Stochastic Matching with Commitment

Kevin P. Costello[2], Prasad Tetali[1], and Pushkar Tripathi[1]

[1] Georgia Institute of Technology
[2] University of California at Riverside

Abstract. We consider the following stochastic optimization problem first introduced by Chen et al. in [6]. We are given a vertex set of a random graph where each possible edge is present with probability p_e. We do not know which edges are actually present unless we scan/probe an edge. However whenever we probe an edge and find it to be present, we are constrained to picking the edge and both its end points are deleted from the graph. We wish to find the maximum matching in this model. We compare our results against the optimal omniscient algorithm that knows the edges of the graph and present a 0.573 factor algorithm using a novel sampling technique. We also prove that no algorithm can attain a factor better than 0.896 in this model.

1 Introduction

The matching problem has been a corner-stone of combinatorial optimization and has received considerable attention starting from the work of Jack Edmonds [9]. There has been recent interest in studying stochastic versions of the problem due to its applications to online advertising and several barter exchange settings [24, 19, 25]. Much of the recent research focused on studying matchings on *bipartite* graphs. In this paper we study a recently introduced variant on the stochastic online matching problem [6] on general graphs as described below.

For p a probability vector indexed by pairs of vertices from a vertex set V, let $G(V, p)$ denote an undirected Erdős-Rényi graph on V. That is, for any $(u, v) \in V \times V$, $p_{uv} = p_{vu}$ denotes the (known) probability that there is an edge connecting u and v in G. For every pair $(u, v) \in V \times V$ we are *not* told a priori whether there is an edge connecting these vertices, until we *probe/scan* this pair. If we scan a pair of vertices and find that there is an edge connecting them we are constrained to *pick* this edge and in this case both u and v are removed from the graph. However, if we find that u and v are not connected by an edge, they continue to be available (to others) in the future. The goal is to maximize the number of vertices that get matched.

We will refer to the above as the Stochastic Matching with Commitment Problem (SMCP), since whenever we probe a pair of adjacent vertices, we are committed to picking them. The performance of our algorithm is compared against the optimal offline algorithm that knows the underlying graph for each instantiation of the problem and finds the maximum matching in it. Note that since the input is itself random, the average performance of the optimal offline-algorithm is the expected size of the maximum matching in this random graph. We use the somewhat non-standard notation of $G(V, p)$, rather than $G_{n,p}$, since we will need to refer to the (fixed) vertex set V and also since p is a vector with typically different entries.

1.1 Our Results

It is easy to see that the simple greedy algorithm, which probes pairs in an arbitrary order, would return a *maximal* matching in every instance of the problem and is therefore a factor 0.5 approximation algorithm. We give a sampling based algorithm for this problem that does better than this:

Theorem 1. *There exists a randomized algorithm that attains a competitive ratio of at least* 0.575 *for the Stochastic Matching with Commitment Problem that runs in time* $\tilde{O}(n^4)$ *for a graph with* n *vertices. Furthermore, the running time can be reduced to* $\tilde{O}(n^3)$ *in the case where the expected size of the optimal matching is a positive fraction of the number of vertices in the graph.*

Our algorithm uses *offline simulations* to determine the relative importance of edges to decide the order in which to scan them. It is based on a novel sampling lemma that might be of independent interest in tackling online optimization settings, wherein an algorithm needs to make irrevocable online decisions with limited stochastic knowledge. This sampling trick is explained in section 2.3. Even though the proof for our sampling lemma is based on solving an exponentially large linear program, we also give a fast combinatorial algorithm for it (see Appendix B). As a result our algorithm can be implemented in time $O(n^3)$.

On the hardness front, we prove the following theorem, using rigorous analysis of the performance of the *optimal* online algorithm for a carefully chosen graph.

Theorem 2. *No algorithm can attain a competitive ratio better than* 0.896 *for the SMCP.*

1.2 Previous Work

The problem has similar flavor to several well known stochastic optimization problems such as the stochastic knapsack [7] and the shortest-path [21, 22]; refer to [26] for a detailed discussion on these problems. SMCP also models several problems of practical significance which are discussed in Appendix A. We will now present more explicit connections between SMCP and several previously studied models of matching with limited information.

Stochastic Matching Problems: Chen et al. [6] considered a more general model for stochastic matching than the one presented above. In their model every vertex $v \in V$ had a *patience parameter* $t(v)$ indicating the maximum number of failed probes v is willing to participate in. After $t(v)$ failed attempts, vertex v would leave the system, and would not be considered for any further matches. Our model can be viewed as a special case of their setting where $t(v) = n$ for every vertex. However Chen et al., and subsequently Bansal et al. [3], compared their performance against the optimal online algorithm. This was necessary because if we consider the case of the star graph, where each edge has a probability of $1/n$ and $t(v) = 1$ for every vertex v, then any online algorithm can match the center of the star with probability at most $1/n$, while there exists an edge incident on the center with probability $1 - 1/e$. In contrast, our results are against the strongest adversary, i.e., the optimal offline omniscient algorithm. Clearly the performance of the optimal online algorithm can be no better than that of such an omniscient algorithm.

In their model [6], Chen et al. presented a 1/4 competitive algorithm. Their results were later extended to the weighted setting by Bansal et al. [3] who used a linear program to bound the performance of the optimal algorithm and gave a 1/4 competitive algorithm for the general case, and a 1/3 competitive algorithm for the special case of bipartite graphs. It is interesting to note that the linear program considered by Bansal et al. has an integrality gap of 2 for general graphs which limits the best factor attainable by LP based algorithms. Another interesting aspect of their model is that the optimal online algorithm is a stochastic dynamic program having exponentially many states, and it is not even known if the problem is NP-hard when the patience parameters $t(.)$, can be arbitrary.

Online Bipartite Matching Problems: Online bipartite matching was first introduced by Karp et al. in [16]. Here one side of a bipartite graph is known in advance and the other side arrives online. For each arriving vertex we are revealed its neighbors in the given side. The task is to match the maximum number arriving vertices. In [16], the authors gave a tight $e/(e-1)$ factor algorithm for this problem. This barrier of $1 - 1/e$ has been breached for various stochastic variants of this problem [11, 4, 18], by assuming prior stochastic knowledge about the arriving vertices. Goel and Mehta [13] studied the random order arrival model where the vertices of the streaming side are presented in a random order and showed that the greedy algorithm attains a factor of $1 - 1/e$. This was later improved to a 0.69 competitive algorithm by [17, 15].

Remark 1. The algorithm in [17, 15] can be thought of as the following randomized algorithm for finding a large matching in a given bipartite graph - randomly permute one of the sides and consider the vertices of the other side also in a (uniformly) random order. Match every vertex to the first available neighbor (according to the permutation) on the other side. It can be viewed as an oblivious algorithm that ignores the edge structure of the graph and can therefore be simulated in our setting. This yields a 0.69 competitive algorithm for the SMCP restricted to bipartite graphs.

Randomized Algorithms for Maximum Matchings: Fast randomized algorithms for finding maximum matchings have been studied particularly in the context of Erdős-Rényi graphs [2, 12, 5] starting from the work of Karp and Sipser [23]. However all these algorithms explicitly exploit the edge structure of the graph and are not applicable in our setting. In [1], Aronson et al. analysed the following simple algorithm for finding a matching in a general graph - consider the vertices of the graph in a random order and match each vertex to a randomly chosen neighbor that is unmatched. This algorithm was shown to achieve a factor of 0.50000025.

Remark 2. To the best of our knowledge, the algorithm in [1] is the only non-trivial algorithm for finding a large matching in a general graph that works without looking at the edge structure. Since this algorithm works for *arbitrary* graphs, it can be simulated in our setting and yields a 0.50000025 factor algorithm for SMCP for general graphs. However we manage to improve the factor by exploiting the additional stochastic information available to us in our model.

1.3 Informal Description of the Proof Technique

Observe that the simple algorithm, which weighs (or probes) an edge e according to probability p_e, is not necessarily the best way to proceed. Consider a path having 3 edges such that the middle edge is present with probability 1 whereas the other two edges are each present with probability 0.9. Even though the middle edge is always present, it is unlikely to be involved in any maximum matching. Conversely, the outer edges will always be a part of some maximum matching when they appear.

To determine the relative importance of edges, our algorithm relies on offline simulations. We sample from the given distribution to obtain a collection of representative graphs. We use maximum matchings from these graphs to estimate the probability (denoted by q_e^*) that a given candidate edge e belongs to the maximum matching. Note that this is done as a preprocessing step *without* probing any of the edges in the given graph (a necessary requirement, as probing an edge could lead to unwanted commitments). Clearly the probability that a vertex would get matched in the optimal solution is the sum of q_e^* for all edges incident on it and this gives us a way to approximate the optimal solution.

Similarly we can also calculate the conditional probability that an edge belongs to the maximum matching, given that it is present in the underlying graph. We use this as a measure of the importance of the edge. Observe that it is safe to probe edges where this conditional probability is large, since we are unlikely to make a mistake on such edges. After we are done probing these edges we are left with a residual graph where this conditional probability is small for every edge.

Ideally at this point what we would like to do is to simulate the fractional matching given by the q^*_e, i.e., include every edge with probability q^*_e. However, this is made impossible by the combination of our lack of knowledge of the graph and the commitments we are forced to make as we scan edges to obtain information about the graph. To overcome these limitations, we devise a novel sampling technique, described in section 2.3, that gives us a partial simulation. This sampling algorithm outputs a (randomized) ordering to scan the edges incident to a given vertex, so as to ensure that edge e is included with probability at least some large positive fraction of q^*_e.

2 Preliminaries

2.1 The Model

We are given a set of vertices V, and for every unordered pair of vertices $u, v \in V$, we have a (known) probability p_{uv} of the edge (u, v) being present. These probabilities are independent over the edges. Let $\mathcal{D}$ denote this distribution over all graphs defined by p. Let $G(V, E)$ be a graph drawn from $\mathcal{D}$. We are given only the vertex set V of G, but the edge-set E is not revealed to us unless we *scan* an unordered pair of vertices. A pair $(u, v) \in V \times V$ may be scanned to check if they are adjacent and if so then they are matched and removed from the graph. The objective is to maximize the expected number of vertices that get matched.

We compare our performance to the optimal off-line algorithm that knows the edges before hand, and reports the maximum matching in the underlying graph. We say an online algorithm $\mathcal{A}$ attains a competitive ratio of γ for the SMCP if, for every problem instance $\mathcal{I} = (G(V, .), p)$, the expected size of the matching returned by $\mathcal{A}$ is at least γ times the expected size of the optimal matching in the Erdős Rényi graph $G(V, p)$. That is, $\gamma = \min_{\mathcal{I}=(G(V,.),p)} \left\{ \frac{E\left[\mathcal{A}(\mathcal{I})\right]}{E\left[\text{max matching in } G(V,p)\right]} \right\}$.

2.2 Definitions

For any graph H drawn from $\mathcal{D}$, let $M(H)$ be an arbitrarily chosen maximum matching on H. We define

$$q^*_{uv} = \Pr_{H \leftarrow \mathcal{D}} \left(u \sim v \text{ in } M(H)\right).$$

Clearly $q^*_{uv} \le p_{uv}$, since an edge cannot be part of a maximal matching unless it is actually in the graph. In general, the ratio q^*_{uv}/p_{uv} can be thought of as the conditional probability that an edge is in the matching, given that it appears in the graph. For a given vertex u, the probability that u is matched in M is exactly

$$Q_u(G) := \sum_v q^*_{uv}.$$

This of course is at most 1. We will compare the performance of our algorithm against the expected size of a maximum matching (denoted by OPT) for a graph drawn from $\mathcal{D}$. Thus we have,

$$\mathbf{E}[|\text{OPT}|] = \mathbf{E}\left[|M(H)|\right] = \frac{1}{2}\sum_u Q_u(G) = \sum_{(u,v)} q^*_{uv}, \tag{1}$$

where the last sum is taken over unordered pairs. Finally define an unordered pair (u, v) to be a *candidate edge* if both u and v are still unmatched and (u, v) is yet to be scanned. At any stage let $F(G) \subseteq V \times V$ be the set of candidate edges, and for any $u \in V$, let $N(u, G) \subseteq F(G)$ denote the candidate edges incident on u. A vertex u is defined to be *alive* if $|N(u, G)| > 0$.

2.3 Sampling Technique

In this section we will describe a sampling technique that will be an important component of our algorithm. A curious reader may directly read Corollary 1 and proceed to Section 3 to see an application of this technique. Frequently over the course of our algorithm we will encounter the following framework: We have a vertex v, whose incident edges have known probabilities p_{uv} of being connected to v. We would like to choose an ordering on the incident edges to probe accordingly so that each edge is included(scanned and found to be present) with some target probability of at least r_{uv} (which may depend on u).

Clearly there are some restrictions on the r_{uv} in order for this to be feasible; for example the situation is clearly hopeless if $r_{uv} > p_{uv}$. More generally, for each subset S of the neighborhood of v, it must be the case that the sum of the target probabilities of vertices in S (the desired probability of choosing some member of S) is at most the probability that at least one vertex of S is adjacent to v. As it turns out, these are the *only* necessary restrictions.

Lemma 1. *Let $A_1, A_2, \ldots A_k$ be independent events having probabilities $p_1, \ldots, p_k$. Let $r_1, \ldots, r_k$ be fixed non-negative constants such that for every $S \subseteq \{1, \ldots, k\}$ we have*

$$\sum_{i \in S} r_i \leq 1 - \prod_{i \in S}(1 - p_i). \tag{2}$$

Then there is a probability distribution over permutations π of $\{1, 2, \ldots, k\}$ such that for each i, we have

$$\mathbf{P}(A_i \textit{ is the earliest occurring event in } \pi) \geq r_i\,. \tag{3}$$

Proof. By the Theorem of the Alternative from Linear Duality [10], it suffices to show that the following system of $n! + 1$ inequalities in $n + 1$ variables $\{x_1, \dots, x_n, y\}$ does not have a non-negative solution:

$$y - \sum_k x_k r_k < 0 \tag{4}$$

$$\forall \pi \in S_n, \qquad y - \sum_k x_k p_k \prod_{\substack{j<k \\ \text{in}\,\pi}} (1 - p_j) \geq 0 \tag{5}$$

Assume such a solution exists. Without loss of generality we may assume $x_1 \geq x_2 \geq \cdots \geq x_n \geq 0$. Combining the first inequality with the inequality from the identity permutation, we have

$$\sum_{i=1}^{n} x_i p_i \prod_{j=1}^{i-1} (1 - p_j) < \sum_{k=1}^{n} x_k r_k. \tag{6}$$

On the other hand, we have for each k by applying (2) to $S = \{1, 2, \dots k\}$ that

$$\sum_{i=1}^{k} r_i \leq 1 - \prod_{j=1}^{k} (1 - p_j).$$

By weighting each of these equations by $(x_i - x_{i+1})$ and treating $x_{n+1} = 0$ (note that each of these weights are nonnegative by assumption) and adding, we obtain

$$\sum_{k=1}^{n} x_k r_k \leq \sum_{k=1}^{n} (x_k - x_{k+1})[1 - \prod_{j=1}^{k} (1 - p_j)]. \tag{7}$$

It can be checked directly that both the left side of (6) and the right hand side of (7) are equal to

$$\sum_{\substack{S \subseteq \{1,2,\dots n\} \\ S \neq \emptyset}} (-1)^{|S|-1} x_{\max(S)} \prod_{i \in S} p_i,$$

implying that the two equations contradict each other. Therefore no such solution to the dual system can exist, so the original system must have been feasible.

In theory, it is possible to find the desired distribution π using linear programming. However, it turns out there is a faster constructive combinatorial algorithm:

Lemma 2. *A probability distribution π on permutations solving the program* (3) *can be constructively found in time $O(n^2)$.*

We defer the proof of this lemma and a description of the relevant algorithm to Appendix B. The following corollary follows immediately from Lemma 2.

Corollary 1. *Given a graph $G(V, E)$ and $u \in V$, such that $q^*_e/p_e < \alpha < 1$ for every $e \in N(u, G)$, there exists a randomized algorithm for scanning the edges in $N(u, G)$ in such a way that the probability of including any edge $e \in N(u, G)$ in the matching is at least $\delta(u, G)q^*_e$, where*

$$\delta(u, G) = \frac{1 - \exp(-\sum_{v \in N(u,G)} q^*_{uv}/\alpha)}{\sum_{v \in N(u,G)} q^*_{uv}} .$$

Proof. Note that for any $u \in V$, and $S \subseteq N(u, G)$, $1 - \prod_{v \in S}(1 - p_{uv}) \geq \sum_{v \in S} q^*_{uv}$, since the right side represents the probability q is matched to S in our chosen maximal matching and the left side the probability that there is at least one edge connecting q to S. Thus (p, q^*) satisfy the condition for Lemma 1. However, we can do better. For any given S, if we scale each q_e by $\left(1 - \prod_{v \in S}(1 - p_{uv})\right)/\sum_{v \in S} q^*_{uv}$, the above condition still remains satisfied for that S. Since $q^*_e/p_e < \alpha$ we have

$$\frac{1 - \prod_{v \in S}(1 - p_{uv})}{\sum_{v \in S} q^*_{uv}} \geq \frac{1 - \exp(-\sum_{v \in S} p_{uv})}{\sum_{v \in S} q^*_{uv}} \geq \frac{1 - \exp(-\sum_{v \in S} q^*_{uv}/\alpha)}{\sum_{v \in S} q^*_{uv}} \tag{8a}$$

$$\geq \frac{1 - \exp(-\sum_{v \in N(u,G)} q^*_{uv}/\alpha)}{\sum_{v \in N(u,G)} q^*_{uv}} = \delta(u, G) , \tag{8b}$$

and (8b) follows since $1 - \exp(-\sum_{v \in S} q^*_{uv}/\alpha)/\sum_{v \in S} q^*_{uv}$ is a decreasing function in $\sum_{v \in S} q^*_{uv}$, thus achieving its minimum value at $S = N(u, G)$. Therefore we can replace our q^* by $\delta(u, G)q^*$ and still have the conditions of Lemma 1 hold.

3 Matching Algorithm on Unweighted Erdős-Rényi graphs

Our algorithm can be divided into two stages. The first stage involves several iterations each consisting of two steps - Estimation and Pruning. The parameters α and C will be determined in Section 4.

- **Step 1 (Estimation)**: Generate samples $H_1, H_2, \ldots H_C$ of the Erdős-Rényi graph by sampling from $\mathcal{D}$. For each sample, generate the corresponding maximum matching $M(H_j)$. For every prospective edge (u, v), let q_{uv} be the proportion of samples in which the edge (u, v) is contained in $M(H_j)$.
- **Step 2 (Pruning)**: Let (u, v) be an edge having maximum (finite) ratio q_{uv}/p_{uv}. If this ratio is less than α, end Stage 1. Otherwise, scan (u, v). If (u, v) is present, add it to the partial matching; remove u and v from V, and return to Step 1; otherwise set p_{uv} to 0 and return to Step 1.

We recompute q_{uv} every time we scan an edge. Stage 1 ends when the maximum (finite) value of q_e/p_e falls below α. Note that at this point we stop recomputing q, and these values of q will remain the same for each pair of vertices for the remainder of the algorithm. We now describe the second stage of the algorithm.

The second stage also has several iterations each consisting of two steps. At the start of this stage define $X = V$.

- **Step 1 (Random Bipartition)**: Randomly partition X into two equal sized sets L and R and let B be the *bipartite graph* induced by L and R.
- **Step 2 (Sample and Match)**: Iterate through the vertices in L in an arbitrary order, and for each vertex $u \in L$ sample a neighbor in $N(u, B)$ by choosing a vertex in R using the sampling technique described in Corollary 1[3]. At the end redefine X to be the set of alive vertices in R and discard the unmatched vertices in L. Recall that a vertex was defined to be alive if it is still unmatched and it has at least one candidate edge incident on it.

The algorithm terminates when X becomes empty.

4 Analysis

In this section we will analyze the competitive ratio for the algorithm described earlier. We begin by analyzing Stage 1 of the algorithm. For each iteration in Stage 1, define the residual graph at the start of the i^{th} iteration to be G_i starting with $G_1 = G$. We denote by $q^*_{e,i}$ the actual probability that e is contained in the maximal matching on G_i and $q_{e,i}$ as our estimate calculated in Step 1. We define

$$\epsilon_e := \max_i |q_{e,i} - q^*_{e,i}| \, .$$

Let the total number of iterations in this stage be k and let $G' = G_k$. Let ALG_1 be the set of edges that are matched in Stage 1 and let $OPT(G_i)$ be the optimal solution in the residual graph at the start of the i^{th} iteration. We have the following lemma.

Lemma 3.

$$\mathbf{E}\left[|OPT| - |OPT(G')|\right] \leq (2-\alpha)\mathbf{E}[|ALG_1|] + \sum_e \epsilon_e \, .$$

Proof. For $i \in [k]$, let $Gain(i)$ be 1 if the edge scanned in the i^{th} iteration is present, and 0 otherwise. We will first show that $E\left[|OPT(G_i)| - |OPT(G_{i+1})|\right] \leq (2-\alpha)\mathbf{E}[Gain(i)]$. Three cases may arise during the i^{th} iteration.

[3] The algorithm described in Corollary 1 requires the exact estimates for q^*_e. However we will show in our analysis that for large enough samples C, q_e defined above is a good estimate of q^*_e.

- Case 1: The edge scanned in the i^{th} iteration is not present. Then $OPT(G_i) = OPT(G_{i+1})$ and $Gain(i) = 0$ thus, $|OPT(G_i)| - |OPT(G_{i+1})| = Gain(i) = 0$.
- Case 2.1: The edge scanned in the i^{th} iteration is present but does not belong to $OPT(G_{i+1})$. This happens with probability $p_e - q^*_{e,i}$. Then $|OPT(G_i)| - |OPT(G_{i+1})| = 2$ and $Gain(i) = 1$.
- Case 2.2: The edge scanned in the i^{th} iteration is present and belongs to $OPT(G_i)$. This happens with probability $q^*_{e,i}$. Then $|OPT(G_i)| - |OPT(G_{i+1})| = 1$ and $Gain(i) = 1$.

Summing over all three cases, we see that

$$\mathbf{E}[|OPT(G_i)| - |OPT(G_{i+1})|] = 2(p_e - q^*_{e,i}) + q^*_{e,i} \leq p_e(2 - \alpha) + \epsilon_e \ ,$$

while the expected gain from scanning the edge is simply p_e. The result follows from adding over all scanned edges, and noting for the additive factor that each edge is scanned at most once in the first stage (and indeed in the whole algorithm).

Analysis of Stage 2: Let us begin by analyzing the first iteration of the second stage of the algorithm. The analysis for the subsequent iterations would follow along similar lines. Let G' be the residual graph at the start of the second stage, where $q_e/p_e < \alpha$ for every candidate edge e, and $1/2 \sum_u Q_u(G') = \mathbf{E}[|OPT(G')|]$. The following lemma bounds the performance of the first iteration of Stage 2 on G'. We defer the proof to Appendix C.

Lemma 4. *The expected number of edges that are matched in the first iteration of Stage* 2 *of the algorithm is at least* $\left(1 - \frac{1}{e}\right)\left(1 - e^{-1/2\alpha}\right)|OPT(G')| - \sum_e \epsilon_e$.

For ease of notation, let $\phi = (1 - 1/e)\left(1 - e^{-1/2\alpha}\right)$. Let ALG_2 be the set of edges that get matched in Stage 2 of the algorithm. The following lemma lower bounds $\mathbf{E}[|ALG_2|]$. Refer to Appendix D for the complete proof.

Lemma 5.

$$\mathbf{E}[|ALG_2|] \geq \mathbf{E}[|OPT(G')|]\left[\frac{\phi}{1 - (\frac{1-\phi}{2})^2}\right] - \sum_e \epsilon_e \ .$$

Now all that is left is to balance the factors for both the stages and set the optimal value of α. In the subsequent theorem we find the optimal value of α.

Theorem 3. *The above algorithm attains a factor of at least* $0.573 - 2\gamma$ *where* $\sum_e \epsilon_e \leq \gamma\mathbf{E}(|OPT|)$.

Proof. By Lemma 3, $\mathbf{E}[|OPT| - |OPT(G')|] \leq 2/(1+\alpha)\mathbf{E}[|ALG_1|] + \sum \epsilon_e$. Also by Lemma 5, $\mathbf{E}[|OPT(G')|] \leq \frac{1-(\frac{1-\phi}{2})^2}{\phi}\mathbf{E}[|ALG_2|] + \sum \epsilon_e$. Combining these two and substituting $\alpha = 0.255$ and $\phi = (1-1/e)\left(1-e^{-1/2\alpha}\right) = 0.543$ we have,

$$\mathbf{E}[|OPT|] \leq (2-\alpha)\mathbf{E}[|ALG_1|] + \frac{1-(\frac{1-\phi}{2})^2}{\phi}\mathbf{E}[|ALG_2|] + 2\sum_e \epsilon_e \tag{9a}$$

$$= 1.74(\mathbf{E}[|ALG_1|] + \mathbf{E}[|ALG_2|]) + 2\sum_e \epsilon_e \tag{9b}$$

$$= 1.74\mathbf{E}[|ALG|] + 2\sum_e \epsilon_e \tag{9c}$$

Thus $\mathbf{E}[|ALG|] \ \geq 0.573 \ \mathbf{E}[|OPT|]$.

4.1 Running Time Analysis

In this section we will analyze the running time of our algorithm and determine the optimal value of parameter C. We will use n to denote the number of vertices and m to denote the number of edges that have a non-zero probability of being present.

By Lemma 2 it takes $O(n^2)$ time to probe the neighborhood of a given vertex, thus the second stage can be implemented in $O(n^3)$ time. Analysis of the first stage is slightly involved, since in this stage we wish to approximate $q^*_{e,i}$ by repeated sampling. The following lemma sets the optimal value of C for which the total error caused by approximating $q^*_{e,i}$ by sampling is small. The proof can be found in Appendix E.

Lemma 6. *For $C = n\log^6(n)$ samples in Step 1, $\sum_e \epsilon_e$ is $o(n)$ with high probability.*

A naive implementation of the algorithm presented in section 3 would require recalculating q_e after every iteration in stage 1. This can be quite time consuming since even the fastest implementation [20] of the maximum matching algorithm takes $O(m\sqrt{n})$ time. In the following lemma, we explain how to circumvent this bottleneck. The details can be found in Appendix F. Finally, using Lemma 6 and 7 our algorithm can be implemented in $\tilde{O}(n^4)$ time.

Lemma 7. *Stage 1 of the algorithm can be implemented in $\tilde{O}(n^2C)$ time.*

On the hardness front, we prove the following theorem.

Theorem 4. *No randomized algorithm can attain a competitive ratio better than* 0.896 *for the SMCP.*

The proof of this theorem, which relies on a full analysis of the performance of the optimal online theorem on the Erdos-Renyi graph G(0,0.64), is deferred to Appendix G.

References

1. Aronson, J., Dyer, M., Frieze, A., Suen, S.: Randomized greedy matching. ii. Random Struct. Algorithms **6** (January 1995) 55–73
2. Aronson, J., Frieze, A., Pittel, B.G.: Maximum matchings in sparse random graphs: Karp-sipser revisited. Random Struct. Algorithms **12** (March 1998) 111–177
3. Bansal, N., Gupta, A., Li, J., Mestre, J., Nagarajan, V., Rudra, A.: When lp is the cure for your matching woes: improved bounds for stochastic matchings. In: Proceedings of the 18th annual European conference on Algorithms: Part II. ESA'10, Berlin, Heidelberg, Springer-Verlag (2010) 218–229
4. Birnbaum, B., Mathieu, C.: On-line bipartite matching made simple. SIGACT News **39** (March 2008) 80–87
5. Chebolu, P., Frieze, A., Melsted, P.: Finding a maximum matching in a sparse random graph in o(n) expected time. J. ACM **57** (May 2010) 24:1–24:27
6. Chen, N., Immorlica, N., Karlin, A.R., Mahdian, M., Rudra, A.: Approximating matches made in heaven. In: Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part I. ICALP '09, Berlin, Heidelberg, Springer-Verlag (2009) 266–278
7. Dean, B.C., Goemans, M.X., Vondrák, J.: Adaptivity and approximation for stochastic packing problems. In: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms. SODA '05, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics (2005) 395–404
8. Duan, R., Pettie, S.: Approximating maximum weight matching in near-linear time. In: Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science. FOCS '10, Washington, DC, USA, IEEE Computer Society (2010) 673–682
9. Edmonds, J.: Paths, trees, and flowers. Canadian Journal of Mathematics **17** (1965) 449–467
10. Farkas, J.G.: Uber die theorie der einfachen ungleichungen. Journal fur die Reine und Angewandte Mathematik **124** (1902) 1–27
11. Feldman, J., Mehta, A., Mirrokni, V.S., Muthukrishnan, S.: Online stochastic matching: Beating 1-1/e. In: FOCS. (2009) 117–126
12. Frieze, A., Pittel, B.: Perfect matchings in random graphs with prescribed minimal degree. In: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms. SODA '03, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics (2003) 148–157
13. Goel, G., Mehta, A.: Online budgeted matching in random input models with applications to adwords. In: SODA. (2008) 982–991
14. Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association **58** (March 1963) 13–30
15. Karande, C., Mehta, A., Tripathi, P.: Online bipartite matching in the unknown distributional model. In: STOC. (2011) 106–117
16. Karp, R., Vazirani, U., Vazirani, V.: An optimal algorithm for online bipartite matching. In: Proceedings of the 22nd Annual ACM Symposium on Theory of Computing. (1990)
17. Mahdian, M., Yan, Q.: Online bipartite matching with random arrivals: An approach based on strongly factor-revealing lps. In: STOC. (2011) 117–126
18. Manshadi, V.H., Oveis-Gharan, S., Saberi, A.: Online stochastic matching: Online actions based on offline statistics. In: SODA. (2011)
19. Mehta, A., Mirrokni, V.: Online ad serving : Theory and practice. Tutorial (2011)
20. Micali, S., Vazirani, V.V.: An O($E\sqrt{V}$) algoithm for finding maximum matching in general graphs. In: Foundations of Computer Science, 1980., 21st Annual Symposium on. (oct. 1980) 17 –27
21. Nikolova, E., Kelner, J.A., Brand, M., Mitzenmacher, M.: Stochastic shortest paths via quasi-convex maximization. In: Proceedings of the 14th conference on Annual European Symposium - Volume 14, London, UK, Springer-Verlag (2006) 552–563
22. Papadimitriou, C.H., Yannakakis, M.: Shortest paths without a map. Theor. Comput. Sci. **84** (July 1991) 127–150
23. R.M.Karp, M.Sipser: Maximum matching in sparse random graphs. In: FOCS. (1981) 364–375
24. Ross, L., Rubin, D., Siegler, M., Josephson, M., Thistlethwaite, J., E.S.Woodle: The case for a living emotionally related international kidney donor exchange registry. Transplantation Proceedings **18** (1986) 5–9
25. Ross, L., Rubin, D., Siegler, M., Josephson, M., Thistlethwaite, J., E.S.Woodle: Ethics of a paired-kidney-exchange program. Volume 336. (1997) 1752–1755
26. Shmoys, D.B., Swamy, C.: An approximation scheme for stochastic linear programming and its application to stochastic integer programs. J. ACM **53** (November 2006) 978–1012
27. Tao, T., Vu, V.: Additive Combinatorics. Cambridge University Press (2006)

A Applications of SMCP

Apart from its theoretical importance, SMCP models several important real-world scenarios. We briefly highlight some of these in this section.

Kidney Exchange: Consider a scenario where there are two incompatible donor/patient pairs where each donor is willing to donate a kidney for the patient, however is incompatible with her patient. In this setting we can perform a kidney exchange in which two incompatible patient/donor pairs are identified such that each donor is compatible with the other pair's patient. Four simultaneous operations are then performed, exchanging the kidneys between the pairs in order to have two successful (compatible) transplants. Two donor/patient pairs can be tested to check if such an exchange is possible. Owing to the cost involved in testing and due to ethical concerns, it is desired that an exchange is performed whenever the test indicates that it is possible. We wish to maximize the number of such kidney exchanges.

This problem can easily be modeled as an instance of SMCP, where each donor and patient pair represents a node of the graph and the edges indicate possible pairs along which this exchange is possible. Furthermore using external data, we can derive prior information about the likelihood of an exchange. We refer the reader to [24, 25] for more details.

Online Advertising: In display advertising, a user is shown ads during a browsing session. Whenever a user clicks on an ad, she is redirected to the merchant's website. The advertiser wishes to maximize the number of ads that get clicked across all users. This can be modeled as an instance of SMCP where the users and advertisers are the two sides of a bipartite graph. Whenever a user is shown an ad it is analogous to scanning the edge connecting them. Here the probability of an edge being present (an ad being clicked by a user) can be determined based on user profiles and is known to the advertiser in advance.

B Combinatorial Algorithm for Implementing the Sampling Technique

In this section we will give a complete proof of Lemma 2.

By Lemma 1, we know that our constraint set is equivalent to

$$\sum_{i \in S} r_i + \prod_{i \in S} (1 - p_i) \leq 1 \tag{10}$$

holding for every S. Suppose S^* is a non-empty set for which the left hand side of (10) is maximized, and there is some $j_1 \in S$ and $j_2 \notin S$. We would then have

$$\sum_{i \in S^*} r_i + \prod_{i \in S^*} (1-p_i) \geq \sum_{i \in S^* \setminus j_1} r_i + \prod_{i \in S^* \setminus j_1} (1-p_i)$$
$$\sum_{i \in S^*} r_i + \prod_{i \in S^*} (1-p_i) \geq \sum_{i \in S^* \cup \{j_2\}} r_i + \prod_{i \in S^* \cup \{j_2\}} (1-p_i)$$

Rearranging both of the above inequalities yields

$$\prod_{i \in S^*} (1-p_i) \leq \frac{r_{j_1}}{\frac{1}{1-p_{j_1}} - 1}$$
$$\prod_{i \in S*} (1-p_i) \geq \frac{r_{j_2}}{p_{j_2}}.$$

Comparing these two inequalities, we have

$$\begin{aligned} \frac{r_{j_2}}{p_{j_2}} &\leq \frac{r_{j_1}}{\frac{1}{1-p_{j_1}} - 1} \\ &= \frac{r_{j_1}(1-p_{j_1})}{p_{j_1}} \\ &\leq \frac{r_{j_1}}{p_{j_1}}. \end{aligned}$$

Assume without loss of generality that the events are sorted in decreasing order by the ratio r/p. By the above, we have proven

Claim. The left hand side of (10) is always maximized by $S = \{1, 2, \ldots, k\}$ for some k.

We also note the following:

Claim. If (10) is satisfied for all S and tight for some S_0, then there is a solution to the program where any permutation having nonzero weight considers all variables in S_0 before any variable outside S_0.

This is simply because the left hand side of (10) measures the sum of the probability that an event in S_0 is first and the probability that no event in S_0 occurs. If the sum is 1, then an event in S_0 must always be first if one is present, and that would not be possible if some other event could appear before it in π.

We now present our algorithm.

Step 0: (pre-processing:) Compute the largest y for which the weights (yr_i, p_i) still satisfy (10). If $y < 1$, the constraints are infeasible. If $y > 1$, replace all r_i by yr_i, at which point (10) is tight for at least one S.

This preprocessing step is only performed once.

Step 1: (slack removal) If there is no $k < n$ for which (10) is tight for $S = \{1, 2, \dots k\}$, compute the largest $z \leq 1$ such that the program remains feasible when for all j we replace r_j by

$$r'_j := \frac{r_j - zp_j \prod_{i>j}(1-p_i)}{1-z}.$$

With probability z, consider the edges in decreasing order of index. Otherwise, consider edges according to a distribution found by solving the problem with r_j replaced by r'_j. If $z = 1$, we are finished.

Step 2: (divide-and-conquer) Find a $k < n$ for which (10) is tight for $S = \{1, 2, \dots, k\}$. Solve, without pre-processing, the problems on S (with the original target distribution) and S^C (replacing the targets in S^C by $r'_j := r_j / \prod_{i=1}^{k}(1 - p_i)$). Form the distribution by independently sampling from the distributions on S and S^C found by solving the subproblems, and consider all variables in S before the variables in S^C.

For step 0, note that multiplication by y does not change the relative ordering of r_i/p_i. This implies that the y in question is the smallest y for which one of the sets $\{1, 2, \dots, k\}$ makes (10) tight. We can examine all such sets and find the y in question in linear time by updating $\sum r_i$ and $\prod(1-p_i)$ each time k increases to $k+1$.

For step 1, note that

$$\begin{aligned}\frac{r'_i}{p_i} - \frac{r'_{i+1}}{p_{i+1}} &= \frac{r_i - zp_i \prod_{j>i}(1-p_j)}{p_i} - \frac{r_i - zp_{i+1} \prod_{j>i+1}(1-p_j)}{p_{i+1}} \\ &= \left(\frac{r_i}{p_i} - \frac{r_{i+1}}{p_{i+1}}\right) + zp_i \prod_{j>i+1}(1-p_j) \\ &\geq 0.\end{aligned}$$

In other words, replacing r_j by r'_j again never alters the relative ordering of the ratios. So again we only need consider n sets to determine z, and can do this in linear time.

The r'_j were chosen such that achieving a target of r'_j with probability $(1-z)$ corresponds to achieving a target of r_j in the original problem, so it is enough to solve this new problem. The only claim that remains to be checked is that we can actually find the desired k in Step 2. Since we know by our first claim that the left hand side of (10) is always maximized for some $S = \{1, 2, \dots, k\}$, it suffices to show that we can take $k < n$.

But for $S = \{1, \dots, n\}$, the left hand side of (10) is

$$\sum_{j=1}^{n} r'_j + \prod_{j=1}^{n}(1-p_j) = \frac{\sum_{j=1}^{n} r_j - z\sum_{j=1}^{n} p_j \prod_{i>j}(1-p_i)}{1-z} + \prod_{j=1}^{n}(1-p_j)$$

$$
\begin{aligned}
&= \frac{\sum_{j=1}^n r_j - z\left(1 - \prod_{j=1}^n (1-p_j)\right)}{1-z} + \prod_{j=1}^n (1-p_j) \\
&= \frac{\sum_{j=1}^n r_j + \prod_{j=1}^n (1-p_j) - z}{1-z} \\
&\le 1,
\end{aligned}
$$

where the last inequality comes from our assumption that (10) holds for $S = \{1, 2, \ldots, n\}$. Intuitively, this corresponds to how imposing constraints on the order in which the events are placed (increasing z) has no effect on the probability at least one event occurs (the left hand side of (10) in the case where S is everything). So at the maximal z (assuming $z < 1$), some other constraint must also be tight, which gives us our k.

Since step 1 takes at most linear time, it follows that the whole algorithm takes at most quadratic time.

C Proof of Lemma 4

Proof. Let us define an indicator random variable Y_u for every $u \in V$ that is 1 if $u \in R$ and 0 otherwise and $\Pr[Y_u = 1] = \Pr[Y_u = 0] = 1/2$. Thus $\mathbf{E}[Y_u] = 1/2$. We will use $u \sim v$ to denote that u and v get matched. The expected number of vertices in R that will get matched in the first iteration is given by

$$
\begin{aligned}
&\mathbf{E}\left[\text{matched vertices in } \; R\right] && \text{(11a)}\\
&= \mathbf{E}_Y\left[\sum_u Y_u \left\{1 - \prod_{v \neq u} \left(1 - Pr\left[u \sim v, \; v \in L \mid u \in R\right]\right)\right\}\right] && \text{(11b)}\\
&= \mathbf{E}_Y\left[\sum_u Y_u \left\{1 - \prod_{v \neq u} \left(1 - Pr\left[u \sim v \mid u \in R, \; v \in L\right](1 - Y_v)\right)\right\}\right] && \text{(11c)}\\
&\ge \mathbf{E}_Y\left[\sum_u Y_u \left\{1 - \prod_{v \neq u} \left(1 - q_{uv}\delta(v, B \mid u \in R, v \in L)(1 - Y_v)\right)\right\}\right] && \text{(11d)}\\
&\ge \mathbf{E}_Y\left[\sum_u Y_u \left\{1 - \prod_{v \neq u} \left(1 - q_{uv}(1 - Y_v)\frac{1 - (1 - q_{uv}/\alpha) \prod_{w \neq u \neq v} (1 - q_{vw}Y_w/\alpha)}{\sum q_{vw}Y_w + q_{uv}}\right)\right\}\right] && \text{(11e)}
\end{aligned}
$$

(11c) follows from conditional probability, and (11d) and (11e) follow from Corollary 1, concerning our sampling technique. Since each of the random variables Y_u are chosen independently, (11e) can be simplified as below.

$$\mathbf{E}\left[\text{matched vertices in } R\right] \tag{12a}$$

$$\geq \sum_u \mathbf{E}_Y\left[Y_u\right]\mathbf{E}_Y\left[1-\prod_{v\neq u}\left(1-q_{uv}(1-Y_v)\frac{1-(1-q_{uv}/\alpha)\prod_{w\neq u\neq v}(1-q_{vw}Y_w/\alpha)}{\sum_{w\neq u\neq v} q_{vw}Y_w+q_{uv}}\right)\right] \tag{12b}$$

$$= \frac{1}{2}\sum_u \mathbf{E}_Y\left[1-\prod_{v\neq u}\left(1-q_{uv}(1-Y_v)\frac{1-(1-q_{uv}/\alpha)\prod_{w\neq u\neq v}(1-q_{vw}Y_w/\alpha)}{\sum_{w\neq u\neq v} q_{vw}Y_w+q_{uv}}\right)\right] \tag{12c}$$

In the following, (13b) can be derived from (12c) by using the identity $1-x < e^{-x}$, and (13c) is obtained by noting that $(1-1/e)x < 1-e^{-x}$ for $x \in [0,1]$.

$$\mathbf{E}\left[\text{matched vertices in } R\right] \tag{13a}$$

$$\geq \frac{1}{2}\sum_u \mathbf{E}_Y\left[1-\exp\left(-\sum_{v\neq u} q_{uv}(1-Y_v)\frac{1-(1-q_{uv}/\alpha)\prod_{w\neq u\neq v}(1-q_{vw}Y_w/\alpha)}{\sum_{w\neq u\neq v} q_{vw}Y_w+q_{uv}}\right)\right] \tag{13b}$$

$$\geq \frac{1}{2}\sum_u \mathbf{E}_Y\left[\left(1-\frac{1}{e}\right)\left(\sum_{v\neq u} q_{uv}(1-Y_v)\frac{1-(1-q_{uv}/\alpha)\prod_{w\neq u\neq v}(1-q_{vw}Y_w/\alpha)}{\sum_{w\neq u\neq v} q_{vw}Y_w+q_{uv}}\right)\right] \tag{13c}$$

$$= \frac{1}{2}\left(1-\frac{1}{e}\right)\sum_u \mathbf{E}_Y\left[\sum_{v\neq u} q_{uv}(1-Y_v)\frac{1-(1-q_{uv}/\alpha)\prod_{w\neq u\neq v}(1-q_{vw}Y_w/\alpha)}{\sum_{w\neq u\neq v} q_{vw}Y_w+q_{uv}}\right] \tag{13d}$$

Next observe that both $1-Y_v$ and $\frac{1-(1-q_{uv}/\alpha)\prod_{w\neq u\neq v}(1-q_{vw}Y_w/\alpha)}{\sum_{w\neq u\neq v} q_{vw}Y_w+q_{uv}}$ are decreasing convex functions in Y, thus their product is also a decreasing convex function. Our next set of simplifications are as follows. Since for any multi-variate convex function f, $\mathbf{E}[f(y)] \geq f(\mathbf{E}[y])$ we can lower bound (13d) by (14b) below. Again (14c) is minimized when each of the q_{vw}'s are equal. Substituting this and simplifying we get (14e). Finally $\frac{1-e^{-Q_v(G')/\alpha}}{Q_v(G')}$ is a decreasing function in $Q_v(G')$ that attains its minimum value when $Q_v(G') = 1$. Putting this value in (14f) gives (14g). Further simplification yields the desired result.

$$\mathbf{E}\left[\text{matched vertices in } \ R\right] \tag{14a}$$

$$\geq \frac{1}{2}\left(1-\frac{1}{e}\right)\sum_{u}\sum_{v\neq u} q_{uv}(1-\mathbf{E}[Y_v])\frac{1-(1-q_{uv}/\alpha)\prod\limits_{w\neq u\neq v}(1-q_{vw}\mathbf{E}[Y_w]/\alpha)}{\sum\limits_{w\neq u\neq v} q_{vw}\mathbf{E}[Y_w]+q_{uv}} \tag{14b}$$

$$\geq \frac{1}{2}\left(1-\frac{1}{e}\right)\sum_{u}\sum_{v\neq u} \frac{q_{uv}}{2}\frac{1-(1-q_{uv}/\alpha)\prod\limits_{w\neq u\neq v}(1-q_{vw}/2\alpha)}{\sum\limits_{w\neq u\neq v} q_{vw}/2+q_{uv}} \tag{14c}$$

$$\approx \frac{1}{2}\left(1-\frac{1}{e}\right)\sum_{u}\sum_{v\neq u} \frac{q_{uv}}{2}\frac{1-\prod\limits_{w\neq v}(1-q_{vw}/2\alpha)}{\sum\limits_{w\neq v} q_{vw}/2} \tag{14d}$$

$$\geq \frac{1}{2}\left(1-\frac{1}{e}\right)\sum_{u}\sum_{v\neq u} q_{uv}\frac{1-\exp\left(-\sum\limits_{w\neq v} q_{vw}/2\alpha\right)}{\sum\limits_{w\neq v} q_{vw}/2} \tag{14e}$$

$$\geq \frac{1}{2}\left(1-\frac{1}{e}\right)\sum_{u}\sum_{v\neq u} q_{uv}\frac{1-\exp\left(-Q_v(G')/2\alpha\right)}{Q_v(G')} \tag{14f}$$

$$\geq \frac{1}{2}\left(1-\frac{1}{e}\right)\sum_{u}\sum_{v\neq u} q_{uv}\left(1-e^{-1/2\alpha}\right) \tag{14g}$$

$$\geq \frac{1}{2}\left(1-\frac{1}{e}\right)\left(1-e^{-1/2\alpha}\right)\sum_{u}\sum_{v\neq u} q_{uv} \tag{14h}$$

$$\geq \frac{1}{2}\left(1-\frac{1}{e}\right)\left(1-e^{-1/2\alpha}\right)\sum_{u} Q_u(G') \tag{14i}$$

$$\geq \left(1-\frac{1}{e}\right)\left(1-e^{-1/2\alpha}\right)|OPT(G')| - \sum_{e}\epsilon_e\,. \tag{14j}$$

D Proof of Lemma 5

Proof. Observe that not all candidate edges in G' have been considered during the first iteration of Stage 2. In particular, candidate edges with both end points in R are yet to be considered. For analyzing the subsequent iterations in Stage 2, we will consider only these candidate edges. Clearly this only lower bounds the performance of the algorithm.

Analyzing (14i), we notice that we have in fact proved something stronger in Lemma 4, i.e., we have shown that every vertex $v \in R$ is chosen with probability at least $\phi\ Q_v(G')$. By slightly altering the algorithm it is easy to ensure that for every $v \in R$, it is chosen with exactly this probability. Thus any vertex in R survives the first iteration with probability $1 - \phi Q_v(G') > 1 - \phi$. Since the partitions L and R are chosen at random, the probability that a vertex is in R and unmatched after the first iteration is at least $\mu = (1-\phi)/2$. Continuing this argument further, the probability that an ordered pair (u, v) is a candidate edge at the start of the i^{th} iteration is the probability that both u and v have always been in R in all previous iterations, and are still unmatched; this probability is at least $\mu^{2(i-1)}$.

Let G'_i be the residual graph at the start of the i^{th} iteration in Stage 2, with $G'_1 = G'$. By the above observation and using linearity of expectation, the expected sum of q_e's on candidate edges in G'_i is lower bounded by $\mu^{2i-2} \sum_{e \in F(G')} q_e$. Appealing to a similar analysis as in (the proof of) Lemma 4, the expected size of the matching returned by the i^{th} iteration in the second stage is at least $\phi\mu^{2i-2} \sum_{e \in F(G')} q_e$. Summing over all iterations in Stage 2 we have,

$$\mathbf{E}[|ALG_2|] \geq \sum_i \phi\mu^{2i-2} \sum_{e \in F(G')} q_e \geq \phi \sum_{e \in F(G')} q_e \sum_{i=1} \mu^{2i-2} \tag{15a}$$

$$= \phi \sum_{e \in F(G')} q_e \frac{1}{1-\mu^2} \quad = \quad \frac{1}{2} \sum_{u \in G'} Q_u(G') \frac{\phi}{1-\mu^2} \tag{15b}$$

$$= |OPT(G')| \frac{\phi}{1-\mu^2} \quad = \quad |OPT(G')| \frac{\phi}{1 - \left(\frac{1-\phi}{2}\right)^2} . \tag{15c}$$

E Proof of Lemma 6

Proof. We will give two separate bounds on the size of ϵ_e. One will hold in the case where $q^*_{e,i}$ is not too small, the other for small $q^*_{e,i}$.

Bound 1: For any given sample, we can think of the event $e \in M(H_j)$ as a Bernoulli trial with success probability $q^*_{e,i}$. By Hoeffding's bound ([14], see

Theorem 1.8 in [27] for the specific formulation used), it follows that for any given edge and sample we have

$$\mathbf{P}(|q_{e,i} - q^*_{e,i}| \geq \beta q^*_{e,i}) \leq \exp(-Cq^*_{e,i}\zeta^2/4)\,.$$

This bound tells us that for any edge such that $Cq^*_{e,i}$ tends to infinity sufficiently quickly, the maximum error coming from such an edge will likely be a tiny fraction of $q^*_{e,i}$. However, it is possible that some q_e could be exponentially small, so we cannot just take C large enough so that all edges fall in this class. We turn to the second bound for the remaining edges.

Bound 2: In this case we focus solely on the upper tail. We know from the union bound that

$$\begin{aligned}\mathbf{P}(q_{e,i} \geq q^*_{e,i} + \kappa) &\leq \binom{C}{Cq^*_{e,i} + \kappa C}(q^*_{e,i})^{Cq^*_{e,i}+\kappa C} \\ &\leq \left(\frac{eq^*_{e,i}}{q^*_{e,i} + \kappa}\right)^{Cq^*_{e,i}+\kappa C},\end{aligned}$$

where the first inequality bounds the probability that the edge participates in at least $C(q^*_{e,i} + \kappa)$ matchings by the expected number of sets of $C(q^*_{e,i} + \kappa)$ matchings in which the edge participates.

Set $q_0 = \log^5 n/C$ where we will use bound 1 for $q^*_{e,i} > q_0$ and bound 2 otherwise. For $q^*_{e,i} > q_0$ using bound 1, for an arbitrarily small constant ζ we have,

$$\mathbf{P}(|q_{e,i} - q^*_{e,i}| \geq \zeta q^*_{e,i}) \leq \exp(-\log^5 n\zeta^2/4) = o(1/n^4)\,.$$

Taking the union bound over all edges and trials and adding, we see

$$\mathbf{P}(\sum_{q^*_{e,i} \geq q_0} \epsilon_e \leq \zeta \sum_e q^*_e) = 1 - o(1)\,.$$

Thus the total error accrued across all iterations is small with high probability.

Now let us set $\kappa = 2q_0$. Applying the upper tail bound 2 above, we have for any $q^*_{e,i} < q_0$ that

$$\begin{aligned}\mathbf{P}(q_{e,i} \geq q^*_{e,i} + \kappa) &\leq \left(\frac{eq^*_{e,i}}{q^*_{e,i} + \kappa}\right)^{Cq^*_{e,i}+\kappa C} \\ &\leq (0.92)^{2\log^5 n}.\end{aligned}$$

The corresponding lower bound $q_{e,i} \geq q^*_{e,i} - \kappa$ follows trivially from the non-negativity of q_e.

Taking the union bound over all samplings and all edges, we have that with high probability the total contribution to the error from this case is at most $n^2\kappa = n^2 \log^5 n/C$. By Theorem 3, it suffices to make this a small fraction of the maximum expected matching. Setting $C = n \log^6 n$ ensures that the total error is $o(n)$ (which suffices in the case where the matching is a constant fraction of all vertices, while setting $C = n^2 \log^6 n$ insures the error is $o(1)$ (which works in general) [4]

F Proof of Lemma 7

Proof. Observe that we are not required to find the maximum matching in Step 1. We can instead work with a matching that is $1 - \zeta$ (ζ is an arbitrary small constant) fraction of maximum matching and lose a small multiplicative factor in our analysis. This can be done in $O(m \log m)$ time using a result by Duan and Pettie et al. [8]. Also note that we can reuse the above matching across multiple iterations in Stage 1. This is because the size of the maximum matching changes by at most 1 across consecutive iterations.

Concretely, we will modify the algorithm to calculate the approximate maximum matching using the algorithm in [8] in $O(m \log m)$ time. Let Λ be the estimate of the size of the maximum matching in the residual graph at any point during Stage 1. We can probe up to $\Lambda\zeta$ edges that have $q_{e,i}/p_e > \alpha$ before recalculating $q_{e,i}$. This will induce only a small constant factor (function of ζ) error in our analysis. Hence we would have at most $O(\log(m))$ iterations in Stage 1 where we would be required to recompute $q_{e,i}$. Therefore we can implement Stage 1 in $\tilde{O}(mC) = \tilde{O}(n^2C)$ time.

G Proof of Theorem 4

Proof. Given any graph along with the probability p_e for every edge the optimal algorithm can be found by writing a stochastic dynamic program. The states of the program are all possible subgraphs of the given graph and for each state we record the solution returned by the optimal algorithm. It is easy to see that such a dynamic program would have exponentially many states and would be quite infeasible to solve for the general problem. However it can be used to find the optimal algorithm for small examples.

We considered the complete graph on 4 vertices where each edge is present with probability $p = 0.64$. In Lemma 8 we evaluate the performance of the optimal

[4] If the expected maximum matching is itself $o(1)$ in size, then it follows from the independence of the edges that any edge which is scanned and found to be present is with high probability the *only* edge in the graph! So any algorithm trivially finds the maximal matching in such a graph.

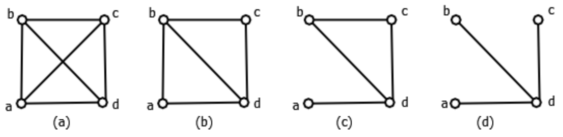

Fig. 1. Intermediate Graphs

online algorithm for this graph. Then in Lemma 9 we calculate the expected size of the maximum matching in this graph.

Lemma 8. *The expected size of the matching found by the optimal online algorithm for SMCP for the Erdős-Rényi graph* $G(4, 0.64)$ *is* 1.607*.*

Proof. Let us consider the complete graph K_4 as shown in Figure 1(a). Without loss of generality we can assume that the optimal algorithm starts by scanning edge ac. If this edge is present, which happens with probability $p = 0.64$, then we are just left with one candidate edge. We can scan this edge next. Thus for this case the optimal algorithm return a matching of expected size $p + p^2$.

Now suppose that ac is not present, then we are left with the graph shown in Figure 1(b). Clearly the optimal algorithm should not probe edge bd since it can potentially lead to a smaller matching. Without loss of generality we may assume that the next edge to be scanned is ab. If this edge is present we are again left with one candidate edge cd that is to be scanned. Hence the expected size of the matching returned is $(1-p)p(1+p)$.

Otherwise we are left with the graph shown in Figure 1(c). One can check that the optimal algorithm must next scan bc or ad. Suppose it scans bc, and finds it to be present then we are again down to a single candidate ad which can be scanned next. In this the expected matching returned is of size $(1-p)^2p(1+p)$.

If bc is absent then the residual graph is shown in Figure 1(d). Clearly for the graph in Figure 1(d) the expected maximum matching is of size $1-(1-p)^3$. The expected size of the matching returned in this case is therefore $(1-p)^3(1-(1-p)^3)$. Computing the expected value across all cases and substituting $p = 0.64$ we find that the expected size of the matching returned by the optimal algorithm is 1.607.

Lemma 9. *The expected size of the maximum matching in the Erdős-Rényi graph* $G(4, 0.64)$ *is* 1.792

Proof. The given graph will not have any edge with probability $(1-p)^6$. It will have a matching of size 1 for the graphs shown in figure 2 and their symmetric rotations.

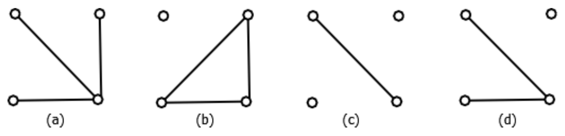

Fig. 2. Graphs with unit size matching

This happens with probability $4p^3(1-p)^3+4p^3(1-p)^3+6p(1-p)^5+12p^2(1-p)^4$. In all other cases the graph will have a matching of size 2 i.e. with probability $1-(1-p)^6-8p^3(1-p)^3-6p(1-p)^5-12p^2(1-p)^4$. Thus the expected size of the maximum matching is $8p^3(1-p)^3+6p(1-p)^5+12p^2(1-p)^4+2(1-(1-p)^6-8p^3(1-p)^3-6p(1-p)^5-12p^2(1-p)^4)$. Substituting $p=0.64$ this is evaluates to 1.792.

Combining the results of Lemma 8 and 9 we conclude that no online algorithm can achieve a factor better than $1.607/1.792=0.896$ for the SMCP.