A New Approach to Maximum Matching in General Graphs

Extended Abstract

Norbert Blum
Informatik IV, Universität Bonn
Römerstr. 164, D-5300 Bonn, West Germany

Abstract

We reduce the problem of finding an augmenting path in a general graph to a reachability problem and show that a slight modification of depth-first search leads to an algorithm for finding such paths. As a consequence, we obtain a straightforward algorithm for maximum matching in general graphs of time complexity $O(\sqrt{n}m)$, where n is the number of nodes and m is the number of edges in the graph.

1 Introduction and motivation

Although since Berge's theorem in 1957 [4] it is well known that for constructing a maximum matching it suffices to search for augmenting paths, until 1965 only exponential algorithms for finding a maximum matching in general graphs were known. The reason was that one did not know how to handle with odd cylces in alternating paths.

In his fundamental paper [6] Edmonds solved this problem by shrinking these odd cylces. The straightforward implementation of his approach led to an $O(n^4)$ algorithm for maximum matching in general graphs. In the subsequence more sophisticated implementations of his approach led to $O(n^3)$ or $O(nm)$ algorithms [2, 8, 14, 20].

In 1973 Hopcroft and Karp [12] proved the following fact. If one computes in one phase a maximal set of shortest augmenting paths, then $O(\sqrt{n})$ such phases would be sufficient. For the bipartite case they showed that a phase can be implemented by a breath-first search followed by a depth-first search. This led to an $O(n+m)$ implementation of one phase and hence to an $O(\sqrt{n}m)$ algorithm for maximum matching in bipartite graphs.

In 1975 Even and Kariv [7, 13] presented a $\min(n^2, m\log n)$ implementation of a phase leading to an $O(\min(n^{2.5}, \sqrt{n}m\log n))$ algorithm for maximum matching in general graphs. Galil [9] called the full paper [13] "a strong contender for the ACM Longest Paper Award." Tarjan [18] called their paper "a remarkable tour-de-force." In 1978 Bartnik [3] has given an alternative $O(n^2)$ implementation in his unpublished Ph.D. thesis (see [11]).

In 1980 Micali and Vijay Vazirani [15] claimed to have an $O(m)$ implementation of a phase without the presentation of a proof of correctness. Although their result is cited in many papers and also in some textbooks no proof of correctness was available. Also the paper of Peterson and Loui [17] does not clarify the situation, since their paper contains only an "informally proof of correctness" which cannot be accepted as correctness proof. Possibly this shortcoming is recently repaired in [19].

The first reason, why I attempted a new approach to maximum matching in general graphs was from a didactical point of view. When I gived lectures on algorithms for maximum matching problems I asked myself having the beautiful well-known approach for the bipartite case in mind, which first reduces the problem of finding an augmenting path to a reachability problem in a directed graph and then solves this problem by depth-first search: Why also in the general case, we cannot reduce the problem of finding an augmenting path to a reachability problem in a directed graph and then solve this problem by something like depth-first search? I hoped then to get an approach for which it is easier to get any intuition than in the known approach. Also I hoped to get simpler algorithms.

Secondly I believed that if I have sucess with this attempt, I get a straightforward $O(m)$ implementation of a phase by something like a breath-first search followed by the new algorithm.

Indeed we get simple and efficient algorithms for maximum matching in general graphs. Only the proof of correctness are more involved.

In chapter 2 definitions and the general method are given. In chapter 3 the reduction to a reachability problem in a directed graph is given. This reachability problem is solved in chapter 4. The correctness proof is sketched in chapter 5. In chapter 6 we outline how to implement the solution effciently. We explain how the new approach leads to a straightforward implementation of a phase using only $O(m)$ time in chapter 7.

2 Definitions and the general method

A graph $G = (V, E)$ consists of a finite nonempty set of *nodes* V and a set of *edges* E. Either G is *directed* or *undirected*. In the (un-)directed case each edge is an (un-)ordered pair of distinct nodes.

A graph $G = (V, E)$ is *bipartite*, if V can be partitioned into disjoint nonempty sets A and B, such that for all $(u, v) \in E$ it holds $u \in A$ and $v \in B$ or vice versa. We then write often $G = (A, B, E)$. A *path* P from $v \in V$ to $w \in W$ is a sequence of nodes $v = v_0, v_1, v_2, \ldots, v_k = w$ satisfying $(v_i, v_{i+1}) \in E$ for $0 \leq i \leq k$.

The *length* $|P|$ of P is the number k of edges on P. P is *simple* if $v_i \neq v_j$ for $0 \leq i < j \leq k$. By abuse of notation, P will denote the path $v_0, v_1, \ldots, v_k$, the set of nodes $\{v_0, v_1, \ldots, v_k\}$ and also the set of edges $\{(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)\}$.

If there exits a path from v to w (of length 1) v is called *(direct) predecessor* of w and w is called *(direct) successor* of v.

Let $G = (V, E)$ be an undirected graph. $M \subseteq E$ is called a *matching* of G if

$$\forall e = (u, v), e' = (x, y) \in M, e \neq e' \Rightarrow \{u, v\} \cap \{x, y\} = \emptyset$$

i.e. no two edges in M have a common node.

A matching M is *maximal* if there exists no $e \in E \setminus M$, such that $M \cup \{e\}$ is a matching. A matching M is *maximum* if there exists no matching $M' \subseteq E$ such that $|M'| > |M|$. Given an undirected graph $G = (V, E)$ the *maximum matching problem* is finding a maximum matching $M \subseteq E$.

A path $P = v_0, v_1, \ldots v_k$ is *M-alternating*, if

$$(v_i, v_{i+1}) \in M \iff (v_{i+1}, v_{i+2}) \in E \setminus M, \quad 0 \leq i \leq k - 2\,.$$

A node $v \in V$ is called *M-free* if $\nexists u \in V : (v, u) \in M$. Let $P = v_0, v_1, \ldots, v_k$ be a simple M-alternating path. P is called *M-augmenting* if v_0 and v_k are M-free.

Let P be an M-augmenting path in G. Then $M \oplus P$ denotes the *symmetric difference* of M and P, i.e. $M \oplus P = M \setminus P \cup P \setminus M$. It is easy to see that $M \oplus P$ is a matching of G and $|M \oplus P| = |M| + 1$.

The key of most algorithms for finding maximum matching in a graph is the following theorem of Berge [4].

Theorem 1 *Let $G = (V, E)$ be an undirected graph and $M \subseteq E$ be a matching. Then M is maximum if and only if there exists no M-augmenting path in G.*

The theorem of Berge implies directly the following general method for finding a maximum matching in a graph G.

```
Algorithm 1
Input: undirected graph $G = (V, E)$
       matching $M \subseteq E$ (possibly $M = \emptyset$)
Output: a maximum matching $M_{\max}$
Method:

while there exists an $M$-augmenting path do
    begin
    construct such a path $P_i$;
    $M := M \oplus P$
    end
```

The key problem is now: How to find an M-augmenting path P, if such exists?

We solve this key problem in the following way:

1. We reduce the key problem to a reachable problem in a directed graph $G_M = (V', E')$
2. We solve this reachability problem constructively.

3 Reduction to a reachability problem

In the bipartite case we construct from $G = (A, B, E)$ and matching $M \subseteq E$ a directed graph $G_M = (V', E')$ by directing the edges in M from A to B and the edges in $E \setminus M$ from B to A. Additionally we add two nodes s and t to $A \cup B$ and add for each M-free node $b \in B$ the edge (s, b) to E' and for each M-free node $a \in A$ the edge (a, t) to E'.

It is easy to prove that there exists an M-augmenting path in G if and only if there exists a simple path from s to t in G_M.

This reachability problem can be solved by a depth-first search (DFS) of G_M with start node s.

Let $G = (V, E)$ be an undirected graph and $M \subseteq E$ be a matching. Let $V_M = \{x \in V \mid x \text{ is } M\text{-free}\}$. For the definition of G_M we have the following difficulty.

A priori we cannot divide the set of nodes V into two sets A and B such that the following holds. There exists an M-augmenting path in G if and only if there exists an M-augmenting path, which uses alternately nodes from A and B. Hence for defining G_M, we introduce for each node $v \in V$ two nodes $[v, A]$ and $[v, B]$ such that a contruction of a graph G_M, analogously to the bipartite case, is possible.

Let $G_M = (V', E')$ where

$$\begin{aligned} V' &= \{[v,A],[v,B] | v \in V\} \cup \{s,t\} \qquad s,t \notin V,\ s \neq t \\ E' &= \{([v,A],[w,B]),([w,A],[v,B]) | (v,w) \in M\} \\ &\quad \cup \{([x,B],[y,A]),([y,B],[x,A]) | (x,y) \in E \setminus M\} \\ &\quad \cup \{(s,[v,B]) | v \in V_M\} \cup \{([x,A],t) | x \in V_M\} \end{aligned}$$

Analogously to the bipartite case, we have directed the edges in M "from A to B" and the edges in $E \setminus M$ "from B to A". Since the distinct nodes $[v, A]$ and $[v, B]$ in V' correspond to the same node v in V, it does not suffice to find a simple path from s to t in G_M for finding an M-augmenting path in G. Hence we define strongly simple paths in G_M.

Let P be a path in G_M. P is *strongly simple* if the following hold

a) P is simple.

b) $\forall [v,A] \in V' : [v,A] \in P \Rightarrow [v,B] \notin P$.

Now we can formulate the reachability problem, which is aquivalent to the problem of finding an M-augmenting path.

Theorem 2 *Let $G = (V, E)$ be an undirected graph, $M \subseteq E$ be a matching and $G_M = (V', E')$ defined as above. Then there exists an M-augmenting path in G if and only if there exists a strongly simple path from s to t in G_M.*

Proof:
"$\Leftarrow$": Let $P = s, [v_1, B], [v_2, A], [v_3, B], \ldots, [v_{k-1}, B], [v_k, A], t$ be a strongly simple path in G_M. Then $v_i \neq v_j$, $1 \leq i < j \leq k$ and $v_1, v_k \in V_M$. Hence

$$P' = v_1, v_2, \ldots, v_k$$

is an M-augmenting path in G.
"$\Rightarrow$": Let $Q = w_1, w_2, \ldots, w_{l-1}, w_l$ be an M-augmenting path in G. Then $w_i \neq w_j$, $1 \leq i < j \leq l$ and $w_1, w_l \in V_M$. Hence by construction of G_M

$$Q' = s, [w_1, B], [w_2, A], \ldots, [w_{l-1}, B], [w_l, A], t$$

is a strongly simple path in G_M. ∎

4 The solution of the reachability problem

Depth-first search finds simple paths in a directed graph. Hence we cannot use DFS directly for the solution of the reachability problem in G_M. We modify the usual DFS such that the modified depth-first search (MDFS) finds exactly the strongly simple paths in G_M.

Let $[v, \overline{A}] = [v, B]$ and $[v, \overline{B}] = [v, A]$. Remember that DFS partitions the edges of the graph into four categories [1]. Similiary the edges of G_M are partioned into five categories by a MDFS of G_M:

1. *Tree edges*, which are edges leading to new nodes $[v, X]$, $X \in \{A, B\}$, for which $[x, \overline{X}]$ is not a predecessor during the search.

2. *Weak back edges*, which are edges leading to new nodes $[v,A]$, for which $[v,B]$ is not a predecessor during the search.
3. *Back edges*, which go from descendants to ancestors.
4. *Forward edges*, which go from ancestors to proper descendants but are neither tree edges nor weak back edges.
5. *Cross edges*, which go between nodes what are neither ancestors nor descendants of one another.

Like DFS, MDFS uses a stack K for the organisation of the search. Analogously to DFS, the MDFS-stack K defines a tree, the *MDFS-tree* T. Before describing MDFS in detail, we describe the algorithm informally. TOP(K) denotes the last node added to the MDFS-stack K. In each step MDFS considers an edge $(\text{TOP}(K),[w,Y])$ which was not considered previously.

Let $e=([v,X],[w,\overline{X}])$ be the edge considered. We distinguish two cases:

case 1: $X=A$, i.e. $(v,w)\in M$.

case 2: $X=B$, i.e. $(v,w)\in E\setminus M$

2.1 $[w,A]\in K$

2.2 $[w,A]\notin K$ but $[w,B]\in K$

2.3 $[w,A]\notin K$ and $[w.B]\notin K$

i) $[w,A]$ was in K previously

ii) $[w,A]$ was not in K previously

Note that these are all cases. Only in the cases 2.2 and 2.3 i) MDFS differs from DFS. We discuss both cases.

case 2.2 If $[w,A]$ was not in K before, DFS would perform the operation PUSH($[w,A]$), since $[w,A]\neq[w,B]$. Since $[w,B]\in K$ and MDFS should only construct strongly simple paths in G_M, MDFS does not perform the operation PUSH($[w,A]$).

case 2.3 i) Since $[w,A]$ was in K before, DFS would not perform any PUSH-operation. But the different treatment of case 2.2 can cause the following situation for MDFS. MDFS has found a strongly simple path from $[w,A]$ to a node $[u,A]$. But the node $[u,B]$ was in K and hence, by case 2.2, the operation PUSH($[w,A]$) was not performed. But now, $[u,B]\notin K$. As we will prove later, the paths P from s to $[v,B]$ and Q from $[w,A]$ to $[u,A]$ are *strongly disjoint*, i.e. there is no $[r,X]$ in P such that $[r,Y]\in Q$, for $X,Y\in\{A,B\}$. Hence the path P,Q is strongly simple. Since MDFS has found a strongly simple path from s to $[u,A]$, MDFS performs the operation PUSH($[u,A]$).

Next we describe MDFS more in detail. We have to solve the following problem: How does MDFS find the node $[u,A]$ in case 2.3 i)?

For solving this problem assume that MDFS is organized such that for all nodes $[w,A]\in V'$, the following hold.

After performing the operation POP($[w,A]$) MDFS has always computed a set $L_{[w,A]}$ of nodes such that: $L_{[w,A]}$ contains exactly the nodes $[u,A]\in V'$ satisfying:

1. MDFS has found a path from $[w,A]$ to $[u,A]$, not containing $[u,B]$.
2. PUSH($[u,A]$) is not performed.
3. POP($[u,B]$) is performed.

Note that MDFS has found a path Q from $[w,A]$ to $[u,A]$ if and only if for all $[v,X]\in Q$, MDFS has performed the operation PUSH($[v,X]$) and all edges on Q are considered.

In the description of MDFS we assume for all $[w,A]\in V'$, that $L_{[w,A]}$ is computed correctly. The computation of $L_{[w,A]}$, as well as an efficient implementation of MDFS can be found in section 6.

Algorithm 2 (MDFS)

Input: $G_M=(V',E')$
Output: *An M-augmenting path P, if such exists.*
Method:

```
begin
PUSH(s);
while $K \neq \emptyset$ and no path from s to t is found do
    SEARCH(TOP(K))
end.
```

SEARCH is a call of the following procedure. Let $h \in V'$.

```
            procedure SEARCH($h$);
            begin
            if $h = t$ then
                contruct the $M$-augmenting path $P$ which is found by the algorithm
            else
                begin
                mark $h$ "pushed"
                for all nodes $[w, Y] \in N[h]$ do
                    begin
(case 1)            if $Y = B$ then
                        begin
                        PUSH($[w, B]$)
                        SEARCH($[w, B]$)
                        end
(case 2)            else
(case 2.1)              if $[w, A] \in K$ then
                            no PUSH-operation is performed
                        else
(case 2.2)                  if $[w, B] \in K$ then
                                no PUSH-operation is performed
(case 2.3)                  else
(case 2.3 i)                    if $[w, A]$ is marked "pushed" then
                                    while $L_{[w,A]} \neq \emptyset$ do
                                        begin
                                        choose any $[u, A] \in L_{[w,A]}$;
                                        PUSH($[u, A]$);
                                        SEARCH($[u, A]$)
                                        end
(case 2.3 ii)                   else
                                    begin
                                    PUSH($[w, A]$);
                                    SEARCH($[w, A]$)
                                    end
                    end
                POP
                end
            end.
```

5 The correctness proof of the algorithm MDFS

The correctness proof of MDFS is inspired by the correctness proof of DFS. But the proof is much more involved. First we prove two fundamental lemmas. The first lemma shows that MDFS finds a path from s to a node, if a strongly simple path exists. With help of the second lemma we shall prove that MDFS finds only strongly simple paths.

Lemma 1 *Let $[w, B] \in V'$ be a node, for which MDFS performs the operation PUSH($[w, B]$). Let $[x, A] \in V'$ such that there exists a strongly simple path $P = [w, B] = [v_0', B], [v_1, A], [v_1', B], \ldots, [v_{t-1}', B], [v_t, A] = [x, A]$ with the following property: In the moment when PUSH($[w, B]$) is performed, it holds $[v, X], [x, \overline{X}] \notin K$, for all $[v, X] \in P$. Then PUSH($[x, A]$) is performed before the performance of the operation POP($[w, B]$).*

Remark: Lemma 1 implies that either PUSH($[x, A]$) and also POP($[x, A]$) are performed before the performance of PUSH($[w, B]$) or both operations are performed between the operations PUSH($[w, B]$) and POP($[w, B]$).

Proof: (by induction on the length t of the path P).
See the full paper [5]. ∎

Lemma 2 *Let $[w, X], [u, B] \in V'$ and assume that MDFS performs the operation PUSH($[u, B]$). If there exists a strongly simple path $P = [v_1, A], [v_2, B], \ldots, [v_j, X], [v_{j+1}, \overline{X}], \ldots, [v_{r-1}, A], [v_r, B]$ such that*

1. *$([u, B], [v_1, A]), ([v_r, B], [u, A]) \in E'$*
2. *$[v_j, X] = [w, X]$*
3. *$[v_i, A]$ and $[v_i, B]$ are not predecessors of $[u, B]$ in the MDFS-tree T for $1 \leq i \leq r$.*

Then PUSH($[w, \overline{X}]$) is performed before POP($[u, B]$).

Proof: Consider the following path $P' = [u, B], [v_r, A], [v_{r-1}, B], \ldots, [v_j, \overline{X}]$. Then P' fulfills the assumptions to lemma 1. Hence the assertion follows directly from lemma 1. ∎
Now we can prove the correctness of the algorithm MDFS.

Theorem 3

a) *MDFS finds a path from s to t, if a strongly simple path from s to t exists.*

b) *MDFS finds only strongly simple paths.*

Proof:

a) Assume that MDFS does not find a path from s to t.
Let $P = s, [v_0', B], [v_1, A], [v_1', B], \ldots, [v_{r-1}', B], [v_r, A], t$ be a strongly simple path from s to t. It is clear that MDFS considers the edge $(s, [v_0', B])$ and performs the operation PUSH($[v_0', B]$) after the consideration of the edge $(s, [v_0', B])$ (note that v_0' is M-free). Hence $[v_0', B]$ and $[v_r, A]$ fulfill the assumptions of lemma 1 with respect to the path $[v_0', B], [v_1, A], \ldots, [v_{r-1}', B], [v_r, A]$. Hence by Lemma 1, MDFS performs the operation PUSH($[v_r, A]$) and hence also PUSH(t).

b) We prove by induction on the number of PUSH-operations that no PUSH-operation destroys the property "strongly simple".
See the full paper [5] ∎

6 An implementation of the algorithm MDFS

Now we sketch how to implement the algorithm MDFS efficiently. For a detailed description see [5]. Only the following two parts of the algorithm are nontrivial to implement.

1. The manipulation of the sets $L_{[w,A]}$, $[w, A] \in V'$.
2. The reconstruction of the M-augmenting path P, which is found by the algorithm.

For the solution of both subproblems it is useful to perform the POP-operations not explicitely and to maintain the whole MDFS-tree T.
Since we are interested on efficient solutions of our subproblems, it is useful to investigate the properties of MDFS.

Lemma 3 *For all $[w, A] \in V'$, the following always holds: $|L_{[w,A]}| \leq 1$.*

Proof: see [5] ∎

In the following we write $L_{[w,A]} = [p, A]$ instead of $L_{[w,A]} = \{[p, A]\}$. The following lemma is the key for an efficient implementation of our method.

Lemma 4 *Assume that the algorithm performs the assignment $L_{[w,A]} := [u,A]$. Then after the performance of PUSH([u,A])) always $L_{[w,A]} = L_{[u,A]}$ is fulfilled.*

Proof: see [5] ∎

Now we describe, how to update the sets $L_{[w,A]}$. By the definition of $L_{[w,A]}$, we have only to change $L_{[w,A]}$ after a PUSH- or after a POP-operation. More exactly, we have to perform:

After PUSH([u,A]) $L_{[w,A]} := \emptyset$, if $L_{[w,A]} = [u,A]$

After POP([u,B]) $L_{[w,A]} := [u,A]$, if MDFS has found a path from $[w,A]$ to $[u,A]$ not containing $[u,B]$, and PUSH($[u,A]$) is not performed previously.

Hence after the performance of the operation POP($[u,B]$), the algorithm MDFS has to find all nodes $[w,A]$, for which MDFS has found a path Q to $[u,A]$ previously.

For $[q,A] \in V'$, for which the operation PUSH($[q,A]$) is performed, we define

$$D_{[q,A]} = \{[p,A] \in V' \mid L_{[p,A]} = [q,A] \text{in the moment when the operation PUSH}([q,A]) \text{ is performed}\} .$$

By lemma 4 for all $[p,A] \in D_{[q,A]}$, the computation of $L_{[p,A]}$ can be reduced to the computation of $L_{[q,A]}$. Hence under the assumption that the sets $D_{[q,A]}$ are given, the following problem remains. After the performance of the operation POP($[u,B]$), we have to find all nodes $[w,A]$, for which $L_{[w,A]}$ was not $\neq \emptyset$ previously and for which MDFS has found a path Q to $[u,A]$ previously.

Let us consider such a path $Q = [w,A] = [v_0,A],[v_1,B],[v_2,A],\ldots,[v_{t-1},B],[v_t,A] = [u,A]$, found by MDFS. Let e_i, $1 \leq i \leq t$ be the edges of Q. Let us characterize the path Q backwards.

By construction, it is clear that e_t is a weak back edge. Then we have a sequence of tree edges, a single cross, forward or back edge, a sequence of tree edges and so on. Note that all edges which correspond to an edge in the actual matching, are tree edges. Hence Q can considered as a sequence of blocks of tree edges, which are separated by single cross, forward or back edges. The last block is concluded by the weak back edge $e_t = ([v_{t-1},B],[v_t,A])$.

Let $L = \{[p,A] \in V' \mid L_{[p,A]} \neq \emptyset \text{ previously}\}$.

A simple way to compute the nodes $[w,A]$ with $[w,A] \notin L$ and for which we have to perform the assignment $L_{[w,A]} := [u,A]$, is by constructing the paths Q backwards.

For doing this, first we construct all such paths Q *without any* cross, forward or back edge. This is done by following backward a weak back edge and then the tree edges until a node in $L \cup \{[u,B]\}$ is reached. Then we construct all paths Q with *exactly one* cross, forward or back edge. For doing this, we follow backwards a cross, forward or back edge $([v,B],[w,A])$, such that we have found a path from $[w,A]$ to $[u,A]$ before and then tree edges until a node in $L \cup \{[u,B]\}$ is reached. Then we construct all paths Q with *exactly two* cross, forward or back edge, and so on.

The considerations above lead to an $O(n+m)$ implementation of the manipulation of the set $L_{[w,A]}$ (see [5]). The M-augmenting path P found by MDFS can be reconstructed in time $O(|P|)$ [5]. Altogether we have obtained the following theorem.

Theorem 4 *MDFS can be implemented such that it uses only linear time and linear space in $(n+m)$.*

7 An $O(\sqrt{n}m)$ algorithm for maximum matching in general graphs

In their excellent paper Hopcroft and Karp [12] present the following algorithm for the computation of a maximum matching.

"Algorithm 3 (Maximum matching algorithm)

Step 0 $M \leftarrow \emptyset$

Step 1 Let $l(M)$ be the length of a shortest augmenting path relative to M. Find a maximal set of paths $\{Q_1, Q_2, \ldots, Q_t\}$ with the properties that

1. for each i, Q_i is an augmenting path relative to M and $|Q_i| = l(M)$;
2. the Q_i are vertex-disjoint.
   Halt if no such paths exist.

Step 2 $M \leftarrow M \oplus Q_1 \oplus Q_2 \oplus \cdots \oplus Q_t$; go to 1"

They proved the following theorem.

Theorem 5 *"If the cardinality of a maximum matching is s, then Algorithm 3 constructs a maximum matching within* $2\lfloor\sqrt{s}\rfloor + 2$ *executions of Step 1."*

They concluded

> "This way of describing the construction of a maximum matching suggests that we should not regard successive augmenting steps as independent computations, but should concentrate instead on the efficient implemenatation of an entire phase (i.e., the execution of Step 1 in Algorithm 3)."

In the bipartite case, they describe an elegant and also simple implementation of an entire phase, which has time complexity $O(m)$. Hence they have obtained an $O(\sqrt{n}m)$ algorithm for maximum matching in bipartite graphs.

Let us sketch the implementation of Hopcroft and Karp. First they reduce the problem of finding augmenting paths to the reachability problem as described in section 3.

Then by performing a breath-first search (BFS) on G_M with start node s until the goal node t is reached, they obtained a layered directed graph $\overline{G}_M$, for which the paths from s to t correspond exactly to the shortest M-augmenting paths in G. Using depth-first search they find a maximal set of disjoint M-augmenting paths. Whenever an M-augmenting path is found, the path and all incident edges are deleted and the depth-first search is continued. Since breath-first search and also depth-first search takes only $O(m)$ time, it is easy to see that the implementation of Hopcroft and Karp has time complexity $O(m)$.

Since M-augmenting paths can be found in general graphs by a slightly modified depth-first search (MDFS), the following question suggests itself. In the general case can we get an implementation of an entire phase by performing a breath-first search followed by a modified depth-first search?

We cannot give an affirmative answer to this question. But we shall show that we can give an affirmative answer, if we replace breath-first search by a slight modification of breath-first search.

Let $G = (V, E)$ be graph, M be a matching of G and $G_M = (V', E')$ be the directed graph as defined in section 3.

Our goal is to construct from G_M a layered directed graph $\overline{G}_M = (V', \overline{E})$ such that

1. the l-th layer contains exactly the nodes $[v, X] \in V'$ with the property that the shortest strongly simple path from s to $[v, X]$ in G_M has length l.
2. $\overline{G}_M$ contains all shortest strongly simple paths from s to t in G_M.

It is clear that s is the only node in the layer with number zero i.e., level(s) $= 0$. Note that by the structure of G_M $X = B$ ($X = A$) implies that level($[v, X]$) is odd (even).

Since breath-first search on G_M with start node s finds shortest simple distances form s and not shortest strongly simple distances, we cannot apply BFS directly for the construction of $\overline{G}_M$. But by applying similary methods as in the case of DFS, we can modify BFS such that the modified breath-first search (MBFS) finds the shortest strongly simple distances.

Remember that for the construction of the $(l+1)$-th level BFS needs only to consider the nodes in the l-th level and to insert all nodes w into the $(l+1)$-th level which fulfill the following properties.

1. There exists a node v in the l-th level with $(v, w) \in E$.
2. Level(w) is not defined before.

In the case of finding strongly simple distances from s, the construction of the $(l+1)$-th is a little bit harder.

Note that by structure of G_M, the level of a non-free node $[w, B]$ is well-defined by the level of the unique node $[v, A]$ with $([v, A], [w, B]) \in E'$. Let us consider how MBFS constructs the $(l+1)$-th level where $(l+1)$ is even under the assumption that the levels $0, 1, 2, \ldots, l$ are constructed previously.

It is clear that similary to BFS MBFS can insert all nodes $[w, A] \in V'$ into the $(l+1)$-th level which fulfill the following properties.

1. There exists a node $[v, B]$ in the l-th level with $([v, B], [w, A]) \in E'$ and the property that there exists a strongly simple path from s to $[v, B]$ of length l which does not contain $[w, B]$.
2. Level($[w, B]$) is not defined before.

$$\begin{tikzcd}
{[v_5,B]} \arrow[rr] & & {[v_6,A]} \arrow[d] & {[v_{17},B]} \arrow[l] \\
{[v_4,A]} \arrow[u] & & {[v_7,B]} \arrow[d] & {[v_{16},A]} \arrow[u] \\
& {[v_3,B]} \arrow[ul] & {[v_3,A]} & \vdots \arrow[u] \\
& {[v_2,A]} \arrow[u] & & {[v_9,B]} \arrow[u] \\
& {[v_1,B]} \arrow[u] \arrow[rr] & & {[v_8,A]} \arrow[u] \\
& s \arrow[u, dashed] & &
\end{tikzcd}$$

$\longrightarrow$ matched edge
$\longrightarrow$ free edge
$--\rightarrow$ additional edge

But these are not all nodes, which MBFS has to insert into the $(l+1)$-th level. Consider the following example. Note that although level$([v_7,B]) = 7$ it holds level$([v_3,A]) \neq 8$ since the unique shortest strongly simple path from s to $[v_7,B]$ contains $[v_3,B]$. The only strongly simple path P from s to $[v_3,A]$ has length 14. Hence level$([v_3,A]) = 14$. Note that P is found when edge $([v_{17},B],[v_6,A])$ is considered.

By the observation above MBFS has also to insert nodes $[w,A] \in V'$ into $(l+1)$-th level, for which there exists a shortest strongly simple path $P = s,[v_1,B],[v_2,A],\ldots,[v_l,B],[w,A]$ with level$([v_l,B]) < l$.

Similary to breath-first search MBFS considers after the construction of the l-th level, l even, all edges $([v,B],[w,A])$ such that level$([v,B]) = l$. Let $([v,B],[w,A])$ be the edge considered. We distinguish three cases.

case 1: level$([w,A]) > l$ and there exists a strongly simple path P from s to $[v,B]$ such that $|P| = l$ and $[w,B] \notin P$.

case 2: level$([w,A]) > l$ and for all strongly simple paths P from s to $[v,B]$ with $|P| = l$ hold $[w,B] \notin P$.

case 3: level$([w,A]) \leq l$.

Note that these are all cases. Next we discuss these three cases.

case 1: Since there is a strongly simple path from s to $[w,A]$ of length $l+1$ which uses the edge $([v,B],[w,A])$, MBFS insert $[w,A]$ into the $(l+1)$-th level and add the edge $([v,B],[w,A])$ to $\overline{E}$.

case 2: Since all strongly simple path from s to $[v,B]$ of length l contain $[w,B]$ no strongly simple path from s to $[w,A]$ of length $(l+1)$ using the edge $([v,B],[w,A])$ exists.

case 3: level$([w,A])$ is defined earlier. But it is possible that the edge $([v,B],[w,A])$ is contained in a shortest strongly simple path to a node $[u,A]$ if the following properties are fulfilled.

1. All shortest strongly simple paths from s to $[w,A]$ contain $[u,B]$.
2. There exists a strongly simple path from s to $[v,B]$ which does not contain $[u,B]$.
3. There is a strongly simple path from $[w,A]$ to $[u,A]$.

Note that for the correct treatment of case 1 and case 2, MBFS has to know if there exists a shortest strongly simple path P from s to $[v,B]$ which does not contain th node $[w,B]$.

For all nodes $[v,X] \in V'$, such that level$([v,X])$ is defined, we denote by $\mathrm{DOM}_{[v,X]}$ the node $[u,B] \in V'$ which satisfies

a) All shortest strongly simple paths from s to $[v,X]$ contain $[u,B]$.

b) level$([u,A]) \in V'$ is not defined.

c) For all $[w,B] \in V'$ which satisfy a) and b) hold level$([w,B]) \leq$ level$([u,B])$.

By the definition of $\mathrm{DOM}_{[v,X]}$ it is clear that $\mathrm{DOM}_{[v,X]}$ is unique if it exists. Next we investigate the properties of shortest strongly simple paths.

Lemma 5 *Assume that MBFS considers the edge* $([v,B],[w,A])$ *and case 2 is fulfilled. Then* $\mathrm{DOM}_{[v,B]} = [w,B]$.

Proof: Under the assumption that $\mathrm{DOM}_{[v,B]} = [u,B] \neq [w,B]$ we prove that $\mathrm{level}([w,A]) \leq \mathrm{level}([v,B])$ (see [5]). This contradicts case 2. ∎

The following lemma follows directly from the definition of $\mathrm{DOM}_{[v,B]}$.

Lemma 6 *Assume that* $\mathrm{DOM}_{[v,X]} = [u,B]$. *Then after the definition of* $\mathrm{level}([u,A])$ *it holds* $\mathrm{DOM}_{[v,X]} = \mathrm{DOM}_{[u,B]}$.

For $w \in V'$, we denote

$$r(w) = \begin{cases} [v,\overline{X}] & \text{if } w = [v,X] \\ t & \text{if } w = s \\ s & \text{if } w = t \end{cases}$$

Let $S = w_1, w_2, \ldots, w_k$ be a path in G_M. Then the *backpath* $r(S)$ of S is defined by

$$r(S) = r(w_k), r(w_{k-1}), \ldots, r(w_1).$$

Lemma 7 is very useful for the treatment of the case 3.

Lemma 7 *Assume that MBFS considers the edge* $([v,B],[w,A])$, $\mathrm{level}([w,A]) \leq \mathrm{level}([v,B])$ *and* $\mathrm{DOM}_{[v,B]} \neq \mathrm{DOM}_{[w,A]}$. *Let* $\mathrm{DOM}_{[w,A]} = [u,B]$. *If the edge* $([v,B],[w,A])$ *is contained in a shortest strongly simple path from* s *to* $[u,A]$, *then the following hold.*

a) The shortest strongly simple paths from s *to* $[u,A]$, *containing the edge* $([v,B],[w,A])$ *are in one-to-one correspondence to the paths* $P = P_1,[w,A],P_2,[u,A]$ *where* P_1 *is any shortest strongly simple path from* s *to* $[v,B]$ *which does not contain the node* $[u,B]$ *and* $P_2 = r(Q_2)$, *where* $Q = Q_1,[u,B],Q_2,[w,B]$ *is any shortest strongly simple path from* s *to* $[w,B]$.

b) $\mathrm{level}([u,A]) = \mathrm{level}([v,B]) + (\mathrm{level}([w,B]) - \mathrm{level}([u,B])) + 1$.

Proof: Note that b) follows directly from a). For proving a) let $P = P_1,[w,A],P_2,[u,A]$ be any shortest strongly simple path from s to $[u,A]$, containing $([u,B],[w,A])$. Let $Q = Q_1,[u,B],Q_2,[w,A]$ be any shortest strongly simple path from s to $[w,A]$. The assertion a) follows directly from the following claim.

Claim: For all $[z,X]$ with $[z,X] \in Q_2,[w,A],P_2$ hold $\mathrm{DOM}_{[z,X]} = \mathrm{DOM}_{[z,\overline{X}]} = [u,B]$.

We prove the claim by showing that the contrary implies that P is not a shortest strongly simple path (see [5]). ∎

Lemma 7 tells us how to compute the potential level of a node $[u,A]$ under the assumption that an edge $([v,B],[w,A])$ with $\mathrm{level}([w,A]) \leq \mathrm{level}([v,B])$ and $\mathrm{DOM}_{[v,B]} \neq \mathrm{DOM}_{[w,A]} = [u,B]$ is contained on a shortest strongly simple path from s to $[u,A]$. Let

$$\mathrm{potlevel}([u,A],([v,B],[w,A]))$$

denote this potential level. It is easy to see that if there exists an edge $([v,B],[w,A])$ with $\mathrm{level}([w,A]) \leq \mathrm{level}([v,B])$ on a shortest strongly simple path P from s to $[u,A]$ then also such an edge with $\mathrm{DOM}_{[v,B]} \neq \mathrm{DOM}_{[w,A]} = [u,B]$ on P exists.

Now we can describe the algorithm MBFS more in detail.

After the consideration of all edges $([v,B],[w,A])$ with $\mathrm{level}([v,B]) = l$, MBFS has also to consider all nodes with potential level $l+1$. If $\mathrm{potlevel}([u,A],([v,B],[w,A])) = l+1$ and $\mathrm{level}([u,A]) > l$ then MBFS defines $\mathrm{level}([u,A]) = l+1$ and add the indexed edge $([v,B],[w,A])_{[u,A]}$ to $\overline{E}$ where the index $[u,A]$ indicate that the edge is used for shortest strongly simple path from s to $[u,A]$. If we need later a subpath from $[w,A]$ to $[u,A]$ of such a path then by lemma 7 such a path can easily be constructed by taking the backpath $r(S)$ of any strongly simple path $S = [u,B],\ldots,[w,B]$ in the layered graph $\overline{G}_M$.

Additionally to the construction of the $(l+1)$-th level, MBFS has to compute potential levels greater than $l+1$ and also to update some $\mathrm{DOM}_{[v,X]}$, $[v,X] \in V'$.

If MBFS considers an edge $([v,B],[w,A])$ and case 3 is fulfilled, then MBFS checks if $\mathrm{DOM}_{[v,B]} \neq \mathrm{DOM}_{[w,A]}$ and if $\mathrm{DOM}_{[w,A]} = \mathrm{DOM}_{[w,B]}$. Possibly, level$([w,B])$ and hence also $\mathrm{DOM}_{[w,B]}$ are not defined. Then MBFS complete this step after the definition of level$([w,B])$. If both is fulfilled then MBFS computes potlevel$([u,A],([v,B],[w,A]))$:= level$([v,B])$ + (level$([w,B])$ − level$([u,B])$) + 1 where $\mathrm{DOM}_{[w,A]} = [u,B]$.

Let $\mathrm{Pred}([u,A]) = \{[v,B] \mid ([v,B],[u,A]) \in \overline{E} \text{ or } ([v,B],[w,A])_{[u,A]}) \in \overline{E} \text{ for some } [w,A] \in V\}$. It is easy to see that

$$\mathrm{DOM}_{[u,A]} = \begin{cases} \mathrm{DOM}_{[v,B]} & \text{if } [v_1,B],[v_2,B] \in \mathrm{Pred}([u,A]) \Rightarrow \mathrm{DOM}_{[v_1,B]} = \mathrm{DOM}_{[v_2,B]}. \\ & [v,B] \in \mathrm{Pred}([u,A]) \\ \text{undefined} & \text{otherwise} \end{cases}$$

MBFS updates the $\mathrm{DOM}_{[v,X]}$, $[v,X] \in V'$, as follows.

After the definition of level$([u,B])$:

$\mathrm{DOM}_{[u,B]} := \mathrm{DOM}_{[v,A]}$ where $[v,A]$ is the unique node with $([v,A],[u,B]) \in E'$.

After the definition of level$([u,A])$:

$\mathrm{DOM}_{[v,X]} := \mathrm{DOM}_{[u,B]}$ for all $[v,X]$ with $\mathrm{DOM}_{[u,X]} = [u,B]$.
$\mathrm{DOM}_{[x,A]} := \mathrm{DOM}_{[v,B]}$, $[v,B] \in \mathrm{Pred}([x,A])$ if $\mathrm{DOM}_{[x,A]}$ was undefined and now $[v_1,B],[v_2,B] \in \mathrm{Pred}([x,A]) \Rightarrow \mathrm{DOM}_{[v_1,A]} = \mathrm{DOM}_{[v_2,A]}$.
$\mathrm{DOM}_{[y,B]} := \mathrm{DOM}_{[x,A]}$ where $[y,B]$ is the unique node with $([x,A],[y,B]) \in E'$.

It is easy to see that MBFS can be implemented such that the needed time is bounded by $O(n+m)$ plus the time needed for the updating of the $\mathrm{DOM}_{[v,X]}$,$[v,X] \in V'$. This can easily be done using time $O(n^2)$. If we use incremental tree set union [9] we can reduce this bound to $O(n+m)$ [5]. The space complexity is also bounded by $O(n+m)$. These considerations lead to the following theorem.

Theorem 6 *MBFS can be implemented such that it uses only linear time and linear space in n and m.*

Note that MBFS needs only to compute all levels $\leq$ level(t). It is easy to prove by induction on the number of levels that $\overline{G}_M$ is correctly computed by MBFS (see [5]).

Knowing $\overline{G}_M$ it is easy to compute a maximal set of shortest strongly simple paths of $\overline{G}_M$ using MDFS in time $O(n+m)$. Everytime when a strongly simple path P from s to t is found, all nodes $[v,A]$, $[v,B]$ with $[v,A] \in P$ or $[v,B] \in P$ and all incident edges are deleted from $\overline{G}_M$. If a node get zero indegree or zero outdegree then also this node and all incident edges are deleted.

These considerations lead to the following simple algorithm for an entire phase.

Algorithm 4 *(maximal set of s-t paths)*

Step 1: Using MBFS, compute $\overline{G}_M$
Step 2: Using MDFS, compute a maximal set of strongly simple paths from s to t in $\overline{G}_M$.

Altogether, we have obtained the following theorem.

Theorem 7 *A maximum matching in a general graph $G = (V,E)$ can be found in time $O(\sqrt{n}m)$ and space $O(m+n)$, where $|V| = n$ and $|E| = m$.*

Conclusion: We have reduced the problem of finding an M-augmenting path in a general graph to a reachability problem in a directed graph. This approach has led to very simple algorithms for finding a maximum matching in general graphs. Especially we have obtained a straightforward algorithm for finding maximum matching in general graphs of time complexity $O(\sqrt{n}m)$.

Acknowledgement: I thank my students in my lectures on algorithms for matching problems held at the university of Saarbrücken during the winter term 1987/88 and at the university of Bonn during the winter term 1988/89 for their interest in hearing very premliminary, uncompletely and partly wrongly version of the paper. Without their patience, which was originated by the interest in learning how to attack hard problems, this paper would never been written.

References

[1] Aho A. V., Hopcroft J. E, Ullman J. D: *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974, 187–189.

[2] Balinski M. L.: *Labelling to Obtain a Maximum Matching*, in Combinatorial Mathematics and its Applications (R. C. Bose and T. A. Dowling, eds.), University of North Carolina Press, Chapel Hill, 1969, 585–602.

[3] Bartnik G.W.: *Algorithmes de couplages dans les graphes*, Thése Doctorat 3^e cycle, Université Paris VI,1978.

[4] Berge C.: *Two Theorems in Graph Theory*, Proc. Nat. Acad. Sci. U.S.A., **43** (1957), 449–844.

[5] Blum N.: *A New Approach to Maximum Matching in General Graphs*, Report No. 8546-CS, Institut für Informatik der Universität Bonn, Mai 1990.

[6] Edmonds J.: *Paths, Trees and Flowers*, Canad. J. Math, **17** (1965), 449–467.

[7] Even S, Kariv O.: *An $O(n^{2.5})$ Algorithm for Maximum Matching in General Graphs*, FOCS, **16** (1975), 100-112.

[8] Gabow H. N.: *An Efficient Implementation of Edmond's Algorithm for Maximum Matching on Graph*, J. ACM, 1976, 221–234.

[9] Gabow H. N., Tarjan R. E.: *A Linear-time Algorithm for a Special Case of Disjoint Set Union*, J. Comput. Syst. Sci., 1985, 209–221.

[10] Galil Z.: *Efficient Algorithms for Finding Maximum Matching in Graphs*, Computing Surveys, 1986, 23–38.

[11] Gondran M., Minoux M.: *Graphs and Algorithms*, Wiley & Sons, 1984, 283–284.

[12] Hopcroft J. E., Karp R. M.: *An $n^{5/2}$ Algorithm for Maximum Matching in Bipartite Graphs*, SIAM J. Comput., 1973, 225–231.

[13] Kariv O.: *An $O(n^{2.5})$ Algorithm for Maximum Matching in General Graphs*, Ph.D. thesis, Dept. of Applied Mathematics, Weizmann Institute of Science, Rehovort, Israel, 1976.

[14] Lawler E.: *Combinatorial Optimization, Networks and Matroids*, Holt, Rinehart and Winston, 1976.

[15] Micali S., Vazirani V. V.: *An $O(\sqrt{|V|} \cdot |E|)$ Algorithm for Finding Maximum Matching in General Graphs*, FOCS, **21** (1980), 12–27.

[16] Papadimitriou C. H., Steiglitz K.: *Combinatorial Optimization, Algorithms and Complexity*, Prentice-Hall, 1982.

[17] Peterson P. A., Loui M. C.: *The General Matching Algorithm of Micali and Vazirani*, Algorithmica, 1988, 511–533.

[18] Tarjan J. E.: *Data Structures and Network Algorithms*, SIAM, 1983.

[19] Vazirani V. V.: *A Theory of Alternating Paths and Blossoms for Proving Correctness of the $O(\sqrt{V}E)$ General Graph Matching Algorithm*, TR 89-1035, Dept. of Computer Science, Cornell University, Sept. 1989.

[20] Witzgall C., Zahn C. T. Jr.: *Modification of Edmond's Maximum Matching Algorithm*, J. Res. Nat. Bur. Standards, **69 B** (1965), 91–98.