Maximum matching in regular and almost regular graphs

Raphael Yuster *

Abstract

We present an $O(n^2 \log n)$-time algorithm that finds a maximum matching in a regular graph with n vertices. More generally, the algorithm runs in $O(rn^2 \log n)$ time if the difference between the maximum degree and the minimum degree is less than r. This running time is faster than applying the fastest known general matching algorithm that runs in $O(\sqrt{n}m)$-time for graphs with m edges, whenever $m = \omega(rn^{1.5} \log n)$.

Keywords: maximum matching, regular graph, algorithm

1 Introduction

A *matching* in a graph is a set of pairwise disjoint edges. A *maximum matching* is a matching of largest possible cardinality. The problem of finding a maximum matching is fundamental in both practical and theoretical computer science, and has numerous applications.

Maximum matching in bipartite graphs is significantly simpler than in general graphs, as computations of augmenting paths do not encounter odd cycles. The seminal algorithm of Hopcroft and Karp [10] solves maximum bipartite matching in $O(\sqrt{n}m)$ time, for graphs with n vertices and m edges. The first polynomial time algorithm for finding a maximum matching in a general graph was obtained by Edmonds [5]. The currently fastest deterministic algorithm for this problem, obtained by Micali and Vazirani, runs in $O(\sqrt{n}m)$ time (see [11, 12, 7]). Faster algorithms are known for several important special classes of graphs.

In this paper we consider the problem of finding a maximum matching in regular and almost regular graphs. Recently, there has been much progress in the *bipartite* version of this problem, and the complexity of the bipartite case is now fairly understood. A simple consequence of Hall's Theorem (see [3]) asserts that a regular bipartite graph has a perfect matching. Now, if the graph is d-regular and d is a power of 2, the following recursive approach due to Gabow and Kariv [6] yields an $O(m)$-time algorithm, which is significantly faster than applying the algorithm of Hopcroft and Karp. First, compute an Euler tour of the graph in time $O(m)$ and then take all the even numbered edges of the tour, to obtain a spanning subgraph of degree $d/2$. Repeating this process eventually

*Department of Mathematics, University of Haifa, Haifa 31905, Israel.
E–mail: raphy@math.haifa.ac.il

gives the desired 1-regular spanning subgraph, i.e. a perfect matching. Extending this idea to the case where d is not a power of 2 is quite difficult. The fastest deterministic algorithm for this problem is due to Cole, Ost, and Schirra [4] and runs in $O(m)$ time. This running time was recently shown to be optimal. It is shown by Goel, Kapralov, and Khanna in [8], that any deterministic algorithm for maximum bipartite matching requires $\Omega(m)$ time (i.e., a deterministic algorithm must, essentially, examine all the edges). In fact, in the same paper, they achieve a breakthrough by presenting a *randomized* algorithm that computes a perfect matching in $O(n \log n)$ expected time, with high probability. This improves an earlier randomized algorithm given in [9].

Unfortunately, these algorithms cease to work for non-bipartite graphs. Presently, the fastest algorithms for regular non-bipartite graphs are obtained using the general algorithms, and require $O(\sqrt{n}m)$ time. The main result of this paper presents a significantly faster algorithm for sufficiently dense graphs. It also applies to graphs that are not necessarily regular, but are not too far from being regular.

For a graph G, the notations $\Delta(G)$ and $\delta(G)$ denote the maximum and minimum degree of G, respectively. A graph G is called *r-almost-regular* if $\Delta(G) - \delta(G) < r$. Thus, regular graphs are 1-almost-regular. Our main result is the following.

Theorem 1.1 *Let G be an r-almost-regular graph with n vertices. There is an algorithm that finds a maximum matching of G in $O(rn^2 \log n)$-time. In particular, if G is a regular graph then a maximum matching of G is found in $O(n^2 \log n)$-time.*

The main idea of the proof is to overcome three obstacles that arise when trying to extend the bipartite approach of [6, 4] to the non-bipartite case. The first problem is that the Euler tour approach causes imbalances when applied to non-bipartite graphs, and these need to be controlled. The second problem is that general regular graphs need not have a perfect matching, and hence maximum matchings computed on their subgraphs need to be considerably augmented. Finally, the number of such augmentations needs to be controlled in such a way that denser subgraphs require less augmentations, as computing augmenting paths requires time that is proportional to the number of edges. The details of the proof appear in the following section. The final section contains some concluding remarks.

2 Proof of the main result

2.1 Almost regular subgraphs of almost regular graphs

For a graph G, let $m(G)$ denote the cardinality of a maximum matching of G. Suppose that G has n vertices and is r-almost-regular. The following lemma establishes a simple lower bound for $m(G)$ in terms of $n, r, \Delta(G)$. It is a consequence of Vizing's Theorem [13] that states that $E(G)$ can be partitioned into at most $\Delta(G) + 1$ matchings.

Lemma 2.1 *If G is an n-vertex graph which is r-almost-regular and $\Delta(G) = d$, then $m(G) \geq (d-r+1)n/(2d+2)$.*

Proof: By the assumption, we have that $2|E(G)| \geq (d-r+1)n$. By Vizing's Theorem, $m(G) \geq |E(G)|/(d+1)$. It follows that $m(G) \geq (d-r+1)n/(2d+2)$. ■

We also need the following lemma which follows from the existence of Euler tours in multigraphs of connected graphs with even degrees.

Lemma 2.2 *Any graph $G = (V, E)$ has a spanning subgraph G' such that $\lceil \Delta(G)/2 \rceil - 1 \leq \Delta(G') \leq \lceil \Delta(G)/2 \rceil$ and $\delta(G') \geq \lfloor \delta(G)/2 \rfloor$ unless G is regular of even degree, in which case $\delta(G') \geq \delta(G)/2 - 1$. Furthermore, G' can be constructed in $O(|V| + |E|)$ time.*

Proof: We can assume that G is connected as otherwise we can prove the result for each connected component separately. Let us first supplement G by adding an arbitrary perfect matching S on the vertices with odd degree. Thus, if there are $2k$ vertices with odd degree, we add k edges such that in the resulting multigraph G^* all the degrees are even. As G^* is connected, it has an Euler tour consisting of edges $e_1, \ldots, e_s$ where e_i precedes e_{i+1} in the tour and $s = |E| + k$. If $k > 0$ we always choose e_1 to be an edge of S. Let $F = \{e_{2i} \ : \ 1 \leq i \leq \lfloor s/2 \rfloor\}$ be the set of edges in even positions of the tour.

The spanning subgraph G' has the edge set $F \cap E = F \setminus S$. As Euler tours can be constructed in linear time, the complexity claim follows. It remains to show that the minimum degree and maximum degree of G' satisfy the claims in the statement. Let x denote the starting vertex of the tour, so that $e_1 = (x, y)$ and $e_s = (z, x)$. Consider first an arbitrary vertex v such that $v \neq x$. Let $d_G(v)$ ($d_{G'}(v)$) be the degree of v in G (resp. G'). If $d_G(v)$ is even, then $d_{G'}(v) = d_G(v)/2$ and we are done. If $d_G(v)$ is odd, then either $d_{G'}(v) = (d_G(v)+1)/2$ or $d_{G'}(v) = (d_G(v)-1)/2$ depending on whether the unique edge of S incident with v is in F or not. Hence, we are done in this case as well. It remains to consider x. If $d_G(x)$ is even, then our assumption on the choice of e_1 implies that $k = 0$ and all the degrees of G are even. In this case we may take x to be any vertex so we may assume that x is a vertex of maximum degree in G. Clearly, $d_{G'}(x) = d_G(x)/2$ or $d_{G'}(x) = d_G(x)/2 - 1$ depending on whether $|E|$ is even or odd. In any case, we are done. Finally, if $d_G(x)$ is odd, then $d_{G'}(x) = (d_G(x)+1)/2$ or $d_{G'}(x) = (d_G(x)-1)/2$ depending on whether s is even or odd (observe that we must have $e_s \in E$ since $e_1 \notin E$ is the unique edge of S incident with x). Hence, we are done in this case as well. ■

Starting with an initial graph $G = G_0$, we can repeatedly apply Lemma 2.2 to obtain a sequence of spanning subgraphs $G_0, \ldots, G_t$ where for $i = 1, \ldots, t$, G_{i-1} and G_i play the roles of G and G' of Lemma 2.2 respectively. More precisely, we obtain the following corollary:

Corollary 2.3 *Let G be an n-vertex graph which is r-almost-regular, $\Delta(G) = d$, and let $t = O(\log d)$ be a positive integer. There is a sequence of n-vertex graphs $G_0, \ldots, G_t$ where $G_0 = G$, having the following properties.*

1. G_i is a spanning subgraph of G_{i-1} for $i = 1, \ldots, t$.

2. $\Delta(G_i) \le \lceil d/2^i \rceil$.

3. G_i is $O(\max\{r/2^i\,, 1\})$-almost-regular.

Furthermore, the sequence can be constructed in $O(nd)$ time.

Proof: The properties follow directly from Lemma 2.2. The complexity claim follows from the fact that G_{i-1} has $O(nd/2^i)$ edges and from the fact that G_i is constructed from G_{i-1} in $O(nd/2^i)$ time. ∎

2.2 Proof of Theorem 1.1

We begin with a short description of the proof of Theorem 1.1. The algorithm starts by constructing the sequence of graphs $G_0, \ldots, G_t$ where $G_0 = G$, having the properties listed in Corollary 2.3. We then gradually compute a maximum matching "bottom-up". We first compute a maximum matching M_t of G_t using a standard matching algorithm for general graphs. However, since G_t is relatively sparse, computing M_t is relatively quick. We then move to G_{t-1} which contains G_t, and hence M_t is a (not necessarily maximum) matching of G_{t-1}. Modifying M_t to a maximum matching M_{t-1} of G_{t-1} requires relatively few augmenting-path steps, that we show how to perform quickly. This process is repeated over and over where we compute a maximum matching M_i of G_i using the matching M_{i+1} as a starting point. Finally, we obtain the desired matching M_0 which is a maximum matching of G. The remainder of this section contains the formal description of the algorithm. We start with two lemmas that state known results that are required for our algorithm.

As noted in the introduction, the presently fastest algorithm for maximum matching in general graphs runs in $O(\sqrt{n}m)$ time (see [11, 12, 7]).

Lemma 2.4 *A maximum matching in a graph with n vertices and m edges can be computed in $O(\sqrt{n}m)$ time.*

Edmond's algorithm, as well as the Micali-Vazirani algorithm [11] and the Goldberg-Tarjan algorithm [7] are augmenting-path based algorithms. Recall that an augmenting path P for a matching M is a path that begins and ends with unmatched vertices, and whose edges alternate between unmatched and matched edges. Clearly, if M is a maximum matching then no augmenting path for M exists. It is also easy to prove the converse; if M is not a maximum matching then an augmenting path exists. Hence, starting with an arbitrary matching, and iteratively computing augmenting paths, one finally ends up with a maximum matching. The algorithms of [11, 7, 2] all show how to compute an augmenting path in linear time.

Lemma 2.5 *An augmenting path for a given matching M (if it exists) in a graph with n vertices and m edges can be computed in $O(m)$ time.*

Proof of Theorem 1.1: Given a graph $G = (V, E)$ with n vertices and m edges, we compute (in linear time) its maximum degree $d = \Delta(G)$ and its minimum degree $\delta(G)$ and hence $r = d - \delta(G) + 1$ is the least integer such that G is r-almost-regular.

We can assume that $m > rn^{1.5}$ (and, in particular, that $r < \sqrt{n}$) as otherwise running the general matching algorithm stated as Lemma 2.4 yields a faster running time than the one stated in Theorem 1.1.

Let t be the least integer such that $d/2^t \le \sqrt{n}$ and observe that $t = O(\log d)$. We construct, in $O(nd)$ time, the sequence of graph $G_0, \ldots, G_t$ with $G_0 = G$, as guaranteed to exist by Corollary 2.3. Let $s_i = m(G_i)$ denote the cardinality of a maximum matching of G_i for $i = 0, \ldots, t$. We next show how to compute, for each G_i, a matching M_i with $|M_i| = s_i$.

A maximum matching M_t of G_t is computed using the algorithm stated as Lemma 2.4. Assume that we have already computed M_i and wish to compute M_{i-1}. As G_i is a (spanning) subgraph of G_{i-1}, we have that M_i is a (not necessarily maximum) matching of G_{i-1}. Using a sequence of $s_{i-1} - s_i$ augmenting path iterations, starting with M_i and applying Lemma 2.5 repeatedly until no further augmenting paths exist, we obtain a maximum matching M_{i-1} of G_{i-1}. Finally, the algorithm returns M_0 which is a maximum matching of G, as required.

It remains to analyze the time complexity of the algorithm, which is dominated by the single application of Lemma 2.4 to obtain M_t, and the repeated applications of Lemma 2.5. For notational clarity, let m_i denote the number of edges of G_i for $i = 0, \ldots, t$, and hence $m = m_0$.

Computing M_t using Lemma 2.4 takes $O(\sqrt{n} m_t)$ time. Since $\Delta(G_t) \le \lceil d/2^t \rceil \le \lceil \sqrt{n} \rceil$, we have that $m_t = O(n^{1.5})$ and hence the time to compute M_t is $O(n^2)$.

The number of applications required to obtain M_{i-1} from M_i using Lemma 2.5 is $s_{i-1} - s_i$ and each application takes $O(m_{i-1})$ time. It therefore remains to prove that

$$\sum_{i=1}^{t} m_{i-1}(s_{i-1} - s_i) = O(rn^2 \log n) \ . \tag{1}$$

As $t = O(\log d)$, it remains to show that each term in the last sum is $O(rn^2)$.

Consider the graph G_i. Denote $d_i = \Delta(G_i)$ and recall that G_i is $O(\max\{r/2^i\ , 1\})$-almost-regular. In particular, it is r_i-almost-regular for $r_i = O(r)$. By Lemma 2.1,

$$s_i = m(G_i) \ge \frac{(d_i - r_i + 1)n}{2(d_i + 1)} = \frac{n}{2} - \frac{r_i n}{2(d_i + 1)} = \frac{n}{2} - O\left(\frac{rn}{d_i}\right) \ .$$

Since $s_{i-1} \le n/2$ we have $s_{i-1} - s_i = O(rn/d_i)$. Also, trivially, $m_{i-1} \le d_{i-1}n/2$. We therefore obtain

$$m_{i-1}(s_{i-1} - s_i) \le O\left(\frac{n}{2} d_{i-1} \cdot \frac{rn}{d_i}\right) = O\left(rn^2 \frac{d_{i-1}}{d_i}\right) = O(rn^2)$$

where in the last equality we have used Lemma 2.1 that states, in particular, that $d_i = \Delta(G_i) \ge \Delta(G_{i-1})/2 - 1 = d_{i-1}/2 - 1$ implying that $d_{i-1}/d_i < 3$.

■

3 Concluding remarks

We presented an $O(rn^2 \log n)$ algorithm for maximum matching in r-almost-regular graphs. The most interesting open problem is whether a faster algorithm exists. Already for the regular case, the result from [8] asserts that any deterministic algorithm requires $\Omega(m)$ time. We note that for the special case of 3-regular bridgeless graphs, an almost linear time algorithm is known. By Petersen's Theorem, such graphs always have a perfect matching, and Biedl et al. [1] show how to find such a matching in $O(n \log^4 n)$ time.

References

[1] T.C. Biedl, P. Bose, E.D. Demaine, and A. Lubiw. Efficient algorithms for petersen's matching theorem. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 130–139. Society for Industrial and Applied Mathematics, 1999.

[2] N. Blum. A new approach to maximum matching in general graphs. In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 586–597, 1990.

[3] B. Bollobás. *Extremal Graph Theory.* Academic Press, 1978.

[4] R. Cole, K. Ost, and S. Schirra. Edge-coloring bipartite multigraphs in $o(elogd)$ time. *Combinatorica*, 21(1):5–12, 2001.

[5] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17(3):449–467, 1965.

[6] H.N. Gabow and O. Kariv. Algorithms for edge coloring bipartite graphs and multigraphs. *SIAM Journal on Computing*, 11:117, 1982.

[7] H.N. Gabow and R.E. Tarjan. Faster scaling algorithms for general graph matching problems. *Journal of the ACM*, 38(4):815–853, 1991.

[8] A. Goel, M. Kapralov, and S. Khanna. Perfect matchings in $o(n \log n)$ time in regular bipartite graphs. In *Proceedings of the 42nd ACM symposium on Theory of computing (STOC)*, pages 39–46. ACM, 2010.

[9] A. Goel, M. Kapralov, and S. Khanna. Perfect matchings via uniform sampling in regular bipartite graphs. *ACM Transactions on Algorithms (TALG)*, 6(2):1–13, 2010.

[10] J.E. Hopcroft and R.M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2:225, 1973.

[11] S. Micali and V. Vazirani. An $o(\sqrt(|v|)|e|)$ algorithm for finding maximum matching in general graphs. In *Proceedings of the 21st annual Symposium on Foundations of Computer Science (FOCS)*, pages 17–27, 1980.

[12] V. Vazirani. A theory of alternating paths and blossoms for proving correctness of the general graph maximum matching algorithm. *Combinatorica*, 14(1):71–109, 1994.

[13] V.G. Vizing. On an estimate of the chromatic class of a p-graph. *Diskret. Analiz*, 3(7):23–30, 1964.