Stochastic Matching in Hypergraphs

Amit Chavan, Srijan Kumar and Pan Xu

Department of Computer Science, University of Maryland, College Park

May 13, 2014

Abstract

We study the stochastic matching problem on k-uniform hypergraphs. In this problem, we are given a hypergraph $\mathcal{H}$ on V vertices. With each hyperedge E, we are given a probability p_e of its existence i.e. an edge e exists in the hypergraph with probability p_e, independent of all other edges. Each edge also has a weight associated with it. Our goal is to construct a matching with maximum expected weight. The only way we can find out if the an edge e exists or not is to try to query it, and if the edge is indeed present in the graph, we must include it in our matching. How should we *adaptively* query the edge set so that we attain our goal?

We present an LP based $\frac{1}{k+1/2} + o(1/k)$ approximation algorithm for this problem. This improves on the current best known bound of $k + 1$ due to Bansal et al. [1]. We use the standard LP relaxation for the hypergraph matching problem to get an upper bound on the optimal algorithm. The LP variables give us a hint on the probabilities with which to probe edges. We then present a new attenuation scheme which "attenuates" the edge probabilities given by the LP to get a good approximation ratio. Intuitively, attenuation is needed because if an edge is probed with very high probability, it hurts some of the other edges. Instead of attenuating all the edges by the same factor (as in Bansal et al. [1]), our scheme attenuates each edge differently.

1 Introduction

Definition Given a (hyper)graph $G(V, E)$ a *matching* or *independent edge set* is a subset of E such that no two of them have a vertex in common.

Matching is a fundamental primitive of many markets including job markets, commercial markets, and even dating markets. While matching is a well understood graph-theoretic concept, its stochastic variants are considerably less well developed. Yet stochastic variants are precisely the relevant framework for most markets which incorporate a degree of uncertainty regarding the preferences of agents.

Matching has many applications applied to real life problems, such as resource allocation problem, stable marriage problem, latin square problem, and others. In resource allocation problem, the aim is get maximum profit/utility by satisfy the requests for resources by the requesters, such that resources can not be shared. This can be modelled by taking resources and requesters as nodes and edges representing the requests made by requesters. A matching in that case would represent an allocation of resources to requesters and a satisfied requester would give a profit/utility. In case of stable marriage, the nodes are men and women and in a stable matching, no one has an incentive to elope with someone else. Latin squares are $n \times n$ matrices which has to be filled with numbers $1 \ldots n$ such that each row and each column has all the numbers exactly once. In such

a setting, each box can be represented by nodes representing numbers $1 \ldots n$ all connected to each other and the node would be connected to the node representing the same number from other boxes in its row and column. Therefore, for filling the Latin square correctly, we need to find the maximum independent set, which is equivalent to finding the maximum matching using the following transformation - by constructing a new graph where nodes correspond to edges from the original graph. The edges in the new graph connect those nodes which have common endpoint in the original graph. Sudoku is a special case of Latin square with an additional constraint that 3×3 matrices should also have all $1 \ldots 9$ numbers.

Motivated by applications in kidney exchanges and online dating, Chen et al. [3] studied the following stochastic matching problem: we want to find a maximum matching in a graph G on n nodes, where each edge $e(u,v) \in [\binom{n}{2}]$ exists with probability p_e, independently of the other edges. However, all we are given are the probability values $\{p_e\}$. To find out whether the random graph G has the edge $e(u,v)$ or not, we have to try to add the edge e to our current matching (assuming that u and v are both unmatched in our current partial matching) – we call this "probing" edge e. As a result of the probe, we also find out if e exists or not – and if it does exists, it gets irrevocably added to M, our current partial matching. Such policies make sense, e.g., for dating agencies, where the only way to find out if two people are actually compatible is to send them on a date; moreover, if they do turn out to be compatible, then it makes sense to match them to each other. Finally, to model the fact that there might be a limit on the number of unsuccessful dates a person might be willing to participate in, "timeouts" on vertices are also provided. More precisely, valid policies are allowed, for each vertex u, to only probe at most t_u edges incident to u. Similar considerations arise in kidney exchanges, details of which appear in [3].

Dean et al. [4] studied the stochastic variant of the classical NP-hard knapsack problem: we are given a collection of n items such that for each item $i \in [n]$, $v_i \geq 0$ denotes its value and $s_i \geq 0$ denotes its size. The v_is are deterministic while the s_is are independent random variables with known, arbitrary distributions. The exact size of an item is only revealed when it is put into the knapsack and the objective is to maximize the expected value of items placed in the knapsack. Formally, we seek to find a "policy" which is a mapping $2^{[n]} \times [0,1] \to [n]$ that specifies the next item to insert into the knapsack given the set of remaining (un-instantiated) available items as well as the remaining capacity in the knapsack. Such a policy can be *adaptive*, making decisions in a dynamic fashion in reaction to the instantiated sizes of items already placed in the knapsack. By contrast, a *non-adaptive* policy specifies an entire solution in advance, making no further decisions as items are being inserted. See figure 1 from Dean et al. [5] for an illustration.

(a)

Item	Size distribution (capacity = 1)
1	0.2 (prob 1/2) 0.6 (prob 1/2)
2	0.8 (prob 1)
3	0.4 (prob 1/2) 0.9 (prob 1/2)

(c)

Item	Value	Size distribution (capacity = 1)
1	ε	0 (prob 1/2) 1 (prob 1/2)
2	ε	1 (prob 1)
3	1	0 (prob ε) 1 + ε (prob 1 − ε)

(b)

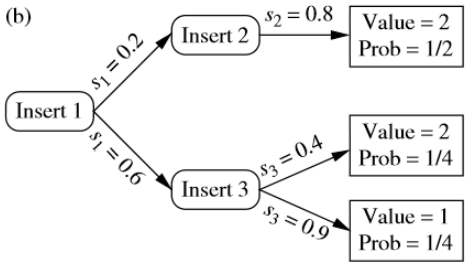

FIGURE 1. Instances of the stochastic knapsack problem.

Notes. For (a), an optimal nonadaptive policy inserts items in the order 1, 2, 3, and achieves expected value 1.5. An optimal adaptive policy, shown as a decision tree in (b), achieves an expected value of 1.75, for an adaptivity gap of 7/6. The instance in (c) has an adaptivity gap arbitrarily close to 5/4: An optimal nonadaptive policy inserts items in the order 1, 3, 2 for an expected value of $2\varepsilon + \frac{1}{2}\varepsilon^2$, and an optimal adaptive policy inserts item 1 followed by items 2 and 3 (if $s_1 = 0$) or item 3 (if $s_1 = 1$), for an expected value of 2.5ε.

There are many problems in stochastic combinatorial optimization for which one could consider designing either adaptive or nonadaptive solution policies. In particular, these are problems in which a solution is incrementally constructed via a series of decisions, each of which establishes a small part of the total solution and also results in the instantiation of a small part of the problem instance. When trying to solve such a problem, it is often a more complicated undertaking to design a good adaptive policy, but this might give us substantially better performance than a nonadaptive policy. To quantify the benefit we gain from adaptivity, Dean et al. [5] advocate the use of *adaptivity gap*, which measures the maximum (i.e., worst-case) ratio over all instances of a problem of the expected value obtained by an optimal adaptive policy to the expected value obtained by an optimal nonadaptive policy.

2 Related Work

Many variations of the stochastic knapsack problem have been studied in the literature. For the one dimenional case, Dean et al. [5] give a non-adaptive 4-approximation algorithm to the optimal adaptive policy and thus show that the adaptive gap in one dimenional case is a constant. They also present an adaptive $(3+\varepsilon)$ approximation to the optimal adaptive strategy. In subsequent work, they show that in the d-dimensional stochastic knapsack, the adaptivity gap is $\Omega(\sqrt{d})$ and is at most $O(d)$.

Bansal et al. [1] measure the performance of their algorithm according to the column sparsity parameter – an instance I is k-column sparse if none of the items are allowed to have non zero size in more than k co-ordinates (out of d). They present a $2k$ approximation algorithm in the general case of k-set packing. However, for the case when the column outcomes are monotone, their algorithm gives a $k+1$ approximation to the optimal adaptive.

In the non-stochastic setting for hypergraph matching, the standard LP relaxation is known to have an integrality gap of $(k - 1 + 1/k)$. This was shown by Füredi et al. [6]. They also show the existence of a matching with the same ratio, but their proof is non-constructive. Chan and Lau [2] later on give a constuctive algorithm to get a matching with maximum weight at least $1/(k - 1 + 1/k)$ times the optimal.

3 Stochastic k-set packing

We first present a result by Bansal et al. [1] which gives a $2k$-approximation to the stochastic k-set packing problem.

3.1 Preliminaries

Formally, the input to this stochastic k-set packing problem consists of

- n items/columns, where each item has a random profit $v_i \in \mathbb{R}_+$, and a random d-dimensional size $s_i \in \{0,1\}^d$; these random values and sizes are drawn from a probability distribution specified as part of the input. We note that the size-vector s_i and profit v_i of each item i are allowed to be correlated (this is what distinguishes the current model from the one considered by [5]). The probability distributions for different items are independent. Additionally, for each item, there is a set C_i of at most k coordinates

such that each size vector takes positive values only in these coordinates; i.e., $s_i \subseteq C_i$ with probability 1 for each item i.

- A capacity vector $b \in \mathbb{Z}_+$ into which the items must be packed.

The parameter k is called the column sparsity of the problem. The instantiation of any column (i.e., its size and profit) is known only when it is probed. The goal is to compute an adaptive strategy of choosing items until there is no more available capacity such that the expectation of the obtained profit is maximized.

3.2 An LP relaxation

For each item $i \in [n]$ and constraint $j \in [d]$, let $\mu_i(j) = \mathbb{E}[s_i(j)]$, the expected value of the j-th coordinate in size-vector s_i. For each column $i \in [n]$, the coordinates $\{j \in [d] | \mu_i(j) > 0\}$ are called the support of column i. By column sparsity, the support of each column has size at most k. Also, let $w_i = \mathbb{E}[v_i]$, be the mean profit, for each $i \in [n]$. We now consider the natural LP relaxation for this problem as below

$$\text{maximize} \sum_{i=1}^{n} w_i y_i \tag{1}$$

$$\text{subject to} \sum_{i=1}^{n} \mu_i(j) y_i \leq b_j, \forall j \in [d] \tag{2}$$

$$y_i \in [0,1] \qquad , \forall i \in [n] \tag{3}$$

We first show that the LP (1)) is a valid relaxation for the stochastic k-set-packing problem.

Lemma 1. *The optimal value for LP (1)) is an upper bound on any (adaptive) algorithm for stochastic k-set-packing.*

Proof. Let p_i be the probability that an adaptive strategy A packs item i. To show the claim, it suffices to show that p_is satisfy the packing constraints for any adaptive strategy $\mathcal{A}$. Consider the j-th constraint. Conditioned on any instantiation of all items, $\mathcal{A}$ can pack at most b_j items for which $\mu_i(j) = 1$, since $\mathcal{A}$ is a safe policy. Hence these constraints hold unconditionally as well, which implies that any valid strategy satisfies the packing constraints. □

Let y^* denote an optimal solution to this linear program, which in turn gives us an upper bound on any adaptive (safe) strategy. The rounding algorithm proceeds as follows. Fix a constant $\alpha > 1$, to be specified later. We pick a uniformly random permutation $\pi : [n] \to [n]$ on all columns, and probes only a subset of the columns as follows. At any point in the algorithm, column c is safe iff there is positive residual capacity in all the coordinates in the support of c-in other words, irrespective of the instantiation of s_c, it can be feasibly packed with the previously chosen columns. The algorithm inspects columns in the order of π, and whenever it is safe to probe the next column $c \in [n]$, it does so with probability y_c/α . Note that the algorithm skips all columns that are unsafe at the time they appear in π.

We show that this algorithm is a $2k$-approximation for a suitable value of α. For any column $c \in [n]$, let $\{I_{c,l}\}_{l=1}^{k}$ denote the indicator random variable that the l-th constraint in the support of c is tight when c is

considered in π. Then

$$\Pr[c \text{ is safe when considered}] = \Pr[\wedge_{l=1}^{k} \neg I_{c,l}] \geq 1 - \sum_{l=1}^{k} \Pr[I_{c,l}]$$

Lemma 2. *For any column c and index l,*

$$\Pr[I_{c,l}] \leq \frac{1}{2\alpha}$$

Proof. Let $j \in [d]$ be the l-th constraint in the support of c. Let U_c^j denote the usage of constraint j, when c is considered.

$$\begin{aligned}
\mathbb{E}[U_c^j] &= \sum_{a=1}^{n} \Pr[\text{column } a \text{ appears before } c \text{ AND } a \text{ is probed}]\mu_a(j) \\
&\leq \sum_{a=1}^{n} \Pr[\text{column } a \text{ appears before } c]\frac{y_a}{\alpha}\mu_a(j) \\
&= \sum_{a=1}^{n} \frac{y_a}{2\alpha}\mu_a(j) \\
&\leq \frac{b_j}{2\alpha}
\end{aligned}$$

Since $I_{c,l} = \{U_c^j \geq b_j\}$, by Markov's inequality, $\Pr[I_{c,l}] \leq \frac{\mathbb{E}[U_c^j]}{b_j} \leq \frac{1}{2\alpha}$. □

Again using the union bound, the probability that a particular column c is safe when considered under π is at least $1 - \frac{k}{2\alpha}$, and thus the probability of actually probing c is at least $\frac{y_c}{\alpha}(1 - \frac{k}{2\alpha})$. Finally, by linearity of expectations (since the instantiation of item c is independent of the event that it is probed) the expected profit is at least $\frac{1}{\alpha}(1 - \frac{k}{2\alpha}) \sum_{c=1}^{n} w_c y_c$. Setting $\alpha = k$ implies an approximation ratio of $2k$.

References

[1] 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. In *Algorithms–ESA 2010*, pages 218–229. Springer, 2010.

[2] Yuk Hei Chan and Lap Chi Lau. On linear and semidefinite programming relaxations for hypergraph matching. *Mathematical programming*, 135(1-2):123–148, 2012.

[3] 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.

[4] Brian C Dean, Michel X Goemans, and Jan Vondrák. Adaptivity and approximation for stochastic packing problems. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 395–404. Society for Industrial and Applied Mathematics, 2005.

[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.