Computing a Maximum Cardinality Matching in a Bipartite Graph in Time $O(n^{1.5}\sqrt{m/\log n})$

by H. ALT[1], N. BLUM[2], K. MEHLHORN[3], M. PAUL[3]

April 26, 2002

Keywords: Analysis of algorithms, bipartite graph, matching

Abstract: We show how to compute a maximum cardinality matching in a bipartite graph of n vertices in time $O(n^{1.5}\sqrt{m/\log n})$. For dense graphs this improves on the $O(\sqrt{n}m)$ algorithm of J.E. Hopcroft and R.M. Karp [HK]. The speed-up is obtained by an application of the fast adjacency matrix scanning technique of J. Cheriyan, T. Hagerup and K. Mehlhorn [CHM].

A *bipartite graph* is an undirected graph $G = (V, E)$ where the vertex set V can be partitioned into disjoint sets V_l and V_r such that every edge $e \in E$ has exactly one endpoint in each of the two sets. A *matching* M is a subset of E such that every vertex is incident to at most one edge in M.

J.E. Hopcroft and R.M. Karp [HK] have shown how to compute a maximum cardinality matching in a bipartite graph in time $O(\sqrt{n}m)$, where $n = |V|$ and $m = |E|$. We give an implementation of their algorithm, which runs in time $O(n^{1.5}\sqrt{m/\log n})$. For dense graphs this is an improvement by a factor of $\sqrt{\log n}$. The speed-up is obtained by the "fast adjacency matrix scanning technique" of J. Cheriyan, T. Hagerup and K. Mehlhorn [CHM].

The algorithm of J.E. Hopcroft and R.M. Karp works in $O(\sqrt{n})$ phases. In each phase, which takes $O(m)$ time, a maximal set (with respect to set inclusion) of shortest augmenting paths is determined by breadth-first and subsequent depth-first search.

An *augmenting path* with respect to a matching M is an alternating path connecting two *free* vertices in V, i.e. vertices which are not incident to an edge in M. An *alternating path* is a path in G which alternately uses edges in M and $E - M$. Interchanging the matching and non-matching edges of an augmenting path increases the cardinality of the matching by one.

[1]FB Mathematik, WE3, Freie Universität Berlin, D-1000 Berlin 33, West Germany
[2]Informatik IV, Universität Bonn, D-5300 Bonn, West Germany
[3]FB Informatik , Universität des Saarlandes, D-6600 Saarbrücken, West Germany

Now we can describe one phase of their algorithm in more detail.

The breadth-first search starts from the free vertices in V_r and constructs a layered directed graph. A vertex $v \in V$ is in layer l iff the shortest alternating path from v to a free vertex in V_r has length l. The edges of the layered digraph are those lying on a shortest alternating path in G. They are directed from the vertex in the higher layer to that in the lower one. In this way, vertices in V_r are put into even layers, vertices in V_l are put into odd layers, matching edges start in even layers, and non-matching edges start in odd layers (cf. Figure **??**). The depth-first search starts in the free vertices in layer L, where L is defined as the minimum layer containing a free vertex in V_l. Whenever a free vertex in layer zero is reached, the edges of the augmenting path are removed from the layered digraph, and the augmenting path is used to increase the current matching.

In our implementation of the algorithm (cf. Figure 1) we combine the breadth-first and the depth-first search following the model of recent network flow algorithms [AO, GT].

Let digraph $G_d = (V, E_d)$ be obtained from G by directing all edges in the current matching from V_r to V_l, and all other edges from V_l to V_r. Thus every path in G_d corresponds to an alternating path in G. For each vertex $v \in V$ we maintain a distance label $layer[v]$. Vertices in V_r occupy even layers, and all free vertices in V_r are in layer zero. Vertices in V_l occupy odd layers, and all free vertices in V_l are in two adjacent layers L and $L+2$.

We maintain the "layered graph invariant" that no edge of the directed graph reaches downwards by two or more layers, i.e.

$$(I_1) \quad \forall (v,w) \in E_d : \qquad layer[v] \quad \leq \quad layer[w] + 1.$$

It follows that $layer[v]$ is a lower bound on the length of an alternating path starting in v and ending in a free vertex in V_r. Call an edge $(v,w) \in E_d$ *eligible*, if $layer[v] = layer[w] + 1$, and let $ce(v)$ be a function which returns an eligible edge starting in v, if there is one, and nil otherwise.

Initially, we put all vertices in V_r into layer zero, all vertices in V_l into layer one, and direct all edges from V_l to V_r. We then search for augmenting paths as follows:

Starting from a free vertex in layer L we construct a path p of eligible edges. Let v be the last vertex of p. There are three cases to distinguish:

Case 1 (breakthrough): v is a free vertex in layer zero:
Then p is an augmenting path with respect to the current matching. We augment the current matching by reversing all edges of p.

Case 2 (advance): v is not a free vertex in layer zero and $ce(v)$ exists:
The path p is extended by adding $endpoint(ce(v))$.

Case 3 (retreat): v is not a free vertex in layer zero and $ce(v) = nil$:
Increase $layer[v]$ by two, and remove v from p.

After a breakthrough or a retreat which leaves us with an empty path p, we start the next search for an augmenting path. If there are no more free vertices

(1) define digraph $G_d = (V_d, E_d)$ by $V_d = V$ and
$E_d = \{(v,w) \in V_l \times V_r;\ \{v,w\} \in E\}$;
(2) **forall** $v \in V$ **do**
  $free[v] \leftarrow 1$;
  $layer[v] \leftarrow \left\{ \begin{array}{ll} 0, & \text{if } v \in V_r \\ 1, & \text{if } v \in V_l \end{array} \right\}$;
od ;
$cardinality \leftarrow 0$;
$L \leftarrow 1$;
(3) **while** $L \leq \sqrt{n}\gamma$ **do** // $\gamma = \sqrt{m \log n}/n$
  (4) **while** $\exists v \in V_l$ with $layer[v] = L$ **and** $free[v] = 1$ **do**
    (5) let $v \in V_l$ be such that $layer[v] = L$ and $free[v] = 1$;
    (6) $p \leftarrow [v]$;
    (7) **while** $p \neq []$ **do**
      (8) $v \leftarrow last\_vertex(p)$;
      (9) **if** $layer[v] = 0$ **and** $free[v] = 1$ **then** // breakthrough
        $cardinality \leftarrow cardinality + 1$;
        $free[first\_vertex(p)] \leftarrow 0$; $free[v] \leftarrow 0$;
        reverse the direction of all edges in p ;
        $p \leftarrow []$;
      else
      (10) **if** $ce(v) \neq nil$ **then** // advance
          $p \leftarrow p + [endpoint(ce(v))]$;
      (11) **else** // retreat
          $layer[v] \leftarrow layer[v] + 2$;
          remove v from p ;
        fi ;
      fi ;
    od ;
  od ;
  (12) $L \leftarrow L + 2$;
(13) **od** ;
(14) find the remaining augmenting paths by the standard method ;

Figure 1: The algorithm

in layer L, we increase L by two and repeat. In this way we proceed until L exceeds $\sqrt{n}\gamma$, where γ is a parameter which we will fix at $\sqrt{m \log n}/n$ later.

Lemma 1: At all times during the execution of the algorithm, the following invariants hold:

(I_1) $\forall (v,w) \in E_d$: $layer[v] \leq layer[w] + 1$;

(I_2) $layer[v]$ is even $\Leftrightarrow$ $v \in V_r$;

(I_3) $p = [v_0, v_1, \ldots, v_l]$ is a path in the current digraph with $layer[v_i] = L - i$, for $0 \leq i \leq l \leq L$ and $free[v_0] = 1$;

(I_4) all vertices $v \in V_l$ with $free[v] = 1$ are in layers L or $L + 2$;

(I_5) the set $M = \{\{v, w\} \in E;\ (w, v) \in (V_r \times V_l) \cap E_d\}$ forms a matching in G with $|M| = cardinality$; furthermore
$free[v] = 1 \quad \text{for } v \in V \qquad \Leftrightarrow \qquad v$ is free with respect to M.

Proof: Certainly all invariants hold initially. Now we do the induction step.

(I_1): Only relabeling a vertex or reversing the direction of an edge may invalidate (I_1).
When a vertex v is relabeled there are no eligible edges out of v, i.e. $layer[v] \leq layer[w] - 1$ for all $(v, w) \in E_d$ by (I_1) and (I_2). Hence increasing $layer[v]$ by two preserves (I_1) for all edges $(v, w) \in E_d$. For edges $(w, v) \in E_d$ the invariant also stays true.
Reversing the edges of the path p in step (9) maintains (I_1) as well, since all edges are eligible.

(I_2): Since layer labels are always increased by two, (I_2) remains true.

(I_3): The path p always starts at a vertex $v \in V_l$ in layer L with $free[v] = 1$ and is only extended by eligible edges.

(I_4): When a vertex is relabeled, it must be on the path p. Thus no free vertex in layer $L + 2$ can be relabeled by (I_3). When L is increased by two, there is no vertex v with $free[v] = 1$ in layer L.

(I_5): In the case of a breakthrough, p is an alternating path from a free vertex $w \in V_l$ to a free vertex $v \in V_r$ by (I_3) and the induction hypothesis, i.e. an augmenting path with respect to the current matching. Thus (9) preserves (I_5).

The correctness of our algorithm is now established. Next we show that it is a derivative of the algorithm of J.E. Hopcroft and R.M. Karp.

Lemma 2: The algorithm in Figure 1 always increases the matching along a shortest augmenting path.
Proof: Let p be the augmenting path of length L found in step (9). (I_4) and (I_5) imply that all free vertices in V_l are in layers L or $L + 2$, and those of V_r are in layer zero. Now the claim follows from (I_1).

Lemma 3: Let M^* be a matching of maximum cardinality in G and M the matching computed by our algorithm after step (13). Then we have $|M| > |M^*| - \sqrt{n}/\gamma$.
Furthermore, step (14) takes time $O(\sqrt{n}m/\gamma)$.
Proof: After step (13) there is no augmenting path with respect to the current matching M of length less than $\gamma\sqrt{n}$. But $M^* \oplus M$ must contain $|M^*| - |M|$ disjoint augmenting paths with respect to M. Thus $(|M^*| - |M|)\gamma\sqrt{n} < n$ and hence $|M| > |M^*| - \sqrt{n}/\gamma$. This implies that further $\sqrt{n}/\gamma$ phases of the standard algorithm with total cost $O(\sqrt{n}m/\gamma)$ suffice to compute a matching of maximum cardinality.

Lemma 4:

a) The total number of increases of layer labels is $O(n^{1.5}\gamma)$.

b) The eligible edge function ce is called $O(n^{1.5}\gamma)$ times.

Proof:

a): (I_4) and step (3) imply that the maximum layer of a vertex during an execution of the algorithm is $\sqrt{n}\gamma + 2$. Thus we execute step (11) at most $O(\sqrt{n}\gamma)$ times for each vertex.

b): Each time the function ce returns an eligible edge $(v, w) \in E_d$, we extend the current path p by this edge. Either it still belongs to the path when p becomes augmenting for the next time, or $layer[w]$ is increased by two when (v, w) is deleted from p. Thus the number of calls to the function ce is bounded by twice the total number of increases of layer labels plus the total length of all augmenting paths found during the execution of steps (1)–(13). Since the length of an augmenting path is at most $\sqrt{n}\gamma$, because of (I_4) and step (3), and since there are at most n of them, part a) of this Lemma completes the proof.

We infer from Lemma 4 that the total time spent in steps (1)–(13) is $O(n^{1.5}\gamma)$ plus the time spent in calls to the current edge function.

In [CHM] it has been shown how to implement this function such that the time spent in calls $ce(v)$ between relabelings of v is $O(n/\log n + \text{number of calls})$. We conclude that the total time spent in steps (1)–(13) is $O(\sqrt{n}\gamma \cdot n \cdot n/\log n + n^{1.5}\gamma) = O(n^{2.5}\gamma/\log n)$.

We summarize in

Theorem 1: A maximum cardinality matching in a bipartite graph with n vertices and m edges can be computed in time

$$O(\min\{\sqrt{n}m, n^{1.5}\sqrt{m/\log n}\}) \quad = \quad O(n^{2.5}/\sqrt{\log n}).$$

Proof: The running time of the standard algorithm is $O(\sqrt{n}m)$, and the running time of our algorithm is $(O(n^{2.5}\gamma/\log n + \sqrt{n}m/\gamma)$ for any $\gamma > 0$. The bound follows with $\gamma = \sqrt{m \log n}/n$.

References

[AO] Ahuja, R.K., J.B. Orlin. *A Fast and Simple Algorithm for the Maximum Flow Problem.* Operation Research, Vol. 37 (1989), 748–759

[CHM] Cheriyan, J., T. Hagerup, K. Mehlhorn. *Can a Maximum Flow be Computed in $O(nm)$ Time?* To be presented at 17^{th} ICALP, 1990

[GT] Goldberg, A.V., R.E. Tarjan. *A New Approach to the Maximum Flow Problem.* J. ACM 35 (1988), 921–940.

[HK] Hopcroft, J.E., R.M. Karp. *An $n^{2.5}$ Algorithm for Maximum Matchings in Bipartite Graphs.* SIAM J. Comp. 2, No. 4 (1973), 225–231