Improved Bounds in Stochastic Matching and Optimization

Alok Baveja[*1], Amit Chavan[2], Andrei Nikiforov[1], Aravind Srinivasan[†2], and Pan Xu[‡2]

1 **School of Business, Rutgers, The State University of New Jersey, Camden, NJ 08102, USA**
{baveja@crab|andnikif@camden}.rutgers.edu

2 **Department of Computer Science, University of Maryland, College Park, MD 20742, USA**
{amitc|srin|panxu}@cs.umd.edu

Abstract

We consider two fundamental problems in stochastic optimization: approximation algorithms for stochastic matching, and sampling bounds in the black-box model. For the former, we improve the current-best bound of 3.709 due to Adamczyk, Grandoni, and Mukherjee [1], to 3.224; we also present improvements on Bansal, Gupta, Li, Mestre, Nagarajan, and Rudra [2] for hypergraph matching and for relaxed versions of the problem. In the context of stochastic optimization, we improve upon the sampling bounds of Charikar, Chekuri, and Pál [3].

1998 ACM Subject Classification F.2.2 Computations on discrete structures

Keywords and phrases stochastic matching, approximation algorithms, sampling complexity

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.124

1 Introduction

Stochastic optimization deals with problems where there is uncertainty inherent in the input [14]; this classical sub-area of optimization has received much attention in computer science over the last decade, especially from the viewpoints of approximation algorithms and of (efficiently) handling various models for the input (see, e.g., [3, 5, 10, 11, 12, 13, 15, 16]). We make progress on two basic problems in this regard. First, matching is well-known to be a bedrock of combinatorial optimization – a problem that has also played a key role in the advancement of new algorithmic paradigms including parallel algorithms, randomized algorithms, and, more recently, online algorithms in sponsored-search advertising. However, we do not yet have a full algorithmic understanding even for various basic *stochastic* versions of the problem. We advance this goal by improving upon the bounds of [2] and [1] for the matching problem in graphs and in uniform hypergraphs. Second, a fundamental model in this field is the *black-box* model: we assume that the input-distribution is represented by a black box, from which we can sample inputs independently any number of times (in addition to other assumptions on the input's structure). Thus, a key question is the number of samples

* Supported in part by grants from (a) United States Department of Transportation (Via the University Transportation Research Center Program) – Grant# 49198-25-26 and (b) British Council's UKIERI research program.
† Supported in part by NSF Awards CNS 1010789 and CCF 1422569 and a research award from Adobe, Inc.
‡ Supported in part by NSF Awards CNS 1010789 and CCF 1422569.

18th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'15) / 19th Int'l Workshop on Randomization and Computation (RANDOM'15).
Editors: Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim; pp. 124–134

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

needed to solve various stochastic-optimization problems in this model, as a function of, e.g., the desired accuracy ϵ and the confidence (probability of successfully estimating to within ϵ).

Informally, the basic stochastic-matching problem is as follows [2, 4]. We are given a graph $G = (V, E)$ with a weight $w_e \geq 0$ and a probability $p_e \in [0, 1]$ for each edge e; each vertex v also has a positive integral "patience" t_v. Our goal is to construct a matching of maximum weight; however, there are a few catches. First, the edges are only present probabilistically: each edge e is present *independently* with probability p_e, and the presence (or lack thereof) of any edge e can only be ascertained by probing for it – adaptively, in any order we choose. However, if we choose to probe $e = (u, v)$ and find that it is present, we are forced to add it to our matching: in particular, all edges incident on e are removed immediately if e is found to be present. Furthermore, the edges incident upon any vertex v can only be probed for up to t_v times; i.e., we cannot exceed the hard constraint of the patience of any vertex. Under these constraints, the goal is to find a matching of maximum expected weight, where the expectation is taken both over the stochastic existence of the edges, and over any internal randomization of our algorithm. Intriguingly, it is not yet known if this problem is NP-hard. The state of the art in terms of approximation is mainly from the work of [2], who present a 3–approximation for bipartite graphs, and a 4–approximation for general graphs. Recently, these bounds have been improved to 2.845 and 3.709 for bipartite and general graphs respectively [1]. We present the following two improvements for the general graphs, with Theorem 2 being a bicriteria result that allows the patience constraints to be violated by at most 1:

▶ **Theorem 1.** *There is a* 3.224*–approximation algorithm for the weighted stochastic matching problem on a general graph.*

▶ **Theorem 2.** *There is a* 2.675*–approximation algorithm for the weighted stochastic matching problem on a general graph if patience constraints are allowed to be violated by* 1*.*

In essence, the LP-based approach of [2] uses a dependent-rounding algorithm of [7] to first guarantee that the patience constraints are satisfied with probability one within the context of their randomized algorithm; the probing is done on top of this setup. In contrast, we randomly permute the edges and then probe them in this order, with probing probabilities suggested by the LP – of course, not probing infeasible edges in the process. An edge is infeasible if a neighboring edge has already been placed in the matching, or if one of the two end-points has had its patience exhausted. While it is not too hard to incorporate the matching constraints here, the patience constraints are far more complex to handle well: e.g., direct use of Chernoff-type bounds will not help. We work to identify extremal input-instances for our algorithm and combine this with rigorous computer-aided calculations in order to conduct our analyses. Theorem 2 follows from a new attenuation idea. The algorithms themselves are quite simple to implement; the main feature of our work is a detailed analysis of the worst-case settings for our algorithms. All calculations and proofs omitted from this preliminary version will appear in the full version.

Theorem 3 and Theorem 4 improve upon the $(k + 1)$–approximation of [2] for weighted matching in k-uniform hypergraphs. Both of these algorithms use first to classify the hyperedges as "small" or "large" based on the LP values, and treat each group separately. The difference is as follows. The algorithm of Theorem 3 attenuates the small edges to boost the performance of large edges; the algorithm of Theorem 4 uses a "weighted permutation" of the hyperedges such that each large edge has a higher chance to fall behind a small edge. Although Theorem 4 is asymptotically better, we present both theorems since their ideas can be useful elsewhere. Note that the LP-based methods of [2] and ours cannot in general do better than $k - 1 + 1/k$ [6]; hence, we are close to optimal for LP-based approaches.

▶ **Theorem 3.** *There is a $(k+\frac{1}{2}+o(1))$–approximation algorithm for the stochastic matching problem on a k-uniform hypergraph, where the "$o(1)$" term is a function of k that vanishes asymptotically.*

▶ **Theorem 4.** *For any given $\epsilon > 0$, there is a $(k+\epsilon+o(1))$–approximation algorithm for the stochastic matching problem on a k-uniform hypergraph, where the "$o(1)$" term is a function of k that vanishes asymptotically.*

Finally, we significantly improve upon the sample complexity of [3] for stochastic optimization in the black-box model. Since the bounds are somewhat technical, we defer discussion of the actual parameters to Section 5: please see Theorems 13 and 14 for statements of the state-of-the-art and of our improvement. The analysis of [3] has different worst-case settings, but we show that the values of the parameters are very different in these different regimes. This enables a careful analysis of how many samples the approach really needs. This black-box model is quite general, and an improved sample complexity translates to more-efficient implementations of the several applications of the work of [3] (see, e.g., [8, 9, 17]).

Preliminaries. We will often consider a uniformly random permutation π on a set of items $\{e_1, \ldots, e_n\}$. We can assume that π is chosen as follows: for each item e, we pick independently and uniformly at random a real number $\pi(e) = a_e \in [0, 1]$, and then sort these in increasing order to obtain π. Note that we abuse notation by letting π denote both the permutation and the reals chosen; however, this choice will be clear from the context.

We make use of the following form of the Chernoff bound:

▶ **Definition 5** (Chernoff Bound.)**.** Let $X_1, \ldots, X_n$ be n independent random variables with $0 \leq X_i \leq 1$. Let $X = X_1 + \ldots + X_n$ and $\mu = \mathbb{E}[X]$. Then for any $\varepsilon > 0$,

$$\Pr[X \geq (1+\varepsilon)\mu] \leq \exp\left(-\frac{\varepsilon^2}{2+\varepsilon}\mu\right), \text{ and}$$
$$\Pr[X \leq (1-\varepsilon)\mu] \leq \exp\left(-\frac{\varepsilon^2}{2}\mu\right)$$

2 Stochastic Matching

We consider the following stochastic matching problem. The input is an undirected graph $G = (V, E)$ with a weight w_e and a probability value p_e on each edge $e \in E$. In addition, there is an integer value t_v – the *patience* – for each vertex $v \in V$. Initially, each vertex $v \in V$ has patience t_v. At any step in the algorithm, only an edge $e(u, v) \in E$ such that $t_u > 0$ and $t_v > 0$ can be probed. Upon probing such an edge e, one of the following happens: (1) with probability p_e, e exists; u and v get matched and are removed from G along with their incident edges, or (2) with probability $(1 - p_e)$, e does not exist; e is removed, and t_u and t_v are reduced by 1. (All these edge-existence events are independent.) We seek to find an adaptive strategy for probing edges; its performance is measured by the expected weight of the matched edges. We prove Theorem 1 now.

Consider the following natural LP relaxation [2]: for any vertex $v \in V$, $\partial(v)$ denotes the edges incident to v. The LP variable y_e denotes the probability that edge $e(u, v)$ gets probed in the adaptive strategy, and $x_e = y_e p_e$ denotes the probability that e gets matched in the strategy.

$$\text{maximize} \sum_{e \in E} w_e x_e \tag{2.1}$$

$$\text{subject to} \sum_{e \in \partial(v)} x_e \leq 1 \qquad \forall v \in V \tag{2.2}$$

$$\sum_{e \in \partial(v)} y_e \leq t_v \qquad \forall v \in V \tag{2.3}$$

$$x_e = y_e p_e \geq 0,\ y_e \leq 1 \qquad \forall e \in E \tag{2.4}$$

▶ **Lemma 6** ([2]). *The optimal value for the LP* (2.1) *is an upper bound on the performance of any adaptive algorithm for stochastic matching.*

We use (x, y) to denote the optimal solution to the LP in equation (2.1). For an edge $e(u, v)$, it is called *safe* at the time it is considered if: (1) neither u nor v is matched, and (2) both of $t_u > 0$ and $t_v > 0$. Our algorithm, denoted by SM_1, first fixes a uniformly random permutation π on the set of edges E. It then inspects the edges one by one in the order of π. If an edge e is safe, the algorithm probes it (independently) with probability y_e, otherwise it skips to the next one. For ease of analysis, we state our algorithm SM_1 in a slightly different but equivalent way in Algorithm 1.

Algorithm 1: SM_1: Stochastic Matching

1 Choose a random permutation π on E.
2 For each edge $e \in E$, generate a random bit $Y_e = 1$ independently with probability y_e. Let E' be the set of edges with $Y_e = 1$.
3 Follow the random order π to inspect edges in E'
4 If an edge e is safe, then probe it; otherwise, skip it.

To analyze the performance of our algorithm, we conduct an edge-by-edge analysis. Recall that the LP variable $x_e = y_e p_e$ denotes the probability that e is matched in the LP (2.1), and the optimal value of the LP is exactly $\sum_{e \in E} w_e p_e y_e$. The expected weight of the matching found by our algorithm is $\mathbb{E}[\mathsf{SM}_1] = \sum_{e \in E} w_e p_e \Pr[e \in E'] \Pr[e \text{ gets probed} | e \in E']$, which is $\sum_{e \in E} w_e p_e y_e \Pr[e \text{ gets probed} | e \in E'] \geq \sum_{e \in E} w_e p_e y_e \lambda$, assuming $\Pr[e \text{ gets probed} | e \in E'] \geq \lambda$. This gives us a λ-approximation algorithm.

We now start to discuss how to compute the value of λ. Focus on a specific edge $e = e(u, v)$, let $E(u)$ be the set of edges incident to u *excluding* e itself, i.e. $E(u) = \partial(u) \setminus \{e\}$. Conditioning on $\pi(e) = x$ with $0 < x < 1$ and $Y_e = 1$, let $\mathcal{P}_u$ be the probability that e is *not blocked* by any of edges in $E(u)$ in the algorithm SM_1. Here we say e is blocked by some edge f in $E(u)$ if f gets matched or patience constraint on u gets tight resulting from probing f ($t_u = 0$). We assume without loss of generality that $|E(u)| \geq t_u$, otherwise the patience constraint for node u will be redundant.

A little thought gives us the following lower bound on $\mathcal{P}_u$:

$$\mathcal{P}_u \geq P_u = \sum_{S \subseteq E(u), |S| \leq t_u - 1} x^{|S|} \prod_{f \in S} y_f (1 - p_f) \prod_{f \notin S} (1 - x y_f) \tag{2.5}$$

To see why this is true, let Y'_f (for any $f \in E(u)$) be the indicator random variable that is 1 iff f gets matched when probed, i.e., $\Pr[Y'_f = 1] = p_f$. For each $S \subseteq E(u)$, such that $|S| \leq t_u - 1$, we associate an event E_S that says: "(1) each edge $f \in S$ falls before e in π

with $Y_f = 1$ and $Y'_f = 0$; and (2) each edge $f \notin S$ either falls after e in π or $Y_f = 0$". We can see that each E_S is a sufficient condition for e being not blocked by any edge of S. Thus $\mathcal{P}_u$ should be at least the probability that one or more of E_S happen, which is exactly P_u.

In the following paragraphs, we carefully investigate the configuration of edges that minimizes the value of P_u. We denote such adversarial configurations as the **worst-case structure (WS)** of $E(u)$. For each of these configurations, we have the constraints: (i) $\sum_{f \in E(u)} y_f p_f \leq 1$, (ii) $\sum_{f \in E(u)} y_f \leq t_u$ and (iii) $0 \leq y_f \leq 1$ for each $f \in E(u)$. Here we view x as a (given) parameter.

▶ **Lemma 7.** *In WS, there will be at most one edge with $p_f = 1$ and at most one edge with $0 < p_f < 1$. All other edges must have $p_f = 0$.*

Proof. We prove by contradiction. Assume there are two edges, say $p_1 = p_2 = 1$ in WS. Then $y_1 + y_2 \leq 1$ since $\sum_i y_i p_i \leq 1$. We perturb the current configuration as follows: merge the two edges into a single edge e_3 where $y_3 = y_1 + y_2$ and $p_3 = 1$. Notice that after this perturbation, both of the values $\sum_{f \in E(u)} y_f p_f$ and $\sum_{f \in E(u)} y_f$ remain unchanged. Thus both of matching and patience constraints are maintained at u, and our perturbation gives a feasible configuration.

The change brought by this perturbation to the value P_u is as follows: for each non-zero term in P_u associated with some $S \subseteq E(u)$ where $e_1 \notin S, e_2 \notin S$, the term $(1 - xy_1)(1 - xy_2)$ will be replaced with $(1 - x(y_1 + y_2))$, which results in a strictly lower value of P_u. This is a contradiction.

Now assume there are two edges a, b with $0 < p_a, p_b < 1$ in WS. Consider the following perturbation: for some small $\varepsilon \neq 0$, set $p'_a = p_a + \varepsilon/y_a$ and $p'_b = p_b - \varepsilon/y_b$. After this perturbation, both of $\sum_{f \in E(u)} y_f p_f$ and $\sum_{f \in E(u)} y_f$ remains unchanged and the perturbed configuration is still feasible.

Let $f(\varepsilon)$ be the value of P_u after this update. In the expression of P_u, the terms contributing to ε^2 must be those associated with S where $a, b \in S$. Notice that

$$(1 - p'_a)(1 - p'_b) = (1 - p_a - \varepsilon/y_a)(1 - p_b + \varepsilon/y_b)$$

has a negative coefficient of ε^2, implying that the second derivative $f'' < 0$. Therefore we can always find a non-zero value of ε to make P_u strictly smaller. Again a contradiction. ◀

Let $E_1(u)$ and $E_0(u)$ be the set of edges in WS which have $p_f = 1$ and $p_f = 0$ respectively. Let (y_a, p_a) be the potential edge taking a floating $0 < p_a < 1$ value. Lemma 7 tells us $E_1(u)$ contains at most one edge in WS. Let $A = \sum_{f \in E_1(u)} y_f$.

Based on Lemma 7, we can update the expression of P_u as

$$P_u = (1 - xA)(1 - xy_a) \Pr[Z_u \leq t_u - 1] + (1 - xA)xy_a(1 - p_a) \Pr[Z_u \leq t_u - 2] \tag{2.6}$$

where $Z_u = \sum_{f \in E_0(u)} Z_f$ and $\{Z_f | f \in E_0(u)\}$ are independent Bernoulli random variables with $\Pr[Z_f = 1] = xy_f, \forall f \in E_0(u)$. Here are two useful lemmas; the proofs will appear in the full version.

▶ **Lemma 8.** *In WS, $p_a = 0$.*

From Lemma 8, we can claim that there is no edge f that takes fractional p_f value. Thus we can further simplify the expression of P_u in equation (2.6) as

$$P_u = (1 - xA) \Pr[Z_u \leq t_u - 1] \tag{2.7}$$

▶ **Lemma 9.** *In WS, $A = 1$ and Z_u follows Poisson distribution with mean $x(t_u - 1)$.*

At this point, we have all the essentials to prove Theorem 1.

Proof. We have $\Pr[e \text{ gets probed } |Y_e = 1] = \int_0^1 \mathcal{P}_u \mathcal{P}_v dx \geq \int_0^1 P_u P_v dx$, i.e., at least

$$H(t_u, t_v) \doteq \int_0^1 (1-x)^2 \Pr[Z_u \leq t_u - 1] \Pr[Z_v \leq t_v - 1] dx,$$

where Z_u and Z_v follow Poisson distributions with means $\mathbb{E}[Z_u] = x(t_u - 1)$ and $\mathbb{E}[Z_v] = x(t_v - 1)$ respectively. The rest of the analysis splits into the following three cases.

- We can numerically verify that $H(t_u, t_v)$ achieves its minimum value of $0.31016 = 1/3.224$ at $t_u = t_v = 2$ when $1 \leq t_u, t_v \leq 20$.
- For $t_u, t_v \geq 20$, by applying the Chernoff bound, we get

$$H(t_u, t_v) \geq \int_0^1 (1-x)^2 \left[1 - \exp\left(\frac{-\epsilon^2}{2+\epsilon} x(t_u - 1)\right)\right] \left[1 - \exp\left(\frac{-\epsilon^2}{2+\epsilon} x(t_v - 1\right)\right] dx,$$

  where $\epsilon = \epsilon(x) = \frac{1}{x} - 1$; by plugging in $t_u = t_v = 20$, we can verify numerically that this integral is at least 0.316324.
- Similarly, for $1 \leq t_u \leq 20$ while $t_v \geq 20$, we can verify numerically (by checking all integers $1 \leq t_u \leq 20$) that with $\epsilon = \frac{1}{x} - 1$,

$$H(t_u, t_v) \geq \int_0^1 (1-x)^2 \Pr(Z_u \leq t_u - 1) \left[1 - \exp\left(\frac{-\epsilon^2}{2+\epsilon} x(20 - 1)\right)\right] dx \geq 0.312253.$$

This establishes the key claim that $\Pr[e \text{ gets probed } |Y_e = 1] \geq 0.3106$ for each $e \in E$. ◀

3 Stochastic Matching with Relaxed Patience

In this section, we consider the variant of the stochastic matching problem in which patience constraints are allowed to be violated by at most 1, and prove Theorem 2. From the analysis of Section 2, we observe that the edges with a large $y_e p_e$ value are probed with a much higher probability than those with small ones. This indicates that small edges are the ones that bottleneck the performance of our algorithm. Our high level idea here is to *attenuate* such "large" edges in order to improve the performance of the small ones. The process of attenuation carefully calculates a value $h_e \in (0, 1]$, called as the *attenuation factor*, for each $e \in E$. Instead of probing an edge e with probability y_e as in algorithm SM_1, our new algorithm probes it with probability $h_e y_e$. We will show that such a strategy balances the performance of large and small edges and improves the overall performance of SM_1.

The overall picture of our algorithm, denoted SM_2, is as follows. First we label each edge $e \in E$ as "large" if $y_e p_e > 1/2$ or "small" if $y_e p_e \leq 1/2$. Similar to SM_1, we follow a random permutation π on the set of edges E to inspect each edge. If an edge e is safe when considered, we probe it with probability $h_e y_e$; otherwise we skip it. Here $h_e = h$ if e is large and $h_e = 1$ otherwise, where $h \geq 1/2$ is a parameter which we optimize later. For ease of analysis, we state the algorithm SM_2 in an alternate but essentially equivalent way in Algorithm 2. In the spirit of Section 2, we conduct an edge-by-edge analysis. The full analysis will appear in the full version.

Algorithm 2: SM_2: Stochastic Matching with relaxed patience

1 Choose a random permutation π of E.
2 For each edge $e \in E$, set $h_e = h$ if $y_e p_e > 1/2$, set $h_e = 1$ otherwise.
3 For each edge $e \in E$, generate a random bit $Y_e = 1$ with probability $h_e y_e$. Let E' be the set of edges with $Y_e = 1$.
4 Follow the random order π to inspect edges in E'
5 If an edge e is safe, probe it; otherwise, skip it.

4 Stochastic Hypergraph Matching

We now consider stochastic matching in a k-uniform hypergraph, i.e., a hypergraph where all edges have size k. The standard LP can be obtained by naturally extending the LP in (2.1) to the one below:

$$\max \sum_{e \in E} w_e x_e : \quad \sum_{e \in \partial(v)} x_e \leq 1, \forall v \in V, x_e = y_e p_e \geq 0, y_e \leq 1, \forall e \in E \tag{4.1}$$

Note that we do not consider the patience parameter at a vertex, as in Section 2. Here $\partial(v)$ denotes the set of hyperedges incident to v.

4.1 An algorithm achieving $(k + 1/2 + o(1))$ approximation ratio

Let (x, y) be an optimal solution to the LP (4.1). At a high level, our algorithm proceeds according to the outline below. Let $c \geq 1/2$ be a parameter, which will be optimized at $1/2$ later.

1. Divide the edges into two sets, the "small" edge set $E_S = \{e | y_e p_e \leq c\}$, and the "large" edge set $E_L = E \setminus E_S$.
2. Choose a random permutation π on E_S.
3. Sample each edge $e \in E_S$ with probability y_e, independent of other edges. Let E'_S be the set of edges sampled.
4. Follow the random order π to inspect if each small edge $e \in E'_S$ is safe or not. If e is safe, probe it with probability h_e; otherwise, skip it. Here $0 < h_e \leq 1$ is a parameter to fix later.
5. After inspecting all small edges, remove all the unsafe large edges from E_L, and probe the rest with probability 1 (in arbitrary order).

Roughly speaking, an edge e being "safe" means none of the edges in the neighborhood of e are matched. Later, we will give a definition that is both stronger and *exactly* computable. Based on the new definition, we compute an *attenuation factor* h_e for each $e \in E_S$, such that at the end of the algorithm, e is probed with probability *exactly equal* to y_e/λ. Here, $\lambda \geq 1$ is our target approximation ratio. All that remains is to analyze the performance of each large edge $e \in E_L$ and show that e is probed with probability at least y_e/λ. That gives us a λ–approximation algorithm.

We redefine the notion of a small edge e being safe. Suppose π is the random order on E_S and $\pi(e) = x, 0 < x < 1$. Let $N_S[e]$ be the set of small edges in the neighborhood of e. For each $f \in N_S[e]$, let X_f, Y_f, Z_f be three random variables such that: $X_f = 1$ if f falls before e in π, $Y_f = 1$ if $f \in E'_S$ and $Z_f = 1$ if f exists in the hypergraph when probed. Note that the collection of random variables $\{X_f, Y_f, Z_f | f \in N_S[e]\}$ are mutually independent.

For each $f \in N_S[e]$, let A_f be the event that $(X_f + Y_f + Z_f \leq 2)$ and $\mathsf{S}_e = \wedge_{f \in N_S[e]} A_f$. We say e is safe iff S_e happens . Lemma 10 computes the probability that a small edge e is safe in our algorithm.

▶ **Lemma 10.**

$$\Pr[\mathsf{S}_e] = \int_0^1 \Pr[\mathsf{S}_e | \pi(e) = x] dx = \int_0^1 \prod_{f \in N_S[e]} (1 - x y_f p_f) dx. \tag{4.2}$$

Proof. By definition, $\Pr[X_f = 1 | \pi(e) = x] = x$. Note that $\Pr[Y_f = 1] = y_f$, $\Pr[Z_f = 1] = p_f$, and the two values are independent of $\pi(e)$. Thus, given $\pi(e) = x$, A_f will occur with probability $(1 - x y_f p_f)$. Since the A_f are independent for $f \in N_S[e]$, the proof is completed. ◀

Here are two interesting points for the event S_e: (1) When S_e happens, e must be safe according to our initial definition, i.e., none of the edges in its neighborhood get matched; the contrary is not true. Thus the new definition is more strict. (2) On checking e in the algorithm, we might not know if S_e occurs or not due to some missing Z_f for $f \in N_S[e]$. For instance, some $f \in N_S[e]$ gets blocked by some small edge $f' \in N_S[f]$ while $X_f = Y_f = 1$. In this case, we do not know the value of Z_f since f will not be probed. In order to continue our algorithm, we simulate Z_f by generating a random bit $Z_f = 1$ with probability p_f and $Z_f = 0$ otherwise. Notice that if $Z_f = 1$, we will view e as not safe and will not probe it, even though it might be safe according to our initial definition.

The full description and analysis of algorithm in Theorem 3 will appear in the full version.

4.2 An algorithm achieving $(k + \epsilon + o(1))$ approximation ratio

In this section, we present a randomized algorithm that achieves an approximation ratio of $(k + \epsilon + o(1))$ for stochastic matching on a k-uniform hypergraph, where ϵ is given in advance.

Let (x, y) be an optimal solution to the LP (4.1). W.L.O.G we assume $1/\epsilon = N$ where N is an integer. Let a be a constant such that $1 - 1/N < a < 1$. We say an edge e is large if $y_e p_e > 1/N$, otherwise we call e small. For each small edge e, we draw a random real number x_e uniformly from $[0, 1]$. For each large edge e, we draw a random real number x_e from $[0, \delta]$ with density a and from $(\delta, 1]$ with density $(1 - a\delta)/(1 - \delta)$, where $\delta = \min(1, N(1 - a^{1/(N-1)})$. Then we derive a random permutation π by sorting $\{x_e, e \in E\}$ in increasing order. Assuming N is sufficiently large, the value δ is at most $1/N + o(1/N)$. Notice that N, a and δ are all fixed constants. Based on π, we sketch our randomized algorithm below:

Algorithm 3: SM_4: Stochastic Matching on a k-uniform hyergraph

1 Initially all edges are safe.
2 Follow the random order π to check each edge $e \in E$ is safe or not.
3 If e is safe, then probe it with probability y_e; otherwise, skip it.

The lemmas below are useful for the proof of Theorem 4.

▶ **Lemma 11.** *For any $c > 1/N$ and $0 < x < \delta$, we have*

$$1 - axc > (1 - x/N)^{cN}$$

Proof. Define $F(x) = 1 - axc - (1 - x/N)^{cN}$. We can verify that: (1) $F(0) = 0$, and (2) $F'(x) > 0$ for any $0 \leq x < \delta$. This gives the desired result. ◀

Consider an edge $e = (v_1, v_2, \cdots, v_k)$. Suppose $y_e p_e = c_e < 1 - 1/N$ and $x_e = x, 0 < x < \delta$. For each $1 \le i \le k$, let $\partial'(v_i)$ denote the set of edges incident to v_i excluding e itself. Denote by $\mathcal{S}_i$ the event that none of edges in $\partial'(v_i)$ come before e and get matched.

▶ **Lemma 12.**

$$\Pr[\mathcal{S}_i] \ge (1 - x/N)^{(1-c_e)N}$$

The proof of Lemma 12 will appear in the full version. Now we start to prove Theorem 4.

Proof.

1. Consider a small edge e, say $e = (v_1, v_2, \cdots, v_k)$ and $x_e = x$. From Lemma 12, we see $\Pr[\mathcal{S}_i] \ge (1 - x/N)^N$ for each $1 \le i \le k$. Thus by applying the FKG inequality, we get $\Pr\left[\bigwedge_i \mathcal{S}_i\right] \ge (1 - x/N)^{kN}$, which is followed by

$$\Pr[\text{ } e \text{ is checked as safe }] \ge \int_0^\delta (1 - x/N)^{kN} dx = \frac{1}{k + 1/N} - O(k_0^k/k)$$

where $k_0 = (1 - \delta/N)^N < 1$ is a constant.

2. Consider a large edge e, say $e = (v_1, v_2, \cdots, v_k)$ and $x_e = x$. From Lemma 12, we see $\Pr[\mathcal{S}_i] \ge (1 - x/N)^{N-1}$ for each $1 \le i \le k$. Thus by applying FKG, we see when $x \le \delta$, $\Pr\left[\bigwedge_i \mathcal{S}_i\right] \ge (1 - x/N)^{k(N-1)}$, which is followed by

$$\Pr[\text{ } e \text{ is checked as safe }] \ge \int_0^\delta a(1-x/N)^{k(N-1)} dx = \frac{aN}{N-1} \frac{1}{k + 1/(N-1)} - O(k_0^k/k) > \frac{1}{k}$$

where $k_0 = (1 - \delta/N)^{N-1} < 1$ is a constant; we use the fact that $a > 1 - 1/N$ to get the last inequality above. ◀

5 Sample Complexity of Black-Box Stochastic Optimization

In this section, we consider the following *two-stage* stochastic minimization program

$$\min_{x \in X} f(x) = c(x) + \mathop{\mathbb{E}}_{\omega}[q(x, \omega)]. \tag{5.1}$$

An important context in which this problem arises is *two-stage stochastic optimization with recourse*. In this model, a *first-stage* decision $x \in X$ has to be made while having only probabilistic information about the future, represented by the probability distribution π on Ω. Then, after a particular future scenario $\omega \in \Omega$ is realized, a *recourse action* $r \in R$ may be taken to ensure that the requirements of the scenario ω are satisfied. In the two-stage model, $c(x)$ denotes the cost of taking the first stage action x. Given a particular scenario ω and a first-stage action x, the cost of the second stage $q(x, \omega)$ is represented as

$$q(x, \omega) = \min_{r \in R} \{\text{cost}_\omega(x, r) | (x, r) \text{ is a feasible solution for scenario } \omega\}.$$

A natural approach to solve problems modeled by equation (5.1) is to take some number, N, of independent samples $\omega_1, \ldots, \omega_N$ from the distribution π, and to approximate the function f by the sample-average function

$$\hat{f}(x) = c(x) + \frac{1}{N} \sum_{i=1}^{N} q(x, \omega_i). \tag{5.2}$$

One might then hope that for a suitably chosen sample size N, a good solution $\hat{x}$ to equation (5.2) would be a good solution to f; more precisely, $\hat{x} \in X$ is an α-approximate minimizer of the function f defined in (5.1), if for all $x \in X$, $f(\hat{x}) \leq \alpha f(x)$. This approach, called the *sample average approximation* method (SAA), was considered by Charikar, Chekuri, and Pál [3], who considered a setting with the following properties:

(P1) Non-negativity. $c(x) \geq 0$ and $q(x, \omega) \geq 0$ for each $x \in X$ and $\omega \in \Omega$.
(P2) Empty First stage. We assume there is an empty first stage action, $0 \in X$ with $c(0) = 0$, $q(x, \omega) \leq q(0, \omega)$ for each $x \in X, \omega \in \Omega$.
(P3) Bounded Inflation Factor. For each $x \in X, \omega \in \Omega$, we have $q(0, \omega) - q(x, \omega) \leq \lambda c(x)$.

In such a setting, a key result of [3] is:

▶ **Theorem 13** ([3]). *There is a constant $K_0 > 0$ such that the following holds. Any exact minimizer $\bar{x}$ of the function $\hat{f}$ defined in* (5.2) *constructed with $K_0 \cdot \frac{\lambda^2}{\varepsilon^4} \log |X| \log \frac{1}{\delta}$ samples is, with probability at least $1 - \delta$, an $(1 + O(\varepsilon))$-approximate minimizer of the function f defined in* (5.1).

Our result on improved sample complexity states as follows. The proof will appear in the full version.

▶ **Theorem 14.** *There is a constant $K_1 > 0$ such that the following holds. Any exact minimizer $\bar{x}$ of the function $\hat{f}$ defined in* (5.2) *constructed with $N = K_1 \cdot \log \frac{|X|}{\delta} \cdot \max\left[\frac{\lambda^2}{\varepsilon^2}, \frac{\lambda}{\varepsilon^3}\right]$ samples is, with probability at least $1-\delta$, an $(1+O(\varepsilon))$-approximate minimizer of the function f defined in* (5.1).

This improvement, in turn, improves the runtime of the several applications that employ this sampling framework; see, e.g., [8, 9, 17].

Acknowledgements. We thank the conference referees for their helpful suggestions.

References

1 Marek Adamczyk, Fabrizio Grandoni, and Joydeep Mukherjee. Improved approximation algorithms for stochastic matching. *CoRR*, abs/1505.01439, 2015.
2 Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When LP is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithmica*, 63(4):733–762, 2012.
3 Moses Charikar, Chandra Chekuri, and Martin Pál. Sampling bounds for stochastic optimization. In *Proceedings of the 8th International Workshop on Approximation, Randomization and Combinatorial Optimization Problems, and Proceedings of the 9th International Conference on Randamization and Computation: Algorithms and Techniques*, APPROX'05/RANDOM'05, pages 257–269. Springer-Verlag, 2005.
4 Ning Chen, Nicole Immorlica, Anna R Karlin, Mohammad Mahdian, and Atri Rudra. Approximating matches made in heaven. *Automata, Languages and Programming*, pages 266–278, 2009.
5 Brian C Dean, Michel X Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Mathematics of Operations Research*, 33(4):945–964, 2008.
6 Zoltán Füredi, Jeff Kahn, and Paul D. Seymour. On the fractional matching polytope of a hypergraph. *Combinatorica*, 13(2):167–180, 1993.

7 Rajiv Gandhi, Samir Khuller, Srinivasan Parthasarathy, and Aravind Srinivasan. Dependent rounding and its applications to approximation algorithms. *Journal of the ACM (JACM)*, 53(3):324–360, 2006.

8 Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems. In *SODA*, pages 942–951, 2008.

9 Anupam Gupta and Amit Kumar. A constant-factor approximation for stochastic steiner forest. In *STOC*, pages 659–668, 2009.

10 Anupam Gupta, R Ravi, and Amitabh Sinha. An edge in time saves nine: Lp rounding approximation algorithms for stochastic network design. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 218–227. IEEE, 2004.

11 Nicole Immorlica, David Karger, Maria Minkoff, and Vahab S Mirrokni. On the costs and benefits of procrastination: approximation algorithms for stochastic combinatorial optimization problems. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 691–700. Society for Industrial and Applied Mathematics, 2004.

12 Anton J Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

13 Jeff Linderoth, Alexander Shapiro, and Stephen Wright. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*, 142(1):215–241, 2006.

14 Andrzej Ruszczynski and Alexander Shapiro. Stochastic programming. *Handbooks in operations research and management science*, 2003.

15 Alexander Shapiro. Monte Carlo sampling methods. *Handbooks in operations research and management science*, 10:353–425, 2003.

16 David B Shmoys and Chaitanya Swamy. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. *Journal of the ACM (JACM)*, 53(6):978–1012, 2006.

17 A. Srinivasan. Approximation algorithms for stochastic and risk-averse optimization. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 1305–1313, 2007.