Greedy algorithm for stochastic matching is a 2-approximation

Marek Adamczyk ∗

July 20, 2010

Abstract

The stochastic matching problem was first introduced by Chen, Immorlica, Karlin, Mahdian, and Rudra (ICALP 2009). They presented greedy algorithm together with an analysis showing that this is a 4-approximation. They also presented modification of this problem called multiple-rounds matching, and gave $O(\log n)$-approximation algorithm. Many questions were remaining after this work: is the greedy algorithm a 2-approximation, is there a constant-ratio algorithm for multiple-rounds matching, and what about weighted graphs? For the last two problems constant-factor approximations were given in the work of Bansal, Gupta, Nagarajan, and Rudra, and in the work of Li and Mestre. In this paper we are answering to the first question by showing that the greedy algorithm is in fact a 2-approximation.

1 Introduction

1.1 Problem statement

We are given an undirected graph $G = (V, E)$ in which every edge $uv \in E$ is assigned a real number $0 < p_{uv} \leq 1$. Every vertex $v \in V$ has a positive integer number $t_v$, called a patience number, associated with it. In every step we can probe any edge $uv \in E$, but only if $t_u > 0$ and $t_v > 0$. Probing edge $uv$ will end with success with probability $p_{uv}$, and after that vertices $u, v$ will be removed from the graph with all edges going out of them. In case of a failure, which has probability $1 - p_{uv}$, edge $uv$ is removed from graph, and patience numbers $t_u, t_v$ are decreased by 1. Results of all probes are independent. If after a certain step the patience number $t_v$ of vertex $v$ becomes 0, we remove vertex $v$ with all edges incident to it. Our goal is to maximize the average number of successful probes.

An instance of our problem will be pair $(G, t)$ – undirected weighted graph $G$ with patience numbers $t_v$ for every vertex $v \in G$. Set of edges which were probed successfully forms a matching. We will call such edges as taken into matching.

∗Faculty of Mathematics, Informatics and Mechanics at the University of Warsaw, Poland. Email: marek.adamczyk@students.mimuw.edu.pl
1.2 Motivation

We will give a motivations for this problem. More detailed justification of the model can be found in [1].

Kidney Exchange  A patient waiting for a kidney may have a donor, a friend or a family member, who would like to donate him a kidney. But it may happen that the patient has incompatible donors among his friends and family. If we consider many such patient-donor pairs, it may be possible to find two such pairs in which donor from the first pair may donate a kidney to a second patient, and the second donor may donate kidney to a first patient. Performing such operations on four people is called kidney exchange. To know whether two pairs can perform kidney exchange, we need to make a set of three tests. The first two tests are easy to make, e.g. they include blood-type tests. The third test is relatively hard, and checks possibility of exchange only between two particular pairs. Also it is required to perform exchange as soon as two pairs were matched. Thus we can not check every two pairs to see whether they can exchange or not. However, we can use the results of first two tests to estimate the probability of a successful result in the third test. Moreover, every patient can only be put to a limited number of tests, depending on his health. The goal is to perform maximal number of transplants.

Dating  Let us consider a dating portal, where users are offered to make acquaintances with other users. On the basis of personal details of each user we can estimate a probability of matching for each pair of users. We can suggest a meeting to a pair of users, but only after the meeting we can say if they like each other. In case of a successful matching, we assume they will leave the portal. While patience of each user is an individual case, every user will give up after several unsuccessful meetings and will stop using the portal. The objective is to suggest this acquaintances to the users in the way which maximize the number of satisfied couples.

1.3 Related work

Problem we are considering belongs to the field of stochastic optimization. It includes many of well known problems to which some elements of uncertainty were added. Since matchings are very popular combinatorial object used in models from many areas, lots of applications motivate stochastic matching problems, e.g. the classic problem of online bipartite matching, solved by Karp et. al. [5], has evolved in many ways to face needs of online advertising [6, 7]. In those problems we are given one side of bipartite graph (advertisers), and each vertex from the other side (ad impression) is revealed to us one by one together with edges coming out of it. Right after arrival of a vertex we need to match it with some vertex from the first side. Although this problem is not precisely related to ours, Bansal et. al. [3] gave an online variant of our stochastic matchings problem which can be seen as a generalization of online bipartite matching.

First work concerning our problem was the paper of Chen et. al. [1]. This work introduced a stochastic matching problem together with its modification – multiple-rounds matching. In this modified version we are given a fixed number of rounds, and in each we are allowed to probe any set of edges which forms a matching. They showed that greedy algorithm for the main version of the problem is a 4-approximation for all graphs. For the multiple-rounds version they gave $O(\log n)$ approximation. They also proved that finding an optimal multiple-rounds strategy is NP-hard. Authors also asked about weighted version of this problem in which every edge has assigned real and positive weight.
Here the goal is to maximize the expected weight of taken matching. Unfortunately, modification of the greedy algorithm which probes edges sorted by its product of weight and probability does not work, and can not give any constant-factor approximation.

Recently Bansal et. al. [3] and Li with Mestre [2] showed how linear programming can be used to obtain constant approximation ratio for stochastic matchings and multiple-rounds matching. A very huge advantage of this approach is that considering weighted edges, almost does not make any difference.

Bansal et. al. obtained 5.75 ratio for weighted case. They also showed LP-based analysis to prove that greedy algorithm is a 5-approximation. For multiple-rounds matching they gave a constant-factor approximation breaking $O(\log n)$ ratio. They also introduced two new modifications. First is an online version which can be seen as a generalization of online bipartite matching mentioned earlier. Second modification considers case when we are probing hyperedges. For both variants constant-ratio approximations were given.

Li and Mestre also gave a constant-factor approximations for weighted and multiple-rounds versions. They also improved bounds for the basic version of the problem. In special case, when all probabilities are equal and graph is bipartite, they got a factor $\frac{2}{1-e^{-1}} \approx 2.3131$. When probabilities can be various their algorithm gives ratio worse than 4, but it can be combined with greedy algorithm to get a 3.51 factor for bipartite graphs and 3.88 for all graphs.

1.4 Our results

In our paper we are focused on the main version of stochastic matching problem, and we do not consider multiple-rounds model nor weighted graphs. The main result of this paper is improved analysis of the greedy algorithm showing that this is a 2-approximation which confirms the hypothesis stated by Chen et. al.

2 Preliminaries

If in a certain step any algorithm probes edge coming out of vertex $\alpha \in G$, then we will say that algorithm probes vertex $\alpha$.

We consider any algorithm deterministic, if probe in each step is unambiguous and depends only on previous steps.

If $ALG$ denotes an algorithm, then $E_{ALG}$ is expected number of edges taken into matching by this algorithm. Moreover, we will use $(G^{ALG}, t^{ALG})$ to denote the instance on which $ALG$ is executed.

Decision tree Each deterministic algorithm $ALG$ can be represented by its decision tree $T_{ALG}$ (which has an exponential size). Each node of that tree corresponds to probing an edge. Node $v \in T_{ALG}$ is assigned a value $p_v$ equal to $p_{\alpha \beta}$ where edge $\alpha \beta$ is probed in $v$. The left subtree of node $v \in T_{ALG}$ represents proceeding of algorithm after a successful probe in node $v$, the right subtree – after failure. More precisely:

- the left subtree corresponds to an algorithm on the instance $(G \setminus \{\alpha, \beta\}, t)$
- the right subtree corresponds to an algorithm on the instance $(G \setminus \{(\alpha, \beta)\}, t')$ where $t'_\alpha = t_\alpha - 1, t'_\beta = t_\beta - 1$ and $t'_\gamma = t_\gamma$ for other vertices $\gamma$
Probability of reaching a node \( v \in T_{ALG} \) will be denoted \( q_v \). The performance of an algorithm \( ALG \) can be expressed using the decision tree:

\[
\mathbb{E}_{ALG} = \sum_{v \in T_{ALG}} q_v p_v.
\]

We will denote above sum as \( \mathbb{E}T_{ALG} \). For a node \( v \in T_{ALG} \), we denote as \( T_v \) subtree of \( T_{ALG} \) rooted in \( v \), and as \( L(v), R(v) \) its left and right subtree respectively.

**Optimal algorithm** The optimal algorithm on instance \((G, t)\) will be denoted \( OPT(G) \) – we will not use \( OPT(G, t) \), because it will always be clear which patience numbers we are using. We can assume without loss of generality that \( OPT(G) \) is deterministic. This lets us represent optimal algorithm by a decision tree. We will also assume that every subtree of the tree \( T_{OPT} \) representing optimal algorithm is optimal on its instance, even when probability of reaching such subtree is zero.

**Greedy algorithm** Let us consider the greedy algorithm for a given graph \( G \). It will be denoted as \( GRD(G) \).

| Greedy algorithm for graph G: |
|-----------------------------|
| 1. sort edges of \( G \) in non-increasing order of weights: |
| \( p_{e_1} \geq p_{e_2} \geq \ldots \geq p_{|E|} \) (order of edges with same weights is not important) |
| 2. for \( i = 1, 2, \ldots, |E| \) probe edge \( e_i \), if it still exists in a graph |

**Empty graph** It is reasonable to consider also an empty graph, because it may appear during our inductive proof. Of course in this case performance of any algorithm is zero.

## 3 Analysis of the greedy algorithm

We will start with an important lemma from [1] (Lemma 3.1.).

**Lemma 3.1.** For any node \( v \in T_{OPT} \), \( \mathbb{E}T(v) \leq \mathbb{E}L(v) + 1 \).

**Proof.** Algorithm which follows \( R(v) \) is a proper algorithm for instance on which \( T(v) \) works, so \( \mathbb{E}R(v) \leq \mathbb{E}T(v) \) – because every subtree of \( T_{OPT} \) is optimal. Hence

\[
\mathbb{E}T(v) = p_v(1 + \mathbb{E}L(v)) + (1 - p_v)\mathbb{E}R(v) \leq p_v(1 + \mathbb{E}L(v)) + (1 - p_v)(\mathbb{E}T(v)).
\]

This gives \( p_v\mathbb{E}T(v) \leq p_v(1 + \mathbb{E}L(v)) \), and finally \( \mathbb{E}T(v) \leq 1 + \mathbb{E}L(v) \). \( \square \)

The following theorem is the main result of this paper.

**Theorem 3.2.** Greedy algorithm for any instance \((G, t)\) of stochastic matching problem is a 2-approximation, i.e. \( \mathbb{E}OPT(G) \leq 2\mathbb{E}GRD(G) \).
The sketch of the proof is as follows. Proof is inductive with respect to subinstances \((G', t')\) of problem \((G, t)\) where \(G'\) is a subgraph of \(G\) and \(t'_v \leq t_v\) for every vertex \(v \in G'\). Using optimal algorithm on instance \((G, t)\), we derive two algorithms – first on instance \((G^{L_{GRD}}, t^{L_{GRD}})\) and second on \((G^{R_{GRD}}, t^{R_{GRD}})\). We do not know explicitly how those algorithms work. We upperbound performance of \(OPT(G)\) by those algorithms and some residues. Then we use inductive assumption to bound performance of those derived algorithms by greedy algorithms, and show that the residues are "small".

**Proof.** The case when graph has no edges is trivial. So suppose it has at least one edge. Let \(\alpha\beta\) be the first edge probed by the greedy algorithm. Denote as \(L_{GRD}, R_{GRD}\) algorithms which follow left and right subtree of \(T_{GRD}\) respectively. When it does not make a problem, we use \(OPT\) instead of \(OPT(G)\).

**Algorithm for instance** \((G^{L_{GRD}}, t^{L_{GRD}})\). In this paragraph we will derive algorithm proper for instance \((G^{L_{GRD}}, t^{L_{GRD}})\). We will not obtain it directly from \(OPT\), but from algorithm \(OPT'\) which also works on the whole instance \((G, t)\).

Let \(X\) be the set of nodes of \(T_{OPT}\) which correspond to probing edge \(\alpha\beta\). Define algorithm \(OPT'\) which follows algorithm \(OPT(G)\), until it reaches a node \(x \in X\). After reaching that node \(OPT'\) probes edge \(\alpha\beta\), but then goes straight to the subtree \(L(x)\) regardless of the probe’s result, i.e. after probe it runs as if the result was successful. This means also that \(OPT'\) after probing \(\alpha\beta\) will not probe \(\alpha\) nor \(\beta\) again. In a node \(x \in X\) the expected number of edges taken in the subtree \(T(x)\) of \(T_{OPT}\) is equal to \(\mathbb{E}T(x) = p_{\alpha\beta} + p_{\alpha\beta} \mathbb{E}L(x) + (1 - p_{\alpha\beta}) \mathbb{E}R(x)\), but in the subtree \(T(x)\) of \(T_{OPT'}\) it is \(\mathbb{E}T(x) = p_{\alpha\beta} + \mathbb{E}L(x)\) – because \(OPT'\) goes to \(L(x)\) regardless of the result. We can bound performance of \(OPT\) using \(OPT'\):

**Lemma 3.3.**

\[
\mathbb{E}OPT \leq \mathbb{E}OPT' + (1 - p_{\alpha\beta}) \mathbb{P}(OPT \text{ probes } \alpha\beta).
\]  

**Proof.** From decision tree of \(OPT\) we get

\[
\mathbb{E}OPT = \sum_{v \in T_{OPT} \setminus T(X)} q_v p_v + \sum_{x \in X} q_x \mathbb{E}T(x).
\]

After using lemma \[3.1\] we obtain

\[
\mathbb{E}OPT \leq \sum_{v \in T_{OPT} \setminus T(X)} q_v p_v + \sum_{x \in X} q_x (1 + \mathbb{E}L(x)).
\]

The performance of \(OPT'\) is equal to

\[
\mathbb{E}OPT' = \sum_{v \in T_{OPT} \setminus T(X)} q_v p_v + \sum_{x \in X} q_x (p_{\alpha\beta} + \mathbb{E}L(x)),
\]

so

\[
\mathbb{E}OPT \leq \sum_{v \in T_{OPT} \setminus T(X)} q_v p_v + \sum_{x \in X} q_x (1 + \mathbb{E}L(x))
= \sum_{v \in T_{OPT} \setminus T(X)} q_v p_v + \sum_{x \in X} q_x (p_{\alpha\beta} + \mathbb{E}L(x)) + \sum_{x \in X} q_x (1 - p_{\alpha\beta}) = \mathbb{E}OPT' + (1 - p_{\alpha\beta}) \sum_{x \in X} q_x.
\]

All we need now is to notice that \(\sum_{x \in X} q_x = \mathbb{P}(OPT \text{ probes } \alpha\beta)\).
Now we will use algorithm $OPT'$ to construct algorithm for instance $(G_{LGRD}, t_{LGRD})$. Let algorithm $ALG_L$ follow all moves of $OPT'$, but only those which does not probe vertices $\alpha$ and $\beta$ – when algorithm $OPT'$ probes vertex $\alpha$ or $\beta$, then algorithm $ALG_L$ does not probe any edge, and waits for result of $OPT'$. We can also define $ALG_L$ using decision trees. $ALG_L$ follows decision tree of $OPT'$, but upon reaching a node $v$, which probes vertex $\alpha$ or $\beta$, it flips a coin, and with probability $p_v$, it goes to the left subtree $L(v)$, and with probability $1 - p_v$ it goes to the right subtree $R(v)$. This (randomized) algorithm is a proper algorithm for instance $(G_{LGRD}, t_{LGRD})$, because graph $G_{LGRD}$ is made from $G$ by removing vertices $\alpha$ and $\beta$. Moreover, for every vertex $v \in G_{LGRD}$ we have $t_{LGRD} = t_v$. Performance of algorithm $ALG_L$ is equal to the performance of $OPT'$ minus penalty for skipped probes – let $R_L$ denote this penalty. Hence

$$E_{OPT'} = E_{ALG_L} + E_{R_L}. \quad (2)$$

Let us look at $R_L$ under two conditions – when $OPT'$ probes $\alpha \beta$, and when it does not probe that edge. If $OPT'$ probes edge $\alpha \beta$, then with probability $p_{\alpha \beta}$ it will take this edge, and after that all probes of $OPT'$ will be valid probes for $ALG_L$, because they will not probe $\alpha$ nor $\beta$. Thus the penalty in that case is equal to $E(R_L|OPT' \text{ probes } \alpha \beta) = p_{\alpha \beta}$. If $OPT'$ does not probe $\alpha \beta$, then the penalty is equal to the expected number of edges incident to $\alpha \beta$ taken by $OPT'$ under this condition:

$$E(R_L|OPT' \text{ does not probe } \alpha \beta) = \mathbb{P}(OPT' \text{ takes } \alpha|OPT' \text{ does not probe } \alpha \beta) + \mathbb{P}(OPT' \text{ takes } \beta|OPT' \text{ does not probe } \alpha \beta).$$

Thus the whole $E_{R_L}$ is equal to

$$\mathbb{P}(OPT' \text{ probes } \alpha \beta)p_{\alpha \beta} + \mathbb{P}(OPT' \text{ does not probe } \alpha \beta)\left(\mathbb{P}(OPT' \text{ takes } \alpha|OPT' \text{ does not probe } \alpha \beta) + \mathbb{P}(OPT' \text{ takes } \beta|OPT' \text{ does not probe } \alpha \beta)\right).$$

From the definition, $OPT'$ works just like $OPT$, unless it reaches $\alpha \beta$, so in place of above expression we can write

$$\mathbb{P}(OPT \text{ probes } \alpha \beta)p_{\alpha \beta} + \mathbb{P}(OPT \text{ does not probe } \alpha \beta)\left(\mathbb{P}(OPT \text{ takes } \alpha|OPT \text{ does not probe } \alpha \beta) + \mathbb{P}(OPT \text{ takes } \beta|OPT \text{ does not probe } \alpha \beta)\right). \quad (3)$$

We will introduce shorter notation. Denote the event of probing $\alpha \beta$ by $OPT$ as ”probe $\alpha \beta$”, and ”$\neg$probe $\alpha \beta$” as opposite event. The event of taking $\alpha$ (or $\beta$) by $OPT$ under the condition of not probing $\alpha \beta$ will be denoted as ”take $\alpha\mid \neg$probe $\alpha \beta$” (or ”take $\beta\mid \neg$probe $\alpha \beta$”). Thus we can write that

$$E_{R_L} = \mathbb{P}(\text{probe } \alpha \beta)p_{\alpha \beta} + \mathbb{P}(\neg \text{ probe } \alpha \beta)(\mathbb{P}(\text{take } \alpha\mid \neg \text{ probe } \alpha \beta) + \mathbb{P}(\text{take } \beta\mid \neg \text{ probe } \alpha \beta)). \quad (3)$$

Joining all gives:

$$E_{OPT} \leq E_{OPT'} + (1 - p_{\alpha \beta})\mathbb{P}(\text{probe } \alpha \beta) \quad \text{by (1)}$$

$$= E_{ALG_L} + E_{R_L} + (1 - p_{\alpha \beta})\mathbb{P}(\text{probe } \alpha \beta) \quad \text{by (2)}$$

$$= E_{ALG_L} + (1 - p_{\alpha \beta})\mathbb{P}(\text{probe } \alpha \beta) + \mathbb{P}(\text{take } \alpha\mid \neg \text{ probe } \alpha \beta) + \mathbb{P}(\text{take } \beta\mid \neg \text{ probe } \alpha \beta) \quad \text{by (3)}.$$ 

Finally we get that

$$E_{OPT} \leq E_{ALG_L} + \mathbb{P}(\text{probe } \alpha \beta) + \mathbb{P}(\neg \text{ probe } \alpha \beta)(\mathbb{P}(\text{take } \alpha\mid \neg \text{ probe } \alpha \beta) + \mathbb{P}(\text{take } \beta\mid \neg \text{ probe } \alpha \beta)). \quad (4)$$
Algorithm for instance \((G^{R_{GRD}}, t^{R_{GRD}})\)  

Instance \((G^{R_{GRD}}, t^{R_{GRD}})\) is made of \((G, t)\) by removing edge \(\alpha \beta\) and decreasing patience numbers of \(\alpha\) and \(\beta\), i.e. \(t_{\alpha}^{R_{GRD}} = t_{\alpha} - 1\) and \(t_{\beta}^{R_{GRD}} = t_{\beta} - 1\). Define algorithm \(ALG_R\) on instance \((G^{R_{GRD}}, t^{R_{GRD}})\) which follows \(OPT(G)\), unless \(OPT(G)\) probes edge which \(ALG_R\) can not. If \(OPT(G)\) makes such probe, then \(ALG_R\) does not do anything, waits for \(OPT(G)\) and follows it further. Definition with decision tree: \(ALG_R\) follows decision tree of \(OPT\), but upon reaching node \(v\) for which it cannot make a probe it flips a coin and with probability \(p_e\) it goes to the left subtree \(L(v)\), and with probability \(1 - p_e\) it goes to the right subtree \(R(v)\).

Let us now describe such invalid probes.

Consider one particular execution of \(OPT(G)\). Suppose that in this execution \(OPT(G)\) probes \(\alpha \beta\). Algorithm \(ALG_R\) can not make this probe because \(\alpha \beta \notin G^{R_{GRD}}\). But before that every probe of \(OPT(G)\) was a valid probe for \(ALG_R\). After \(OPT(G)\) probes \(\alpha \beta\), the patience numbers of \(\alpha\) and \(\beta\) will become equal for algorithms \(OPT(G)\) and \(ALG_R\), so afterwards every probe of \(OPT(G)\) will be a valid probe for \(ALG_R\). So in this case any possible loss of algorithm \(ALG_R\) can be the edge \(\alpha \beta\), but only if \(OPT(G)\) probed it successfully. Suppose now that \(OPT(G)\) did not probe \(\alpha \beta\) in that execution. The only probes of \(OPT(G)\) which are invalid for \(ALG_R\) are probe number \(t_{\alpha} = t_{\alpha}^{R_{GRD}} + 1\) of vertex \(\alpha\), and probe number \(t_{\beta} = t_{\beta}^{R_{GRD}} + 1\) of vertex \(\beta\). Thus in this case \(ALG_R\) can lose only edges probed then.

As before, we write performance of \(OPT\) as performance of \(ALG_R\) plus penalty \(R_R\):

\[
\mathbb{E}OPT = \mathbb{E}ALG_R + \mathbb{E}R_R. \tag{5}
\]

From the above explanation we have

\[
\mathbb{E}R_R = \mathbb{P}(OPT \text{ probes } \alpha \beta) p_{\alpha \beta} + \mathbb{P}(OPT \text{ does not probe } \alpha \beta) \times \\
\times (\mathbb{P}(OPT \text{ takes } \alpha \text{ in probe number } t_{\alpha}|OPT \text{ does not probe } \alpha \beta) \\
+ \mathbb{P}(OPT \text{ takes } \beta \text{ in probe number } t_{\beta}|OPT \text{ does not probe } \alpha \beta)).
\]

Let us write ”take \(\alpha\) in \(t_{\alpha}\)” instead of ”\(OPT\) takes \(\alpha\) in probe number \(t_{\alpha}\)”, and analogically with \(\beta\). Now we can write above equality shorter:

\[
\mathbb{E}R_R = \mathbb{P}(\text{probe } \alpha \beta)p_{\alpha \beta} + \mathbb{P}(\neg\text{probe } \alpha \beta)\left(\mathbb{P}(\text{take } \alpha \text{ in } t_{\alpha}|\neg\text{probe } \alpha \beta) + \mathbb{P}(\text{take } \beta \text{ in } t_{\beta}|\neg\text{probe } \alpha \beta)\right).
\]

Putting this into \((4)\) gives

\[
\mathbb{E}OPT = \mathbb{E}ALG_R + \mathbb{P}(\text{probe } \alpha \beta)p_{\alpha \beta} \\
+ \mathbb{P}(\neg\text{probe } \alpha \beta)\left(\mathbb{P}(\text{take } \alpha |\neg\text{probe } \alpha \beta) + \mathbb{P}(\text{take } \beta |\neg\text{probe } \alpha \beta)\right). \tag{6}
\]

Combining  

Multiplying inequality \((4)\) by \(p_{\alpha \beta}\), and equality \((5)\) by \(1 - p_{\alpha \beta}\), and adding them give

\[
\mathbb{E}OPT \leq p_{\alpha \beta}\mathbb{E}ALG_L + p_{\alpha \beta}\mathbb{P}(\text{probe } \alpha \beta) \\
+ p_{\alpha \beta}\mathbb{P}(\neg\text{probe } \alpha \beta)\left(\mathbb{P}(\text{take } \alpha |\neg\text{probe } \alpha \beta) + \mathbb{P}(\text{take } \beta |\neg\text{probe } \alpha \beta)\right) \\
+ (1 - p_{\alpha \beta})\mathbb{E}ALG_R + (1 - p_{\alpha \beta})\mathbb{P}(\text{probe } \alpha \beta)p_{\alpha \beta} \\
+ (1 - p_{\alpha \beta})\mathbb{P}(\neg\text{probe } \alpha \beta)\left(\mathbb{P}(\text{take } \alpha |\neg\text{probe } \alpha \beta) + \mathbb{P}(\text{take } \beta |\neg\text{probe } \alpha \beta)\right).
\]
After grouping terms in above expression we obtain
\[
E_{OPT} \leq p_{\alpha\beta}E_{ALG_L} + (1 - p_{\alpha\beta})E_{ALG_R} + p_{\alpha\beta}P(\text{probe } \alpha\beta)(2 - p_{\alpha\beta}) \\
+ p_{\alpha\beta}P(\neg \text{ probe } \alpha\beta)(P(\text{take } \alpha | \neg \text{ probe } \alpha\beta) + \frac{1 - p_{\alpha\beta}}{p_{\alpha\beta}}P(\text{take } \alpha \text{ in } t_{\alpha} | \neg \text{ probe } \alpha\beta) \\
+ P(\text{take } \beta | \neg \text{ probe } \alpha\beta) + \frac{1 - p_{\alpha\beta}}{p_{\alpha\beta}}P(\text{take } \beta \text{ in } t_{\beta} | \neg \text{ probe } \alpha\beta)).
\]

We state now the key lemma.

**Lemma 3.4.**
\[
\frac{1 - p_{\alpha\beta}}{p_{\alpha\beta}}P(\text{take } \alpha \text{ in } t_{\alpha} | \neg \text{ probe } \alpha\beta) \leq P(OPT \text{ does not take } \alpha \text{ despite of } t_{\alpha} \text{ probes} | \neg \text{ probe } \alpha\beta).
\]

**Proof.** Probability of taking \( \alpha \) into matching is the sum of probabilities of taking each edge incident to \( \alpha \), so
\[
P(\text{take } \alpha \text{ in } t_{\alpha} | \neg \text{ probe } \alpha\beta) = \sum_{\gamma \in \text{Adj}(\alpha)} P(OPT \text{ takes } \alpha\gamma \text{ in probe number } t_{\alpha} | \neg \text{ probe } \alpha\beta).
\]
Edge \( \alpha\gamma \) can be taken into matching, if we probed this edge, and the probe was successful, i.e.
\[
= \sum_{\gamma \in \text{Adj}(\alpha)} P(OPT \text{ probes } \alpha\gamma \text{ in probe number } t_{\alpha} \text{ AND probe is successful} | \neg \text{ probe } \alpha\beta).
\]
Probe number \( t_{\alpha} \) is the last probe of vertex \( \alpha \) regardless of its result. Thus its result and the fact that \( OPT \) does not probe \( \alpha\beta \) are independent. This gives
\[
\sum_{\gamma \in \text{Adj}(\alpha)} P(OPT \text{ probes } \alpha\gamma \text{ in probe number } t_{\alpha} \text{ AND probe is successful} | \neg \text{ probe } \alpha\beta) \\
= \sum_{\gamma \in \text{Adj}(\alpha)} P(OPT \text{ probes } \alpha\gamma \text{ in probe number } t_{\alpha} | \neg \text{ probe } \alpha\beta)p_{\alpha\gamma}.
\]
Function \( \frac{1 - x}{x} \) is decreasing, and \( p_{\alpha\beta} \) is the greatest weight in the whole graph, so we get
\[
\frac{1 - p_{\alpha\beta}}{p_{\alpha\beta}}P(\text{take } \alpha \text{ in } t_{\alpha} | \neg \text{ probe } \alpha\beta) \\
= \sum_{\gamma \in \text{Adj}(\alpha)} \frac{1 - p_{\alpha\beta}}{p_{\alpha\beta}}P(OPT \text{ probes } \alpha\gamma \text{ in probe number } t_{\alpha} | \neg \text{ probe } \alpha\beta) \\
\leq \sum_{\gamma \in \text{Adj}(\alpha)} \frac{1 - p_{\alpha\gamma}}{p_{\alpha\gamma}}P(OPT \text{ probes } \alpha\gamma \text{ in probe number } t_{\alpha} | \neg \text{ probe } \alpha\beta) \\
= \sum_{\gamma \in \text{Adj}(\alpha)} (1 - p_{\alpha\gamma})P(OPT \text{ probes } \alpha\gamma \text{ in probe number } t_{\alpha} | \neg \text{ probe } \alpha\beta) \\
= \sum_{\gamma \in \text{Adj}(\alpha)} P(OPT \text{ probes } \alpha\gamma \text{ in probe number } t_{\alpha} \text{ AND this probe is unsuccessful} | \neg \text{ probe } \alpha\beta).
The justification of the last equality is the same as in [8].

Just like at the beginning

\[
\sum_{\gamma \in \text{Adj}(\alpha)} \mathbb{P}(OPT \text{ probes } \alpha \gamma \text{ in probe number } t_{\alpha} \text{ AND probe is unsuccessful}|\neg \text{probe } \alpha \beta)
= \mathbb{P}(OPT \text{ does not take } \alpha \text{ despite of } t_{\alpha} \text{ probes}|\neg \text{probe } \alpha \beta),
\]

so the lemma is proved.

**Corollary 3.5.**

\[
\frac{1 - p_{\alpha \beta}}{p_{\alpha \beta}} \mathbb{P}(\text{take } \alpha \text{ in } t_{\alpha}|\neg \text{probe } \alpha \beta) \leq \mathbb{P}(OPT \text{ does not take } \alpha|\neg \text{probe } \alpha \beta)
\]

**Proof.** It follows from the lemma [3.4] and the obvious inequality

\[
\mathbb{P}(OPT \text{ does not take } \alpha \text{ despite of } t_{\alpha} \text{ probes}|\neg \text{probe } \alpha \beta) \leq \mathbb{P}(OPT \text{ does not take } \alpha|\neg \text{probe } \alpha \beta). \]

Of course, lemma [3.4] and corollary [3.5] are true with \(\beta\) in place of \(\alpha\).

The event that \(OPT\) does not take \(\alpha\) is opposite to ”take \(\alpha\)”, so we will denote it as ”\(\neg\) take \(\alpha\)”, and analogically for \(\beta\). This means also that

\[
1 = \mathbb{P}(\text{take } \alpha|\neg \text{probe } \alpha \beta) + \mathbb{P}(\neg \text{take } \alpha|\neg \text{probe } \alpha \beta)
= \mathbb{P}(\text{take } \beta|\neg \text{probe } \alpha \beta) + \mathbb{P}(\neg \text{take } \beta|\neg \text{probe } \alpha \beta).
\]

Finally we get

\[
E^{OPT} \leq p_{\alpha \beta}E^{ALG_L} + (1 - p_{\alpha \beta})E^{ALG_R} + p_{\alpha \beta}\mathbb{P}(\text{probe } \alpha \beta)(2 - p_{\alpha \beta})
+ p_{\alpha \beta}\mathbb{P}(\neg \text{probe } \alpha \beta)(\mathbb{P}(\text{take } \alpha|\neg \text{probe } \alpha \beta) + \frac{1 - p_{\alpha \beta}}{p_{\alpha \beta}}\mathbb{P}(\text{take } \alpha \text{ in } t_{\alpha}|\neg \text{probe } \alpha \beta)
+ \mathbb{P}(\text{take } \beta|\neg \text{probe } \alpha \beta) + \frac{1 - p_{\alpha \beta}}{p_{\alpha \beta}}\mathbb{P}(\text{take } \beta \text{ in } t_{\beta}|\neg \text{probe } \alpha \beta))
\leq p_{\alpha \beta}E^{ALG_L} + (1 - p_{\alpha \beta})E^{ALG_R} + p_{\alpha \beta}\mathbb{P}(\text{probe } \alpha \beta)(2 - p_{\alpha \beta})
+ p_{\alpha \beta}\mathbb{P}(\neg \text{probe } \alpha \beta)(\mathbb{P}(\text{take } \alpha|\neg \text{probe } \alpha \beta) + \mathbb{P}(\neg \text{take } \alpha|\neg \text{probe } \alpha \beta)
+ \mathbb{P}(\text{take } \beta|\neg \text{probe } \alpha \beta) + 2p_{\alpha \beta}\mathbb{P}(\neg \text{probe } \alpha \beta))
= p_{\alpha \beta}E^{ALG_L} + (1 - p_{\alpha \beta})E^{ALG_R} + 2p_{\alpha \beta}\mathbb{P}(\text{probe } \alpha \beta) + 2p_{\alpha \beta}\mathbb{P}(\neg \text{probe } \alpha \beta)
\]

Inductive assumption gives

\[
E^{ALG_L} \leq E^{OPT}(G^{L_{GRD}}) \leq 2E^{GRD}(G^{L_{GRD}}) = 2E^{L_{GRD}}, \text{ and}
\]

\[
E^{ALG_R} \leq E^{OPT}(G^{R_{GRD}}) \leq 2E^{GRD}(G^{R_{GRD}}) = 2E^{R_{GRD}}.
\]

Hence

\[
E^{OPT} \leq p_{\alpha \beta}E^{ALG_L} + (1 - p_{\alpha \beta})E^{ALG_R} + 2p_{\alpha \beta}
\leq 2p_{\alpha \beta}E^{L_{GRD}} + 2(1 - p_{\alpha \beta})E^{R_{GRD}} + 2p_{\alpha \beta}
= 2p_{\alpha \beta}(E^{L_{GRD}} + 1) + 2(1 - p_{\alpha \beta})E^{R_{GRD}} = 2E^{GRD}(G).
\]

This completes the proof. \(\square\)
References

[1] N. Chen, N. Immorlica, A. Karlin, M. Mahdian and A. Rudra. *Approximating Matches Made in Heaven*, ICALP 2009.

[2] J. Li and J. Mestre. *Improved bounds for stochastic matching*, 2010.

[3] N. Bansal, A. Gupta, V. Nagarajan and A. Rudra. *When LP is the Cure for Your Matching Woes: Approximating Stochastic Matchings*, 2010.

[4] N. Bansal, A. Gupta, J. Li, J. Mestre, V. Nagarajan and A. Rudra. *When LP is the Cure for Your Matching Woes: Improved Bounds for Stochastic Matchings*, ESA 2010.

[5] R. Karp, U. Vazirani, and V. Vazirani. *An Optimal Algorithm for On-line Bipartite Matching*, 1990.

[6] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. *Adwords and generalized online matching*, 2005.

[7] J. Feldman, A. Mehta, V. Mirrokni, and S. Muthukrishnan. *Online stochastic matching: Beating $1 - 1/e$*, 2009.