Matching Algorithms are Fast in Sparse Random Graphs

Holger Bast[†] *Kurt Mehlhorn*[†] *Guido Schäfer*[†] *Hisao Tamaki*[‡⋆]

[†] Max-Planck-Institut für Informatik, Saarbrücken, Germany
Email: {bast,mehlhorn,schaefer}@mpi-sb.mpg.de
[‡] Meiji University, Kawasaki, Japan
Email: tamaki@cs.meiji.ac.jp

Abstract

We present an improved average case analysis of the maximum cardinality matching problem. We show that in a bipartite or general random graph on n vertices, with high probability every non-maximum matching has an augmenting path of length $O(\log n)$. This implies that augmenting path algorithms like the Hopcroft–Karp algorithm for bipartite graphs and the Micali–Vazirani algorithm for general graphs, which have a worst case running time of $O(m\sqrt{n})$, run in time $O(m \log n)$ with high probability, where m is the number of edges in the graph. Motwani proved these results for random graphs when the average degree is at least $\ln(n)$ [*Average Case Analysis of Algorithms for Matchings and Related Problems*, Journal of the ACM, **41**(6), 1994]. Our result holds, if only the average degree is a large enough constant. At the same time we simplify the analysis of Motwani.

1 Introduction

We consider the problem of computing a matching of maximum cardinality in an undirected graph $G = (V, E)$ with vertex set V and edge set E. A matching is a subset $M \subseteq E$ of the edges of G such that no two edges in M have a vertex in common. The edges in M are called *matching edges*, edges not in M are called *free edges*. A vertex is *matched* if it has an incident matching edge, otherwise it is *free*.

Augmenting Path Algorithms. Most matching algorithms are augmenting path algorithms. An *augmenting path* for a non-maximum matching M is a simple path between two free vertices, where the edges along the path are alternately free edges and matching edges. For every non-maximum matching, an augmenting path exists (e.g., obtained by taking the symmetric difference of the set of matching edges with the edge set of an arbitrary optimal matching). By making each free edge a matching edge and vice versa along such a path, a matching that is larger by one edge is obtained. Augmenting path algorithms search for augmenting paths and augment, until the matching is maximum. The algorithms differ in the way they search for augmenting paths.

Worst Case Complexity. Maximum matchings can be computed efficiently. Let n and m denote the number of vertices and edges of G, respectively. In bipartite graphs, the

[⋆] This work was done while the author was visiting the Max-Planck Institut für Informatik.

algorithm of Hopcroft and Karp [HK73] computes a maximum matching in time $O(m\sqrt{n})$. For dense graphs, i.e., with $m = \Theta(n^2)$, slightly better algorithms are known. Cheriyan and Mehlhorn [CM96] obtained $O(n^{2.5}/\log n)$ and Feder and Motwani [FM95] achieved, via graph compression, $O(m\sqrt{n}/\varphi(n,m))$, where $\varphi(n,m) = \log n/\log(n^2/m)$. In general graphs, Edmonds' blossom-shrinking algorithm [Edm65b, Edm65a, Gab76] computes a maximum matching in time $O(nm\alpha(m,n))$, where $\alpha(m,n)$ denotes the inverse of Ackerman's function. Micali and Vazirani [MV80] gave an $O(m\sqrt{n})$ algorithm, which is similar to the algorithm of Hopcroft and Karp for bipartite graphs.

The algorithms of Hopcroft and Karp [HK73] and Micali and Vazirani [MV80] are of particular interest in this paper. The algorithms run in phases. In each phase we first construct a maximal set of vertex-disjoint shortest augmenting paths, and then augment the current matching along these paths. A phase requires time $O(m)$. In both algorithms the length of the shortest augmenting path strictly increases from one phase to the next. The following fact is immediate.

Fact 1. *If every non-maximum matching in a bipartite (general) graph has an augmenting path of length at most $f(n)$, then the Hopcroft–Karp (Micali–Vazirani) algorithm runs in time $O(m \cdot f(n))$.*

In practice, augmenting path algorithms perform significantly better than suggested by the worst case running times, see, e.g., [MN99, CGM$^+$98]. The worst case running time seems to be an over-pessimistic estimation of the actual running time in practice. We are therefore interested in the average case behavior of augmenting path algorithms.

Random Graph Models. We define the probability distribution on graphs according to the model introduced by Erdös and Rényi [ER59]. We consider both bipartite and general graphs. We denote by $G(n;n)$ the set of all undirected bipartite graphs with n vertices on each side, and by $G(n;n;p)$ the probability distribution on $G(n;n)$, where each of the n^2 potential edges is present with probability p, independent of other edges. Similarly, we denote by $G(n)$ the set of all undirected graphs with n vertices and by $G(n;p)$ the probability distribution on $G(n)$, where each of the $n(n-1)/2$ potential edges is present with probability p, independent of other edges. The average degree of each vertex in a graph drawn from $G(n;n;p)$ or $G(n;p)$ is pn and $p(n-1)$, respectively. We will use c to denote the average degree of a random graph.

Related Work. Motwani [Mot94] presented the first average case analysis for matching algorithms. He showed that every non-maximum matching in a random graph from $G(n;n;c/n)$ of from $G(n;c/(n-1))$ with $c > \ln n$ has a short augmenting path, i.e., of length $O(\log n)$, with high probability. The analysis rests on two key observations: (i) expander graphs[1] admit short augmenting paths with respect to any non-maximum matching, and (ii) random graphs with $c \geq \ln n$ are structurally so similar to expander graphs that the short augmenting path property carries over. Motwani's analysis breaks down when c is significantly below $\ln n$. When c is constant, for example, with high probability a constant fraction of the vertices is isolated and a constant fraction of the vertices has degree one, and such graphs are certainly not structurally similar to expanders.

[1] In an expander graph the cardinality of the set of neighbours of any vertex set S with $|S| \leq n/2$ is at least $(1+\epsilon)|S|$ for some positive constant ϵ.

Motwani also proved the existence of short augmenting paths for bipartite random graphs in the $G(n; n; d\text{-out})$ model with $d \geq 4$. In this model, each vertex chooses uniformly and independently exactly d neighbours. This model generates sparse graphs, but guarantees that every vertex has at least d neighbours. An expansion property follows.

Our Contribution. We prove that in a random graph drawn from $G(n; n; c/n)$ or from $G(n; c/(n-1)$, with high probability every non-maximum matching has an augmenting path of length $O(\log n)$, if only c is above a certain constant. For bipartite graphs, our analysis requires that $c \geq 9.6$, for general graphs it requires that $c \geq 35.1$. It follows that under these conditions, the running time of the algorithms of Hopcroft and Karp on bipartite random graphs and Micali and Vazirani on general random graphs is $O(m \log n)$ with high probability.

We conjecture the existence of short augmenting paths for every value of c. Observe that for tiny values of c, for example $c < 1$, all paths are of length $O(\log n)$ and hence also all augmenting paths must be short. It is conceivable that our analysis can be strengthened so as to cover all values of c; we comment further on this in our conclusions.

Our high-level line of proof is very similar to that of Motwani. The main novelty of our work is an expansion lemma for *augmenting path trees* as they are constructed in augmenting path algorithms. An augmenting path tree is rooted at a free vertex and the edges along each path from the root to a vertex in the tree are alternately free edges and matching edges. We are able to obtain our result, because we exploit several structural properties of these trees, such as connectivity, degree-one descendence due to the matching edges, etc. We show that such trees must become bushy after the first $\Theta(\log n)$ layers; we remark that they may stay skinny for up to $\Theta(\log n)$ layers. In contrast to this, Motwani works with expansion for plain sets of vertices, which gives rise to several complications in the analysis, which we can avoid here. Our result is therefore at the same time stronger and simpler.

2 Main Result

In this section we state our main result, Theorem 1, explain the central ideas of its proof, and give an overview over the rest of the paper.

Theorem 1. *There is a constant c_0 such that a random graph from $G(n; n; c/n)$ or from $G(n; c/(n-1))$, where $c \geq c_0$, with high probability has the property that every non-maximum matching has an augmenting path of length $O(\log n)$. In a graph with this property, by Fact 1, a maximum matching can be computed in $O(m \log n)$ time, where m is the number of edges.*

Remark 1. For a random graph from $G(n; n; c/n)$, the theorem holds for $c \geq 9.6$. For a random graph from $G(n; c/(n-1))$, it holds for $c \geq 35.1$.

A central notion in our analysis will be that of an *augmenting path tree*. Augmenting path trees arise in the standard breadth-first search for augmenting paths for a given non-maximum matching: start from a free vertex, add all its neighbours, if none of them is free (otherwise an augmenting path is found) add all the incident matching edges and their other endpoints, and so on. We first give the formal definition, then Figure 1 provides an example.

Definition 1. *For a rooted tree T, let* $\mathrm{Even}(T)$ *denote the set of vertices at even non-zero levels (i.e., excluding the root), and let* $\mathrm{Odd}(T)$ *denote the set of vertices at odd levels, where the root has level* 0*, its children have level* 1*, and so on. The largest level of a vertex in T is denoted by* $\mathrm{depth}(T)$*.*

An augmenting path tree *is a rooted tree T of even depth, where each vertex of* $\mathrm{Odd}(T)$ *has exactly one child; in particular,* $|\mathrm{Odd}(T)| = |\mathrm{Even}(T)|$*. An augmenting path tree is* for *a particular matching, if its root is free with respect to that matching, and all edges between an odd level and the next larger even level are in the matching.*

Figure 1: Left: An augmenting path tree T with $|\mathrm{Even}(T)| = |\mathrm{Odd}(T)| = 8$. Right: The tree with vertices on odd levels "removed", as used in the proof of Lemma 2.

Our approach to proving Theorem 1 is as follows. Given a non-maximum matching, we pick the two free vertices of an augmenting path, and, from each of these vertices we grow two augmenting path trees T_1 and T_2. The following lemma names a set of properties, which are sufficient for the existence of a short augmenting path.

Lemma 1. *Let T_1 and T_2 be two augmenting path trees for a given non-maximum matching in a given graph. Then the following properties imply that there is an augmenting path of length at most* $\mathrm{depth}(T_1) + \mathrm{depth}(T_2) + 1$

(a) T_1 and T_2 are (vertex and edge) disjoint;

(b) One of the following holds:

- *(b1) either there is a free vertex adjacent to* $\mathrm{Even}(T_1)$ *or to* $\mathrm{Even}(T_2)$*,*
- *(b2) or there is an edge between* $\mathrm{Even}(T_1)$ *and* $\mathrm{Even}(T_2)$*.*

Proof. If property (b1) holds, there is an augmenting path via just one of the trees, of length at most $\max\{\mathrm{depth}(T_1), \mathrm{depth}(T_2)\} + 1$. If property (b2) holds, then owing to (a) there is an augmenting path from the root of T_1 to the root of T_2 of length at most $\mathrm{depth}(T_1) + \mathrm{depth}(T_2) + 1$.

□

Our construction of the trees T_1 and T_2 with these properties will be incremental, terminating as soon as property (b1) or (b2) is fulfilled. In Section 3, we will give the construction for bipartite random graphs. In Section 4, we deal with general random graphs.

The main difficulty will be to prove that the construction terminates with at most logarithmic depth for both trees. The key will be the following lemma, which establishes an expansion property for augmenting path trees, when the average degree is above a certain constant.

While the lemma is formulated and proven completely independently from its later use, some readers might prefer to first study the construction from Section 3 in more detail, see how the lemma is used there, and then come back to this section. In the lemma below, as well as in our constructions, we will use $\Gamma_G(X)$ to denote the *neighbourhood* of a vertex set X in G, i.e., the set of vertices adjacent to X in G.

Lemma 2. *For each $\varepsilon > 0$ and $\beta > 1+\varepsilon$, there exist constants α and c_0 such that a random graph G from $G(n;n;c/n)$ or from $G(n;c/(n-1))$, where $c \geq c_0$, with high probability has the following property: for each augmenting path tree T with $\alpha \cdot \log n \leq |\mathrm{Even}(T)| \leq n/\beta$, it holds that $|\Gamma_G(\mathrm{Even}(T))| \geq (1+\varepsilon) \cdot |\mathrm{Even}(T)|$.*

Remark 2. For a random graph from $G(n;n;c/n)$, for $\varepsilon = 0.01$ and $\beta = 2.5$, the lemma holds with $c_0 = 9.6$. For a random graph from $G(n;c/(n-1))$, for $\varepsilon = 2.01$ and $\beta = 6.2$, the lemma holds with $c_0 = 35.1$. These will be the settings when we apply the lemma in Sections 3 and 4.

Proof. If a graph G does not have the property from the lemma, the following *bottleneck*[2] constellation occurs in G:

(i) an augmenting path tree T with $\alpha \log n \leq |\mathrm{Even}(T)| \leq n/\beta$;

(ii) a set $\Gamma \supseteq \mathrm{Odd}(T)$ with $|\Gamma| \leq (1+\varepsilon) \cdot |\mathrm{Even}(T)|$;

(iii) for each vertex from $\Gamma \backslash \mathrm{Odd}(T)$, an edge to a vertex from $\mathrm{Even}(T)$ (the edges from $\mathrm{Odd}(T)$ to $\mathrm{Even}(T)$ are already taken care of in (i));

(iv) no edge from $\mathrm{Even}(T) \times V \backslash \Gamma$, where V is the set of all vertices of G.

We will show that the probability that any such bottleneck constellation occurs is polynomially small in n. We first give the proof for a bipartite random graph, and then describe the (few) changes required for a general random graph.

If a fixed bottleneck constellation occurs in a graph from $G(n;n)$, the following events occur, where we write $l = |\mathrm{Even}(T)|$ and $r = |\Gamma|$: (i) the $2l$ edges from T are present, (ii) the $r-l$ edges from $\Gamma \backslash \mathrm{Odd}(T)$ to $\mathrm{Even}(T)$ are present, and (iii) none of the $l(n-r)$ edges from $\mathrm{Even}(T) \times V' \backslash \Gamma$ are present, where V' is the side of the bipartite graph containing Γ and we exploit that in a bipartite graph $\mathrm{Even}(T)$ and Γ lie on opposite sides of the graph. It follows that the probability that each of these, obviously independent, events occurs in a random graph from $G(n;n;c/n)$ is at most

$$(c/n)^{l+r} \cdot (1-c/n)^{l(n-r)},$$

which, using that $l \leq n/\beta$, $l \leq r \leq (1+\varepsilon) \cdot l$, and $1 - c/n \leq e^{-c/n}$, is bounded by

$$n^{-(l+r)} \cdot c^{(2+\varepsilon) \cdot l} \cdot e^{-c(1-(1+\varepsilon)/\beta) \cdot l} = n^{-(l+r)} \cdot \left(\frac{c^{2+\varepsilon}}{e^{c(1-(1+\varepsilon)/\beta)}} \right)^{l}.$$

[2]In his work, Motwani uses this name in a related context.

The number of potential bottleneck constellations, i.e., the number of different bottleneck constellations in the complete bipartite graph on $2n$ vertices, with $|\mathrm{Even}(T)| = l$ and $|\Gamma| = r$ is (i) the number of augmenting path trees T with $|\mathrm{Even}(T)| = l$, times (ii) the number of ways to choose the $r - l$ vertices for $\Gamma\backslash\mathrm{Odd}(T)$ from $V'\backslash\mathrm{Odd}(T)$, where V' are the vertices on that side of the bipartite graph containing $\mathrm{Odd}(T)$ (vertices on the other side of the graph cannot be in the neighbourhood of $\mathrm{Even}(T)$), times (iii) the number of ways to choose for each of these $r-l$ vertices an edge to one of the l vertices from $\mathrm{Even}(T)$.

Clearly, the number for (iii) is l^{r-l}, and the number for (ii) is $\binom{n-l}{r-l} \leq \binom{n}{r-l}$. To count the number of augmenting path trees T with $|\mathrm{Even}(T)| = l$, observe that via "removing" the vertices in $\mathrm{Odd}(T)$, as illustrated by an example in Figure 1, each such tree corresponds to a unique combination of a tree on $l + 1$ vertices, and a sequence of l distinct vertices. By Cayley's theorem [AZ01] the number of trees on $l + 1$ vertices is $(l + 1)^{l-1}$, and the number of sequences of l distinct vertices from one side of a graph from $G(n; n)$ is $n \cdot (n - 1) \cdots (n - l + 1) \leq n^l$.

The total number of potential bottleneck constellations in a $G(n; n)$ graph is hence at most

$$\binom{n}{l+1} \cdot (l+1)^{l-1} \cdot \binom{n}{r-l} \cdot l^{r-l} \cdot n^l \leq n^{r+l+1} \cdot e^{r+1} \cdot \left(\frac{l}{r-l}\right)^{r-l} \leq n^{r+l+1} \cdot e^{r+1} \cdot \left(\varepsilon^{-\varepsilon}\right)^l,$$

where the last inequality holds[3] for $r \leq (1 + \varepsilon) \cdot l$ and $\varepsilon \leq 1/e$.

Combining the bounds, we conclude that a random graph from $G(n; n; c/n)$ contains *any* bottleneck constellation with $|\mathrm{Even}(T)| = l$ and $|\mathrm{Odd}(T)| = r$, with probability at most

$$en \cdot \left(\frac{\varepsilon^{-\varepsilon} e^{1+\varepsilon} c^{2+\varepsilon}}{e^{c(1-(1+\varepsilon)/\beta)}}\right)^l = en \cdot q^l,$$

where q is just an abbreviation for the fractional term. For $\varepsilon = 0.01$, we get $q < 1$ under the following conditions on c: for $\beta = 2$, $c \geq 12.2$ suffices; for $\beta = 2.5$, $c \geq 9.3$ suffices, for sufficiently large β, $c \geq 3.7$ suffices. For $\beta > 2$, Lemma 3 below will put the additional constraint $c > 2\beta(1 + \ln \beta)$, which for $\beta = 2.5$ is $c > 9.58$. Hence the 9.6 in the remark to Theorem 1.

We finally sum over all r, l with $\alpha \cdot \log n \leq l \leq n/\beta$ and $l \leq r \leq (1 + \varepsilon) \cdot l$, and get a total probability of at most

$$en^3 \cdot q^{\alpha \log n} = en^{3-\alpha \log(1/q)},$$

which for sufficiently large α is polynomially small in n. This finishes the proof of Lemma 2 for bipartite random graphs.

In a random graph from $G(n; c/(n - 1))$, the bound on the probability that a fixed constellation with $|\mathrm{Even}(T)| = l$ and $|\Gamma| = r$ occurs, is

$$(c/(n-1))^{l+r} \cdot (1 - c/n)^{l(n-r)} \leq (n-1)^{-(l+r)} \cdot \left(\frac{c^{2+\varepsilon}}{e^{c(1-(1+\varepsilon)/\beta)}}\right)^l.$$

[3] Let $r = (1 + \kappa) \cdot l$ with $0 \leq \kappa \leq \varepsilon$. If $\kappa = 0$, the claim is obvious (recall $0^0 = 1$). If $\gamma > 0$, we have $(l/(r - l))^{r-l} = (1/\kappa)^{\kappa \cdot l} \leq (1/\varepsilon)^{\varepsilon \cdot l}$, since $(1/\kappa)^\kappa$ is increasing for $\kappa \leq 1/e$.

The number of bottleneck constellations can be bounded just like before by

$$\binom{n}{l+1} \cdot (l+1)^{l-1} \cdot \binom{n}{r-l} \cdot l^{r-l} \cdot n^l \leq n^{r+l+1} \cdot e^{r+1} \cdot \left(\frac{l}{r-l}\right)^{r-l} \leq n^{r+l+1} \cdot e^{r+1} \cdot 1.45^l,$$

where the last inequality now holds for $r \leq (1+\varepsilon) \cdot l$, but without restriction on ε (for arbitrary random graphs, we will apply the lemma with $\varepsilon > 2$). Altogether, a random graph from $G(n; c/(n-1))$ then contains *any* bottleneck constellation with probability at most

$$\sum_{\alpha \log n \leq l \leq n} \sum_{l \leq r \leq n} e^3 n \cdot \left(\frac{e^{1.38+\varepsilon} c^{2+\varepsilon}}{e^{c(1-(1+\varepsilon)/\beta)}}\right)^l \leq en^3 \cdot q^{\alpha \log n} = en^{3-\alpha \log(1/q)},$$

where the additional e^2 factor comes from $(n-1)^{-(l+r)} \cdot n^{r+l} \leq (1+1/(n-1))^{r+l} \leq e^2$, the 1.38 is just a number $\geq 1 + \ln 1.45$, and q is again an abbreviation for the fractional term. For $\varepsilon = 2.01$, $\beta = 6.2$ and $c \geq 35.1$, we have $q < 1$, so that for large enough α the probability is negligible. This proves Lemma 2 also for arbitrary random graphs.

□

The following simple property of random graphs was already stated and proven in [Mot94, Lemma 3(d)], except that Motwani did not make the threshold on c explicit. We remark that this threshold is one of the major bottlenecks for reducing the threshold on c in our main result, Theorem 1.

Lemma 3. *For every $\beta > 1$, a random graph from $G(n; n; c/n)$ or from $G(n; c/(n-1))$, where $c > 2\beta(1 + \ln \beta)$, with high probability has the property that every two disjoint sets of vertices, both of size at least n/β, have an edge between them.*

Proof. The probability that no edge runs between two disjoint sets of sizes l and r is exactly $(1 - c/n)^{lr}$. If two disjoint subsets of size at least n/β and with no edge between them exist, then there exist also two subsets of size exactly $\lceil n/\beta \rceil$ with no edge between them (just remove the necessary number of vertices from each set), and this happens with probability at most

$$\binom{n}{\lceil n/\beta \rceil}^2 \cdot (1 - c/n)^{\lceil n/\beta \rceil^2} \leq (e\beta)^{2n/\beta} \cdot e^{-cn/\beta^2} = (e\beta / e^{c/(2\beta)})^{2n/\beta}.$$

Provided that $e^{c/(2\beta)} > e\beta$, or equivalently, $c > 2\beta \ln(e\beta) = 2\beta(1 + \ln \beta)$, this is a negligible probability.

□

3 Constructing the Trees for Bipartite Random Graphs

For a given non-maximum matching of a graph G from $G(n; n)$, consider an augmenting path and pick its two free endpoints, f_1 and f_2. Note that since in a bipartite graph every augmenting path has odd length, these two free vertices lie on opposite sides of G. The following procedure constructs T_1 and T_2.

0. Initially let T_1 and T_2 be the trees with f_1 and f_2 as the only vertex and root, respectively. Each of the following iterations will add two more levels to T_1 and to T_2.

1. Let $\Gamma(T) = \Gamma_G(\mathrm{Even}(T)) \setminus \mathrm{Odd}(T)$, for $T = T_1, T_2$.

2. If $\Gamma(T_1)$ or $\Gamma(T_2)$ contains a free vertex, STOP.

3. If $\Gamma(T_1)$ contains a vertex which is already in $\mathrm{Even}(T_2)$, or vice versa, STOP.

4. Otherwise add to T all the vertices from $\Gamma(T)$, together with the edges connecting them to $\mathrm{Even}(T)$ (by construction all vertices from $\Gamma(T)$ are in fact adjacent to the largest level of T), for $T = T_1, T_2$.

5. Add the matching edges incident to $\Gamma(T)$ together with their other endpoints to T, for $T = T_1, T_2$.

6. Repeat 1.–5.

We first show that this construction fulfills the properties of Lemma 1. When the procedure stops in step 2, we have property (b1). When it stops in step 3, we have encountered a neighbour of $\mathrm{Even}(T_1)$ which is already in $\mathrm{Even}(T_2)$ or vice versa, i.e., there is an edge between $\mathrm{Even}(T_1)$ and $\mathrm{Even}(T_2)$, which is property (b2). Since the roots of T_1 and T_2 lie on opposite sides of the bipartite graph G, we have $\mathrm{Even}(T_1) \cap \mathrm{Even}(T_2) = \mathrm{Odd}(T_1) \cap \mathrm{Odd}(T_2) = \emptyset$. Step 3 ensures that $\mathrm{Odd}(T_1) \cap \mathrm{Even}(T_2) = \mathrm{Odd}(T_2) \cap \mathrm{Even}(T_1) = \emptyset$, hence we have complete disjointness of T_1 and T_2, which is property (a).

It remains to show that the procedure terminates within $O(\log n)$ iterations (note that by what we have shown so far, the procedure could even run forever, namely when at some point $\Gamma(T) = \emptyset$ in step 1). Since each iteration adds two levels to each tree, the depth of the trees would then be $O(\log n)$, which by Lemma 1 would prove Theorem 1.

By construction, in every iteration at least the matching edge of the augmenting path starting in f_1 is added to T_1, and the same holds for f_2 and T_2. After $\alpha \log n$ iterations therefore, $|\mathrm{Even}(T)| \geq \alpha \cdot \log n$. Consider an iteration i, $i > \alpha \cdot \log n$. Let T denote one of the trees (the following argument holds for T_1 as well as for T_2) after a new odd level has been added in step 4, and let T' denote the tree after a new even level has been added in step 5. If $|\mathrm{Even}(T)| < n/\beta$, Lemma 2 applies, and we have $|\Gamma_G(\mathrm{Even}(T))| \geq (1+\varepsilon) \cdot |\mathrm{Even}(T)|$. Since $|\mathrm{Even}(T')| = |\mathrm{Even}(T)| + |\Gamma(T)| = |\mathrm{Even}(T)| + |\Gamma_G(\mathrm{Even}(T)) \setminus \mathrm{Odd}(T)| = |\Gamma_G(\mathrm{Even}(T))|$, we have $|\mathrm{Even}(T')| \geq (1+\varepsilon) \cdot |\mathrm{Even}(T)|$. This proves that when the procedure runs for $\alpha \log n + \log_{1+\varepsilon}(n/\beta) = O(\log n)$ iterations, then certainly $|\mathrm{Even}(T)| \geq n/\beta$, for $T = T_1, T_2$.

Consider the first iteration, where both $|\mathrm{Even}(T_1)|$ and $|\mathrm{Even}(T_2)|$ are at least n/β. By property (a), already established above, the two sets are disjoint, hence by Lemma 3, with high probability there is an edge between them. With such an edge, the procedure stops in step 3. This proves that with high probability the procedure terminates within $O(\log n)$ iterations, and hence with two trees of depth $O(\log n)$. This finishes the proof of Theorem 1 for random bipartite graphs.

4 Constructing the Trees for Arbitrary Random Graphs

For a given non-maximum matching of a graph G from $G(n)$, consider an augmenting path and pick its two free endpoints, f_1 and f_2. The procedure for constructing T_1 and T_2 is similar as for bipartite graphs but with three complications: (i) two vertices from the neighbourhood of $\mathrm{Even}(T_1)$ or of $\mathrm{Even}(T_2)$ may be incident to the same matching edge, so that we can add only one of them to the tree (step 4 below), (ii) the disjointness of the neighbourhoods of $\mathrm{Even}(T_1)$ and $\mathrm{Even}(T_2)$ has to be taken care of explicitly now (step 5 below), and (iii) because only part of the neighbourhood of $\mathrm{Even}(T)$ is eventually added to T, for $T = T_1, T_2$, starting from the free vertices alone it could now indeed happen that $\Gamma(T_1) = \emptyset$ or $\Gamma(T_2) = \emptyset$ in one of the first $\alpha \log n$ iterations; therefore in step 0 we now start with a piece of size $2\lceil \alpha \log n \rceil$ of the augmenting path for each tree.

0. Let T_1 be the prefix of length $2\lceil \alpha \log n \rceil$ of the augmenting path starting at f_1, and let T_2 be the suffix of length $2\lceil \alpha \log n \rceil$. If the two are not disjoint, i.e., the length of the augmenting path is $4\alpha \log n$ or less, remove $T_1 \cap T_2$ from one of the trees and STOP (the properties of Lemma 1 are then fulfilled). Otherwise, $|\mathrm{Even}(T_1)|, |\mathrm{Even}(T_2)| \geq \alpha \log n$, and each of the following iterations will add two more levels to both T_1 and T_2.

1. Let $\Gamma(T_1) = \Gamma_G(\mathrm{Even}(T_1)) \backslash (T_1 \cup \mathrm{Odd}(T_2))$, and let $\Gamma(T_2) = \Gamma_G(\mathrm{Even}(T_2)) \backslash (T_2 \cup \mathrm{Odd}(T_1))$.

2. If $\Gamma(T_1)$ or $\Gamma(T_2)$ contains a free vertex, STOP.

3. If $\Gamma(T_1)$ contains a vertex which is already contained in $\mathrm{Even}(T_2)$, or vice versa, STOP.

4. Let $\Gamma'(T)$ be a maximal subset of $\Gamma(T)$ in which no two vertices match each other, for $T = T_1, T_2$; then $|\Gamma'(T)| \geq \lceil |\Gamma(T)|/2 \rceil$.

5. Let $\Gamma''(T_1) \subseteq \Gamma'(T_1)$ and $\Gamma''(T_2) \subseteq \Gamma'(T_2)$ such that $|\Gamma''(T_1)| = |\Gamma''(T_2)| \geq \lfloor \min\{|\Gamma'(T_1)|, |\Gamma'(T_2)|\}/2 \rfloor$ and $\Gamma''(T_1) \cap \Gamma''(T_2) = \emptyset$; this takes from $\Gamma'(T_1)$ and $\Gamma'(T_2)$ two maximally large subsets that are disjoint and of equal size (the worst case is when $\Gamma'(T_1)$ and $\Gamma'(T_2)$ are equal and of odd size).

6. Add to T the vertices from $\Gamma''(T)$, together with the edges connecting them to $\mathrm{Even}(T)$ (not necessarily only to vertices at the largest level of T, as in the bipartite case), for $T = T_1, T_2$.

7. Add the matching edges incident to $\Gamma''(T)$ together with their other endpoints, to T, for $T = T_1, T_2$.

8. Repeat 1.–7.

Like in the bipartite case, it is easy to see that the properties of Lemma 1 are fulfilled. After step 0, T_1 and T_2 are disjoint, and by steps 4 and 5, disjoint sets of vertices are added to T_1 and T_2 in steps 6 and 7, which yields property (a). When the procedure stops in step 2, we have property (b1), if it stops in step 3, we have property (b2).

Let T denote one of the trees at the beginning of a fixed iteration. Assume that $|\mathrm{Even}(T)| < n/\beta$. Then by Lemma 2, applied with $\varepsilon = 2.01$ and $\beta = 6.2$, $|\Gamma_G(\mathrm{Even}(T))| \geq (3 + 9\varepsilon')|\mathrm{Even}(T)|$, where $\varepsilon' = 0.001$. Steps 0 and 5 ensure that at the beginning and end of every iteration, $|\mathrm{Even}(T_1)| = |\mathrm{Even}(T_2)|$, so that $|\mathrm{Even}(T_1)|$, $|\mathrm{Odd}(T_1)|$, $|\mathrm{Even}(T_2)$, $|\mathrm{Odd}(T_2)|$ are all equal, and thus $|T_1 \cup \mathrm{Odd}(T_2)| = |T_2 \cup \mathrm{Odd}(T_1)| = 3|\mathrm{Even}(T)| + 1$. Hence after step 1, $|\Gamma(T)| = |\Gamma_G(\mathrm{Even}(T))| - (3|\mathrm{Even}(T) + 1) \geq 9\varepsilon'|\mathrm{Even}(T)| - 1 \geq 8\varepsilon'|\mathrm{Even}(T)|$, where we assume without loss of generality that $\alpha \geq 1/\varepsilon'$ and hence $\varepsilon'|\mathrm{Even}(T)| \geq \varepsilon'\alpha \log n \geq 1$. Then after step 4, $|\Gamma'(T)| \geq 4\varepsilon'|\mathrm{Even}(T)|$, and since this holds for $T = T_1$ and for $T = T_2$, after step 5, $|\Gamma''(T)| \geq \lfloor 4\varepsilon'|\mathrm{Even}(T)|/2 \rfloor \geq 2\varepsilon'|\mathrm{Even}(T)| - 1 \geq \varepsilon'|\mathrm{Even}(T)|$. In step 7, one vertex per vertex in $\Gamma''(T)$ is added, so that, if T' denotes the tree at the end of the iteration, we have

$$|\mathrm{Even}(T')| = |\mathrm{Even}(T)| + |\Gamma''(T)| \geq |\mathrm{Even}(T)| + \varepsilon'|\mathrm{Even}(T)| = (1 + \varepsilon')|\mathrm{Even}(T)|.$$

This proves that within $O(\log n)$ iterations, either the procedure terminates or at some point $|\mathrm{Even}(T_1)| = |\mathrm{Even}(T_2)| \geq n/\beta$.

As for the bipartite case, once $\mathrm{Even}(T_1)$ and $\mathrm{Even}(T_2)$ contain n/β or more vertices each, by Lemma 3 there will be an edge between the two sets, and the procedure will stop in step 3. This proves that with high probability the procedure terminates within $O(\log n)$ iterations, so that upon termination both trees have depth $O(\log n)$. This finishes the proof of Theorem 1 for general random graphs.

5 Conclusion

We proved that in a random graph on n vertices with high probability every non-maximum matching has an augmenting path of length $O(\log n)$. Motwani could prove this when the average degree is at least $\ln n$, whereas we only require that c is above a certain constant. Our expansion lemma is more powerful than Motwani's and at the same time makes the whole analysis simpler; in fact, the present writeup contains all proofs with all details.

While the expansion property on which the analysis in [Mot94] is built does not hold when c is significantly smaller than $\ln n$, our condition on c does not appear to reflect a principal limit of our analysis. More refined versions of Lemma 2 (an idea would be to consider augmenting path trees which have expansion not in every level but only over a certain constant number of levels, we have not pursued this yet for lack of time), and of Lemma 3 (so far, we have not exploited the special structure of the two large sets between which we need an edge), might well be able to do without any condition on c.

References

[AZ01] M. Aigner and G. M. Ziegler. *Proofs from THE BOOK*. Springer, 2001.

[CGM+98] B. V. Cherkassky, A. V. Goldberg, P. Martin, J. C. Setubal, and J. Stolfi. Augment or push: A computational study of bipartite matching and unit-capacity flow algorithms. *The ACM Journal of Experimental Algorithmics*, 3, 1998. `http://www.jea.acm.org/1998/CherkasskyAugment`.

[CM96] J. Cheriyan and K. Mehlhorn. Algorithms for dense graphs and networks on the random access computer. *Algorithmica*, 15(6):521–549, 1996.

[Edm65a] J. Edmonds. Maximum matching and a polyhedron with $(0, 1)$ vertices. *Journal of Research of the National Bureau of Standards*, 69(b):125–130, 1965.

[Edm65b] J. Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[ER59] P. Erdös and A. Rènyi. On random graphs. *Publ. Math. Debrecen*, 6:290–297, 1959.

[FM95] T. Feder and R. Motwani. Clique partitions, graph compression and speeding-up algorithms. *Journal of Computer and System Sciences (JCSS)*, 51(2):261–272, 1995.

[Gab76] H. N. Gabow. An efficient implementation of Edmond's algorithm for maximum matching on graphs. *Journal of the ACM*, 23:221–234, 1976.

[HK73] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal of Computing*, 2(4):225–231, 1973.

[MN99] K. Mehlhorn and S. Näher. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, 1999.

[Mot94] R. Motwani. Average-case analysis of algorithms for matchings and related problems. *Journal of the ACM*, 41(6):1329–1356, 1994.

[MV80] S. Micali and V. Vazirani. An $O(|V|^{0.5}|E|)$ algorithm for finding maximum matchings in general graphs. In *ACM Symposium on Theory of Computing (STOC)*, pages 17–27, 1980.