111–177, 1998.

[7] A. M. Frieze, A. J. Radcliffe, and S. Suen, "Analysis of a simple greedy matching algorithm on random cubic graphs," *Combinatorics, Probability & Computing*, vol. 4, pp. 47–66, 1995.

[8] G. Tinhofer, "A probabilistic analysis of some greedy cardinality matching algorithms," *Annals of Operations Research*, vol. 1, pp. 239–254, 1984.

[9] O. Goldschmidt and D. S. Hochbaum, "A fast perfect-matching algorithm in random graphs," *SIAM J. Discrete Math.*, vol. 3, no. 1, pp. 48–57, 1990.

[10] M. E. Dyer, A. M. Frieze, and B. Pittel, "The average performance of the greedy matching algorithm," *Ann. Appl. Probab.*, vol. 3, no. 2, pp. 526–552, 1993.

[11] R. M. Karp, U. V. Vazirani, and V. V. Vazirani, "An optimal algorithm for on-line bipartite matching," in *STOC*, 1990, pp. 352–358.

[12] B. E. Birnbaum and C. Mathieu, "On-line bipartite matching made simple," *SIGACT News*, vol. 39, no. 1, pp. 80–87, 2008.

[13] G. Goel and A. Mehta, "Online budgeted matching in random input models with applications to adwords," in *SODA*, 2008, pp. 982–991.

[14] M. Mahdian and Q. Yan, "Online bipartite matching with random arrivals: an approach based on strongly factor-revealing LPs," in *STOC*, 2011, pp. 597–606.

[15] C. Karande, A. Mehta, and P. Tripathi, "Online bipartite matching with unknown distributions," in *STOC*, 2011, pp. 587–596.

[16] P. Tripathi, "Allocation problems with partial information," Ph.D. dissertation, Georgia Institute of Technology, 2012.

[17] G. Goel and P. Tripathi, "Matching with our eyes closed," in *FOCS*, 2012.

[18] N. Alon and J. Spencer, *The Probabilistic Method*. John Wiley & Sons, 2008.

[19] R. Durstenfeld, "Algorithm 235: Random permutation," *Commun. ACM*, vol. 7, no. 7, p. 420, 1964.