AN $O(N^{2.5})$ ALGORITHM FOR MAXIMUM MATCHING IN GENERAL GRAPHS†

S. Even* and O. Kariv**

Abstract

This work presents a new efficient algorithm for finding a maximum matching in an arbitrary graph. Two implementations are suggested, the complexity of the first is $O(n^{2.5})$ and the complexity of the second is $O(m\sqrt{n}\cdot\log n)$ where n,m are the numbers of the vertices and the edges in the graph.

0. Introduction

A matching M in an undirected graph G is a subset of the edges of the graph such that no two edges of M have a common vertex. The problem of finding a maximum matching in a graph means finding a matching M whose number of edges is maximum. This problem was discussed in detail by J. Edmonds [1], who investigated the properties of graphs which determine the maximum matching, and described an efficient algorithm for finding a maximum matching in an arbitrary graph. The complexity of his algorithm is $O(n^4)$. C. Witzgall and C. T. Zahn [2] suggested an improvement of Edmonds' algorithm but it still had the complexity $O(n^4)$. H. Gabow [3] gave an implementation of Edmonds' algorithm which works in $O(n^3)$ steps.

All the above algorithms are based on Berge's theorem [4] which states that a given matching is not maximum if and only if there exists an augmenting path which joins two exposed vertices in the graph. Such a path can be used to improve the present matching by changing each free edge on the augmenting path to be a matched edge and vice-versa.

J. E. Hopcroft and R. M. Karp [5] showed that it is possible to carry out the construction of a maximum matching in $\sqrt{n}$ phases, when in each phase a maximal set of vertex disjoint minimum length augmenting paths is found. For bipartite graphs they showed how to carry out each phase in $O(m)$ steps (where m is the number of edges in the graph) and thus achieved an algorithm which finds a maximum matching in $O(m\sqrt{n})$ steps. The construction of such an algorithm for the general case still remained an open question. Our algorithm uses at most $O(n^{2.5})$ steps, and by changing the data structure we get a version which uses at most $O(m\sqrt{n}\log n)$ steps.

Definitions.

Let G be an arbitrary graph and let M be a matching of this graph. We refer to the edges of the graph as if they are directed (i.e. XY is distinct from YX). The following terminology is used:

Matched (free) edge - an edge which belongs (does not belong) to M.
MATE(X) - the other end of a matched edge which is incident to X.
Exposed vertex - a vertex which is not incident to any edge of M.
An alternating path to a matched vertex X (from an exposed vertex A) - a path* whose edges are alternatingly matched and free such that in one end of the path there is a matched edge (which meets X) and in the other end there is a free edge (which meets A).
An augmenting path - a path which connects two exposed vertices and whose edges are matched and free alternatingly.
(Odd) loop - a simple circuit containing k matched edges and k+1 free edges. (Clearly, each matched edge lies between two free edges).
The base of an odd loop - the (only) vertex of the odd loop in which two free edges of the loop are adjacent.
Illegal alternating (augmenting) path - an alternating (augmenting) path which contains an odd loop.
The length of a path - the number of free edges in the path.
The level of a matched (exposed) vertex - the length of a shortest legal alternating (augmenting) path which leads to this vertex.

General ideas

As in all the attempts known to us to find a maximum matching, we too search for augmenting paths in the graph. The main difficulty in constructing an augmenting path arises in the case where the path "closes on itself", thus creating an odd loop. Edmonds [1] overcomes this difficulty by "shrinking" all the loops which are discovered. Witzgall and Zahn [2] as well as Gabow [3] register these loops in a special data structure. In our algorithm we use some techniques based on the principles of Breadth First Search (BFS) and Depth First Search (DFS) (with slight differences forced by the characteristics of the problem). We perform shrinking of certain

† This paper describes the results of a Ph.D. thesis of O. Kariv, to be submitted to the Feinberg Graduate School of the Weizmann Institute of Science, Rehovot, Israel. S. Even is the thesis supervisor.

* Department of Computer Science, Technion, Haifa, Israel. On leave of absence from the Weizmann Institute of Science, Rehovot, Israel.

** Department of Applied Mathematics, the Weizmann Institute of Science, Rehovot, Israel.

* By the term "path" we mean a sequence of vertices. Since we assume that there are no parallel edges in the graph, the sequence of vertices defines unambiguously a sequence of edges.

odd loops (resembling the blossoms' shrinking of Edmonds) and we define a data structure (similar to that defined by Gabow) which serves to restore the augmenting paths. Yet, the use of these techniques alternates according to an arranged and well defined system, in order to improve the efficiency of each stage. The data structure enables to trace minimum augmenting paths and yet it contains chains of information which make it possible to find a new augmenting path in case one of the former paths fails. This is done without searching the graph again.

Description of a phase. Our algorithm works in phases (using Hopcroft and Karp's results [5]). Each phase consists of four stages:

In the first stage we simultaneously start from all the exposed vertices and we search the graph by BFS. We find for each vertex one of the minimum alternating paths leading to it. (These alternating paths are all loop-free and thus are legal. This is achieved without loop shrinking. The method is similar to that of Gabow except that it is performed in parallel from all exposed vertices and information about alternative alternating paths is stored.) Paths are registered in a special data structure which makes it restorable later on. Finding minimum alternating paths enables to determine the level of vertices. During the search the first return to an exposed vertex indicates a minimum augmenting path. Therefore, at the end of the first stage we know the level of each vertex and the length r of a minimum augmenting path in the graph (if there is no such path the matching is maximum and the algorithm terminates).

In the second stage we use the levels defined for the vertices and the minimum alternating paths found for them in the first stage, in order to shrink all the odd loops which lie on illegal augmenting paths whose length is less than r. Each such odd loop is replaced by its base. We also remove from the graph all the edges which do not lie on minimum (legal) augmenting paths. In the end of the second stage we have a reduced graph in which each vertex lies on a minimum augmenting path (of the original graph), and the place of the vertex on this graph is identical to its level. Every maximal set of disjoint legal augmenting paths of the reduced graph corresponds to a maximal set of disjoint and legal minimum augmenting paths (of length r) of the original graph.

In the third stage we construct a maximal set of disjoint legal augmenting paths of the reduced graph. For each exposed vertex we search the reduced graph by the DFS method (or rather by the HLFS method - Highest Level First Search) in order to find a legal augmenting path leading from it to another exposed vertex. Each edge is searched at most once in each direction. It is shown that if an edge is not used on a legal augmenting path in the current search, then it is not useful in the following searches. The paths are registered in a data structure similar to the one used in the first stage.

In the fourth stage we restore the augmenting paths found in the third stage. At first we trace the path as it appeared in the reduced graph and then we trace the path in the original graph from which the path of the reduced graph was derived. We improve the matching by converting each free edge on the path to a matched one and vice-versa. Once the tracing and the utilization of all the augmenting paths found is done the phase ends.

1. The First Stage

Description of the first stage.

The goals of the first stage are: finding for each matched vertex X a minimum alternating path leading to it from an exposed vertex and defining the level of X as the length of this minimum path; finding the length of a minimum augmenting path in the graph and identifying all the exposed vertices at the ends of minimum augmenting paths.

Representation of exposed vertices. For the performance of the first stage it is convenient to replace every exposed vertex V by a matched edge $V^{(1)}V^{(2)}$, such that the vertex $V^{(1)}$ meets all the (free) edges which are incident to V, while $V^{(2)}$ meets no free edges. We shall refer to the vertex $V^{(1)}$ ($V^{(2)}$) as a 1-exposed vertex (2-exposed vertex). During the first stage the level of the 1-exposed vertices is 0. By this definition the treatment of the exposed vertices during the first stage will be the same as if they are matched, and we can reformulate the goals of the first stage to be: Finding for each vertex X in the graph a minimum alternating path leading to it from a 1-exposed vertex, and defining the level of X as the length of this minimum path. (In particular if X is a 2-exposed vertex $B^{(2)}$, then a (minimum) alternating path leading to it from a 1-exposed vertex $A^{(1)}$ is really a (minimum) augmenting path leading from the exposed vertex A to the exposed vertex B).

Notations of levels and alternating paths. Assume that we find a minimum alternating path leading from a 1-exposed vertex to the vertex X. We refer to this path as the (minimum) al-path leading to X and we denot it p(0→X). Let Q be a vertex which lies on p(0→X): We use the notation p(Q→X) to denote the segment from Q to X on the path p(0→X). While referring to this segment in the opposite direction (from X to Q) we use the notation p(Q←X).

Let ST be an edge lying on p(0→X) such that on this path T is closer to X than S. We say that ST is forward-lying (f-lying) on p(0→X). If ST is a matched edge we say that T is f-lying on p(0→X). Let X be a vertex and let ST be a free edge f-lying on p(0→X) such that p(0→X) = p(0→S)·p(T→X). Then p(0→S) is called a head of p(0→X).

Assume that we find an alternating path leading to a vertex X from a 1-exposed vertex. In order to define the level of X as the length of this alternating path we have to be sure that this alternating path is both legal and minimum. Until these two conditions are verified we refer to this alternating path and to its length as the potential al-path p(0→X) and the potential level of X, respectively. When we know that p(0→X) is legal and minimum we refer to it as the (well-)defined al-path p(0→X)

and we refer to its length as the (well-) defined level of X. We use the notation $\ell(X)$ for the level of the vertex X, potential or well-defined, as the case may be.

It is convenient to assume that in the beginning of the first stage all the vertices (except the 1-exposed vertices) have a potential level of $\lfloor \frac{n}{2} \rfloor + 1$. Since this level is greater than any actual level that may be defined by an alternating path, we can think of the process of finding levels as a process of minimizing levels of vertices. For each of the possible values of the levels, $0,1,\ldots\lfloor \frac{n}{2} \rfloor$, $\lfloor \frac{n}{2} \rfloor+1$, a LIST is defined which contains all vertices of the same level.

The BFS method. The first stage of the algorithm is performed in substages. On the j-th substage we search all free edges incident to vertices whose well-defined level is j-1, find all vertices whose well-defined level is j and possibly label some vertices with higher potential levels.

Let X be a vertex whose well-defined level is j-1, let XY be a free edge, and let U and Z be the mates of X and Y respectively. If the present (potential) level of Z is greater than j, we try to replace the present potential al-path $p(0\to Z)$ by a shorter alternating path defined by the concatenation: $p(0\to Z)=p(0\to X)\cdot Y\cdot Z$. The length of the new al-path is j, thus the new potential level of Z will be j. However, we do not discard the information which enabled the previous level but we keep it in a special data structure as will be explained later.

The registration of an al-path. In order to record an al-path $p(0\to Z)$ in an economical yet traceable way, we use a recursive record: Each vertex Z to which an al-path $p(0\to Z)$ is found, is given a link through which we are able to recursively retrace $p(0\to Z)$. Following Gabow [3] there are two types of links: If Y (the mate of Z) has no link yet, Z is given a first-type link (denoted by 1-link). Else Z is given a second-type link (denoted by 2-link).

As we shall see, if the vertices Y and Z are mates, then the first of them to get a link is given a 1-link which well-defines its level. If the other vertex is given (later on) a link, it must be a 2-link, and it defines a potential level. When a potential al-path $p(0\to Z)$ and a potential level of Z are replaced by a shorter potential al-path and a lower level, the former 2-link of Z is also replaced by the new one. The former 2-link is recorded in a data structure to be described later.

1-link. A 1-link is a link to a vertex whose level is already well-defined: Let Z be a vertex which has a 1-link to vertex X (LINK(Z)=X), and let Y=MATE(Z). Then the al-path $p(0\to Z)$ leading to Z is defined by the concatenation:

$$p(0\to Z) = p(0\to X)\cdot Y\cdot Z \qquad (1.1)$$

The 1-link is given to Z in the j-th substage when we search the free edge XY where X has a (well-defined) level j-1 and Y and Z both have no link yet (thus, both have the initial level $\lfloor \frac{n}{2} \rfloor + 1$). Since X has a well-defined level, the al-path $p(0\to X)$ is legal. Since Y and Z have the initial level, the edge YZ can not lie on $p(0\to X)$. Thus, $p(0\to Z)$ is legal and $\ell(Z)$ is well-defined.

Bridge. Let XY be a free edge where X and Y both have well-defined levels, and let U and Z be the mates of X and Y respectively. If neither LINK(U)=Y nor LINK(Z)=X then XY is a well-defined bridge which connects the two al-paths $p(0\to X)$ and $p(0\to Y)$. Clearly YX is then a well-defined bridge too.

If X has a well-defined level while LINK(Y) is still not well-defined and LINK(Z)≠X, then LINK(U)=Y can not hold and we call XY a potential bridge. When LINK(Y) becomes well-defined XY becomes a well-defined bridge.

2-link. A 2-link of a vertex Z whose mate is Y is a well-defined bridge PQ such that $p(0\to Y)$ is a head of $p(0\to Q)$. The al-path defined by this link is:

$$p(0\to Z) = p(0\to P)\cdot p(Z\leftarrow Q) \qquad (1.2)$$

In the algorithm, a 2-link may be defined through two possible methods. The first method is as follows: Let XY be a free edge which is searched in the j-th substage (thus, X has a well-defined level j-1). Let U=MATE(X) and Z=MATE(Y). Assume that Y has a 1-link while $\ell(Z)>j$. Since Z cannot have a 1-link, LINK(Z)≠X. Thus XY is not a (well-defined) bridge if and only if LINK(U)=Y. In this case X must have a 2-link to some bridge PQ. We assign:

$$\ell(Z)=j;\quad \mathrm{LINK}(Z)=\begin{cases} XY & \text{if } \mathrm{LINK}(U)\neq Y \\ \mathrm{LINK}(X) & \text{if } \mathrm{LINK}(U)=Y \end{cases} \qquad (1.3)$$

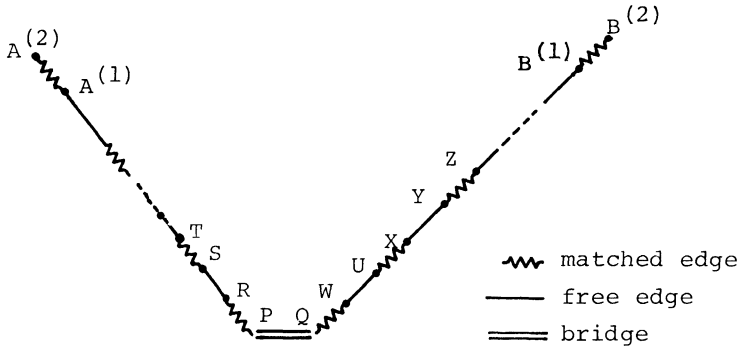
This assignment can be proved, by induction on j, to be consistent with (1.2). A 2-link assigned through the first method is denoted by 2(1)-link.

The second method of assigning a 2-link to a vertex Z is performed when the former 2-link of Z (which was assigned through either method) is found to be illegal. In that case Z is given a 2-link to a (well-defined) bridge PQ such that (in addition to (1.2)) all the vertices which are f-lying on $p(Q\to Z)$ (except Z itself) have already well-defined levels. The exact application of the second method will be described later. A 2-link assigned through the second method is denoted by 2(2)-link.

Notice that if Z is assigned a 2-link then $p(0\to Z)$ as defined by (1.2) is still potential, unless a further check of its legality is carried out. For example, assume that both $p(0\to P)$ and $p(0\to Q)$ emerge from the same exposed vertex, and let $p(0\to M)$ be their longest common head. Then if Z is f-lying on $p(0\to M)$, the al-path $p(0\to Z)$ as defined by (1.2) contains an odd loop (whose base is M) and therefore is illegal. As we shall see, the structure of the 2-link enables a simple check of the legality of $p(0\to Z)$ when the search of the edges in the j-th substage is ended. Thus, in the end of the j-th substage either $p(0\to Z)$ becomes well-defined, or it is found to be illegal; in the latter case an attempt is made to define a (new) 2(2)-link for Z, or (if the attempt fails) LINK(Z) is left undefined.

(See Fig. 1 for example of 1-links and 2-links.)

Example of 1-links and 2-links:



Vertex	Link-type	Link
Y	1	...
U	1	Y
Q	1	U
S	1	...
P	1	S
W	2	PQ
X	2	PQ
Z	2	PQ
R	2	QP
T	2	QP

Figure 1.

The three following lemmas are derived from the definition of LINK(Z):

Lemma 1.1: In the end of the j-th substage the vertices whose levels are well-defined are exactly the vertices whose levels are not greater than j.

Proof: By induction on j.

Lemma 1.2: If a vertex Z has a link, then all the vertices which are f-lying on $p(0\rightarrow Z)$ (except Z itself) have well-defined levels less than $\ell(Z)$, and all the free edges which are f-lying on $p(0\rightarrow Z)$ were already searched.

Proof: By induction on $\ell(Z)$.

Corollary 1.1: If LINK(Z)=PQ, then all the vertices lying on $p(P\rightarrow Z)$ have well-defined levels less than $\ell(Z)$, and all the free edges on $p(P\rightarrow Z)$ were already searched in both directions.

TOP(Z). For the purpose of checking the legality of an al-path defined by a 2-link and defining an alternative al-path in case the former is illegal - we define for each vertex a variable called TOP as follows:

(a) Let Z be a vertex with a 2-link and let RT be the first free edge f-lying on $p(0\leftarrow MATE(Z))$ which has not been searched yet (in this direction) during the present performance of the first stage. Then TOP(Z)=R.

(b) Let Z be a vertex with a 1-link. If MATE(Z) has a well-defined level through a link to PQ, then TOP(Z)=TOP(MATE(P)).

(c) Else (if Z or MATE(Z) have no well-defined link), TOP(Z)=Z.

The intuitive meaning of TOP(Z) is the "farthest" vertex on the "continuation" of $p(0\rightarrow Z)$ which has been reached so far during the search of the edges of the graph. Clearly this is true in case (a). If Z has a 1-link and its mate has a well-defined 2-link to the bridge PQ (case (b)) then the "continuation" of $p(0\rightarrow Z)$ is along $p(0\leftarrow MATE(Z))$ and thus along $p(0\leftarrow P)$. The segment $p(P\leftarrow Z)$ has already been searched (corollary 1.1) thus TOP(Z)=TOP(MATE(P)). Finally, if Z or MATE(Z) have no well-defined link yet (case (c)), the "continuation" of $p(0\rightarrow Z)$ may still not be unique, thus we define TOP(Z)=Z.

TOP(Z) is f-lying on $p(0\leftarrow MATE(Z))$. It is clear that all the vertices lying on $p(0\leftarrow MATE(Z))$ between Z and TOP(Z) must have well-defined levels less than $\ell(TOP(Z))$ (by corollary 1.1). At the end of the j-th substage TOP(Z) itself has a level not less than j.

Lemma 1.3: If LINK(Z)=PQ then:

$$TOP(Z) = TOP(MATE(Q)) \qquad (1.4)$$

Proof: According to corollary 1.1 all the edges between P and Z have already been searched in both directions. Thus TOP(MATE(Q)) is not lying on $p(0\leftarrow Q)$ between Q and Z. Since $p(0\rightarrow MATE(Z))$ is a head of $p(0\rightarrow Q)$ (definition (1.2)) one may conclude that (1.4) holds.

QED

Lemma 1.4: Let R=TOP(Z) and let Y=MATE(Z). Then $p(0\rightarrow MATE(R))$ is a head of $p(0\rightarrow Y)$.

Proof: By induction on $\ell(Y)$. If Y has a 1-link to a vertex W then $p(0\rightarrow Y)=p(0\rightarrow W)\cdot Z\cdot Y$. According to the definition, either TOP(Z)=Z in which case the lemma holds, or TOP(Z)=TOP(MATE(W)). In the latter case R=TOP(MATE(W)) thus by the induction $p(0\rightarrow MATE(R))$ is a head of $p(0\rightarrow W)$ and therefore of $p(0\rightarrow Y)$. If Y has a 2-link to the well-defined bridge PQ, then $p(0\rightarrow Y)=p(0\rightarrow P)\cdot p(Y\leftarrow Q)$ and according to the definition TOP(Z)=TOP(MATE(P)). Thus R=TOP(MATE(P)) and by the induction $p(0\rightarrow MATE(R))$ is a head of $p(0\rightarrow P)$ and therefore of $p(0\rightarrow Y)$.

QED

The set T(R). Define:

$$T(R) = \{Z \mid TOP(Z) = R\} \qquad (1.5)$$

In the beginning of the first stage each vertex R satisfies $T(R)=\{R\}$. But during the search of the free edges the set T(R) is changed as follows:

(a) Assume we are in the j-th substage and the free edge XY is being searched. Let R=TOP(MATE(Y)). If LINK(MATE(X))=Y then by Lemma 1.4 and the definition of TOP, the sets T(X) and T(R) are updated as follows:

$$T(R)\leftarrow T(R) \cup T(X) \;;\; T(X)\leftarrow\phi \qquad (1.6)$$

(b) Let Y and Z be mates. If at the end of the j-th substage LINK(Z)=PQ and $\ell(Z)=j$ are found to be well-defined then by the definition of TOP, T(TOP(MATE(P))) and T(Y) are updated as follows:

$$T(TOP(MATE(P)))\leftarrow T(TOP(MATE(P)))\cup\{Y\};\quad T(Y)\leftarrow\phi \tag{1.7}$$

Theorem 1.1:

(a) Let LINK(R)=XY and Z=MATE(Y). If TOP(R)=R then $Z\in T(R)$.

(b) Let $Z\in T(R)$, Y=MATE(Z) and XY be a well-defined bridge. If MATE(R) has a 1-link, then an alternating path to R can be defined by LINK(R)=XY.

Proof: (a) by Lemma 1.3.

(b) by Lemma 1.4, $p(0\rightarrow MATE(R))$ is a head of $p(0\rightarrow Y)$, thus LINK(R)=XY is consistent with the definition of a 2-link.

Therefore when a 2-link of a vertex R fails, we have to look for the alternative 2-link among the well-defined bridges XY such that $MATE(Y)\in T(R)$.

The value of an edge. Let XY be an edge. The value of XY is defined to be the sum of the levels of X and Y:

$$\ell(XY)=\ell(X)+\ell(Y) \tag{1.8}$$

If LINK(R)=XY then

$$\ell(R)=\ell(XY)-\ell(MATE(R))+1 \tag{1.9}$$

The assignment of a 2(2)-link. Theorem 1.1 and Eq. (1.9) enable the assignment of a 2(2)-link:

(a) If an edge XY is being searched such that TOP(MATE(Y))=R, and $\ell(R)>\ell(XY)-\ell(MATE(R))+1$ then assign LINK(R) by eq. (1.3) and $\ell(R)$ by eq. (1.9). (If R=MATE(Y) then this assignment coincides with the 2(1)-link assignment).

(b) If a 2-link of a vertex R fails and there are well-defined bridges XY which so far have not been found to close a loop and such that $MATE(Y)\in T(R)$, then assign one of these bridges whose value is minimum as the new link of R and assign $\ell(R)$ by eq. (1.9); else LINK(R) is left undefined and $\ell(R)=\lfloor\frac{n}{2}\rfloor+1$.

The definition of a 2-link. The reader may turn to the claims following the 2(1)-link assignment to observe that they remain valid under this broader assignment. Also, both the 2(1)-link and the 2(2)-link are consistent with the definition of a 2-link.

The chain of a vertex. For the purpose of realizing part (b) of the 2(2)-link assignment we define for each vertex Z a chain CH(Z) which is a list of (potential or well-defined) bridges XY, such that Y=MATE(Z), as follows:

In the beginning of the first stage all the chains are empty. When a (potential or well-defined) bridge is searched, the bridge XY is attached to the end of CH(Z). If the bridges of type XY (both potential and well-defined) are searched in the order $X_1Y, X_2Y\ldots X_kY$, then $\ell(X_1)\leq\ell(X_2)\leq\ldots\leq\ell(X_k)$. Thus $\ell(X_1Y)\leq\ell(X_2Y)\leq\ldots\leq\ell(X_kY)$. The bridges are listed in CH(Z) in the order of their searching, and thus in nondecreasing order of their values. The value of the first bridge is therefore the least, and is defined as VAL(Z) (the value of Z). If $CH(Z)=\phi$ then $VAL(Z)=n$. Clarly, if the first bridge is found to close a loop it is dropped from CH(Z) and VAL(Z) has to be updated per the amended CH(Z).

By the discussion following the definition of TOP, if $TOP(Z)\neq Z$ then all the bridges of CH(Z) are already well-defined. If the bridge XY can define a level j for TOP(Z) (where Z=MATE(Y)) then X must have a level less than j, thus in the beginning of the j-th substage XY has already been searched and thus belongs to CH(Z).

Checking the legality of a 2-link.

Theorem 1.2: Let Z be a vertex which has a 2-link to the bridge PQ, and let $\ell(Z)=j$. Then the al-path $p(0\rightarrow Z)$ defined by LINK(Z)=PQ is illegal if and only if at the end of the search of the edges in the j-th substage the following equation holds:

$$TOP(MATE(P))=TOP(MATE(Q)) \tag{1.10}$$

or equivalently, by (1.4) (since at the end of the search in the j-th substage TOP(Z)=Z):

$$TOP(MATE(P))=Z \tag{1.11}$$

Proof: (a) Assume that (1.11) holds. Then Z is f-lying on $p(0\leftarrow P)$. But by (1.2) $p(0\rightarrow Z)=p(0\rightarrow P)\cdot p(Z\leftarrow Q)$. Thus Z is f-lying on $p(0\leftarrow Z)$ and therefore $p(0\rightarrow Z)$ is illegal.

(b) The proof that if $p(0\rightarrow Z)$ is illegal then (1.11) holds is complex since several cases must be considered. However the main idea is that if $p(0\rightarrow Z)$ is illegal and (1.11) does not hold, then a shorter and legal al-path to Z can be found - contrary to the assumption that $\ell(Z)=j$ at the end of the search of the edges in the j-th substage.

Corollary 1.2: Let Z be a vertex such that LINK(Z)=PQ and $\ell(Z)=j$. In order to make sure that the link and the level of Z are well-defined it is enough to check eq. (1.11) at the end of the search of the j-th substage: If $TOP(MATE(P))\neq Z$ then LINK(Z)=PQ and (Z)=j are declared well-defined. Otherwise PQ is deleted from CH(MATE(Q)) and VAL(MATE(Q)) is reassigned: If $CH(MATE(Q))=\phi$ then VAL(MATE(Q))=n, else VAL(MATE(Q)) is assigned $\ell(P'Q)$ where P'Q is the first bridge in the updated CH(MATE(Q)). Also, let V be a vertex in T(Z) for which VAL(V) is minimum and let ST be the first bridge in CH(V). If TOP(MATE(S))=Z then delete ST from CH(V) and repeat the search for a V in T(Z) etc. If $TOP(MATE(S))\neq Z$ then assign LINK(Z)=ST (assignment of a 2(2)-link, case (b)); $\ell(Z)$ is assigned by (1.9). If $\ell(Z)=j$ then it becomes well-defined; else it remains potential.

The algorithm of the first stage.

0. (Initialization).
   (a) For $i=1,2\ldots\lfloor\frac{n}{2}\rfloor$ assign: $LIST(i)\leftarrow\phi$.
   (b) For each vertex Z assign: $TOP(Z)\leftarrow Z$; $LINK(Z)\leftarrow 0$; $VAL(Z)\leftarrow n$; $T(Z)\leftarrow\{Z\}$; $CH(Z)\leftarrow\phi$.

(c) For each vertex Z such that Z is not 1-exposed, insert Z into LIST($\lfloor \frac{n}{2} \rfloor$+1) and assign $\ell(Z) \leftarrow \lfloor \frac{n}{2} \rfloor + 1$.

(d) For each 1-exposed vertex B, insert B into LIST(0) and assign $\ell(B) \leftarrow 0$;

(e) $j \leftarrow 1$.

UBSTAGE:

1. If LIST(j-1)=ϕ then go to VERIFICATION.

EARCH:

2. If all the vertices in LIST(j-1) have already been searched then go to VERIFICATION; else, let X be a vertex in LIST(j-1) which has not been searched yet.

3. If all the free edges XY have already been searched, return to SEARCH; else let XY be a free edge which has not been searched yet.

4. Let Z=MATE(Y). If LINK(Y)$\neq$0 go to 7.

5. If LINK(Z)=0 then go to 6; else (XY is a (potential) bridge), insert XY into CH(Z), and return to 3.

6. (1-link assignment). Assign: LINK(Z)$\leftarrow$X; $\ell(Z) \leftarrow j$; delete Z from LIST($\lfloor \frac{n}{2} \rfloor$+1) and insert Z into LIST(j). Return to 3.

7. (Y has a link). If LINK(MATE(X))=Y then go to 9; else (XY is a bridge), if TOP(MATE(X))=TOP(MATE(Y)) then return to 3.

8. (XY is a bridge). If CH(Z)=ϕ and ($\ell(Y)<j$ or Y has 1-link) then assign: $VAL(Z) \leftarrow j-1+\ell(Y)$.

   Insert XY as the last bridge in CH(Z). Let PQ$\leftarrow$XY and go to 10.

9. (XY is not a bridge). Execute: T(TOP(Z))$\leftarrow$T(TOP(Z)) $\cup$ T(X); T(X)$\leftarrow\phi$. Let LINK(X)=PQ.

10. (A 2-link assignment). Let $\ell_1 \leftarrow \ell(PQ) - \ell(MATE(TOP(Z))) + 1$; if $\ell(TOP(Z)) \leq \ell_1$ then return to 3. Else, delete TOP(Z) from LIST(ℓ(TOP(Z))) and insert TOP(Z) into LIST(ℓ_1). Assign: $\ell(TOP(Z)) \leftarrow \ell_1$; LINK(TOP(Z))$\leftarrow$PQ. Return to 3.

END OF SEARCH.

VERIFICATION:

11. If LIST(j)=ϕ then go to 18.

12. If all the vertices in LIST(j) have already been verified then go to END OF VERIFICATION; else let Z be a vertex in LIST(j) which has not been verified yet. If Z has a 1-link (it is verified) repeat 12.

13. (Z has a 2-link). Let ST=LINK(Z), and let V=MATE(T). If TOP(MATE(S))$\neq$Z then go to 16; else delete Z from LIST(j).

14. (ST is found to close a loop). Delete ST from CH(V). If CH(V)=ϕ then VAL(V)$\leftarrow$n; else let UT be the first edge in CH(V), then assign: $VAL(V) \leftarrow \ell(U) + \ell(T)$.

15. (Find a new link for Z). Let V$\in$T(Z) such that VAL(V) is minimum. If CH(V)=ϕ then assign: LINK(Z)$\leftarrow$0, $\ell(Z) \leftarrow \lfloor \frac{n}{2} \rfloor + 1$, insert Z into LIST($\lfloor \frac{n}{2} \rfloor$+1) and return to 12; else, let ST be the first edge in CH(V). If TOP(MATE(S))=Z then return to 14; else assign (2(2)-link assignment): LINK(Z)$\leftarrow$ST, $\ell(Z) \leftarrow \ell(ST) - \ell(MATE(Z)) + 1$; insert Z into LIST($\ell$(Z)). If $\ell(Z) > j$ then return to 12.

16. (Update T(MATE(Z))). Let Y=MATE(Z) and let ST=LINK(Z). Execute: T(TOP(MATE(S)))$\leftarrow$T(TOP(MATE(S))) $\cup$ T(Y); T(Y)$\leftarrow\phi$. If CH(Y)=ϕ then assign: VAL(Y)$\leftarrow$n; else let WZ be the first edge in CH(Y), then assign: $VAL(Y) \leftarrow \ell(W) + \ell(Z)$. Return to 12.

END OF VERIFICATION.

17. If LIST(j) contains a 2-exposed vertex assign r$\leftarrow$j and proceed to the second stage.

18. Assign: j$\leftarrow$j+1. If j$\leq\lfloor \frac{n}{2} \rfloor$ then go to SUBSTAGE.

END OF SUBSTAGE.

19. Halt. (The matching is maximum).

Validity and complexity of the first stage.

Comments on the validity of the first stage. The goal of the first stage is to define for each vertex its level, i.e., to find for each vertex Z a legal al-path leading to Z whose length is minimum. The problem of legality is solved by using the TOP variable. But the question of minimality needs further explanation.

As one can see, during the first stage we construct and deal only with alternating paths which can be described by a 1-link or a 2-link. But there are many alternating paths which can not be described by these links. In fact, any alternating path which is a concatenation of several segments of different al-paths defined by 2-links - is not referred in our treatment.

For example, in the following figure 2, the al-path p(0$\rightarrow$P) is emerging from the 1-exposed vertex $A^{(1)}$, while the al-path p(0$\rightarrow$Q) is emerging from the 1-exposed vertex $B^{(1)}$. Our treatment discovers the alternating paths from $A^{(1)}$ to L and to M (through the bridges PQ and WR respectively), from $B^{(1)}$ to K and to N (through the bridges QP and TS respectively), from $C^{(1)}$ to L (through the bridge ST) and from $D^{(1)}$ to K (through the bridge RW). Our treatment cannot discover the alternating paths from $C^{(1)}$ to M and from $D^{(1)}$ to N. (See Fig. 2.)

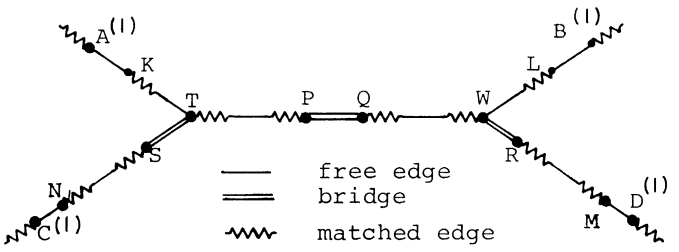


Figure 2.

Yet it can be proved that those paths which are not discovered by our algorithm, are not essential for the definition of levels since other paths which are not longer are used. The proof is by contradiction: We assume that Z is the vertex with the least real level for which a wrong (higher) level has been defined by our algorithm. Then there is a real alternating path $p(0\to Z)$ leading to Z which is shorter than the al-path defined by LINK(Z). By searching the levels of the vertices along $p(0\leftarrow Z)$ a contradiction is reached. We omit further details.

The complexity of the first stage. We need efficient algorithms and data structures to perform the following:

(1) Changes of VAL(Z); at most m such changes may be called for.

(2) Finding a vertex Z in T(R) such that VAL(Z) is minimum; at most m such operations may be required.

(3) Merging $T(R_1)$ with $T(R_2)$; at most n unions may be required.

The simplest implementation is to record T(R) just as a simple list. Each change of VAL then costs one step while each merging costs n steps. In this implementation steps 14-15 of the algorithm are performed as follows: The set T(Z) is searched. For each vertex $V\in T(Z)$, VAL(V) is verified and updated if necessary (by deleting the first bridges of CH(V) which close odd loops). Then we find in T(Z) a vertex V such that VAL(V) is minimum and the first bridge of CH(V) becomes the link of Z. Thus, each finding of a vertex V in T(Z) whose VAL is minimum costs n steps and at most n such operations are performed. The complexity of the whole implementation is therefore $0(n^2)$.

Another implementation is by replacing each change of VAL(Z) by the two operations DELETE and INSERT (delete Z with its old value and insert Z with its new value). An efficient implementation using 2-3 trees (described by Aho, Hopcroft and Ullman [6]) works in $m\cdot\log n$ steps.

A more complicated implementation, not described here, has complexity 0(g) where $g=\min(m\cdot\log n, \max(m,n^{1+\varepsilon}))$ where $\varepsilon>0$ may have any positive small value.

As one can see, except for the implementation of the operations on the sets {T(R)}, the whole first stage is just a search of the edges in substages where a constant number of operations are performed per each edge. Thus the complexity of the search is 0(m). Since the complexity of the operations on the sets {T(R)} is always not less than 0(m), the complexity of the whole first stage is thus equal to the complexity of the implementation of the operations on the sets {T(R)}. (In fact, this implementation is the bottle-neck of the first stage and of the whole algorithm.) Therefore, the complexity of the first stage is $0(n^2)$ if we use the simple data-structure for T(R), or it is $0(m\cdot\log n)$ if we use 2-3 trees.

2. The Second Stage

General description.

The final goal of each phase of the algorithm is to find a maximal set of disjoint minimum augmenting paths. In the second stage we reduce the original graph by removing some of its edges and by shrinking others. The reduced graph will have the following property: Any maximal set of disjoint augmenting paths in the reduced graph corresponds to a maximal set of disjoint minimum augmenting paths in the original graph. Clearly the renouncement of the requirement that the augmenting paths be minimum makes the search of a maximal set of disjoint paths simpler.

In the beginning of the second stage we know the length r of the minimum augmenting paths in the graph. For each i, $i\leq r$, we also know the list LIST(i) of all the vertices Z whose levels are i, and we can retrace the al-paths $p(0\to Z)$ leading to these vertices. The other vertices are not relevant for the present phase since they are not lying on any minimum augmenting path.

The irrelevant edges. Let XY be a free (or a matched) edge. Any augmenting path passing through XY consists in fact of two alternating paths, one leading to X and the other leading to Y. Thus the length of an augmenting path passing through XY must be at least $\ell(X)+\ell(Y)+1$ (or $\ell(X)+\ell(Y)$ if XY is matched). Define:

$$d(XY)=\begin{cases}\ell(XY)+1 & \text{if XY is free}\\ \ell(XY) & \text{if XY is matched}\end{cases} \qquad (2.1)$$

Then, if $d(XY)>r$ it is clear that XY is not lying on any minimum augmenting path and thus it is removed from the graph.

If $d(XY)<r$ then the augmenting path $p(0\to X)\cdot p(0\leftarrow Y)$ can not be legal but must contain an odd loop on which XY is lying. We refer to a loop which is formed by an illegal augmenting path whose length is less than r as a small loop. An edge XY is lying on a small loop if and only if $d(XY)<r$.

Let XY be an edge lying on a small loop and let M be the base of this loop. We shall show that if a minimum augmenting path is passing through XY then it must pass also through M. Thus, if X_1Y_1 and X_2Y_2 are two edges lying on two small loops whose common base is M, and if there are two minimum augmenting paths such that one is passing through X_1Y_1 and the other is passing through X_2Y_2 then these paths are

not disjoint but have M as a common vertex. Therefore, for the purpose of finding a set of disjoint minimum augmenting paths we can think of the two edges X_1Y_1 and X_2Y_2 as if they are identical with the base M, or as if they are "shrunk" to the vertex M. Thus, the basic idea of the second stage is to shrink each vertex X which is lying on a small loop to the base of this loop. But since this statement may be ambiguous (X may lie on two or more different small loops) we use the following definitions (here we assume that edges XY for which $d(XY)>r$ have already been removed).

The base of a vertex. Let X be a vertex and let Y be its mate. Define:

$$\text{BASE}(X)=\begin{cases} X & \text{if } d(XY)=r \\ \text{BASE}(V) & \text{if } d(XY)<r \text{ and } \text{LINK}(X)=V \\ \text{BASE}(P) & \text{if } d(XY)<r \text{ and } \text{LINK}(X)=PQ \end{cases} \tag{2.2}$$

Lemma 2.1: Let XY be a matched edge such that $d(X,Y)<r$ and X has a 2-link, $\text{LINK}(X)=PQ$. Then $d(PQ)=d(XY)$ and for every edge ST on the loop formed by $p(0\to P)\cdot p(0\leftarrow Q)$, $d(ST)\leq d(PQ)$.

Proof: By eq. (1.9).

Corollary 2.1: By lemma 2.1 and by induction on $\ell(X)$, one can prove the following properties:

(a) $M=\text{BASE}(X)$ if and only if M is the last vertex on $p(0\to X)$ for which $d(M,\text{MATE}(M))=r$; also $p(0\to M)$ is the longest head of $p(0\to X)$ for which $d(M,\text{MATE}(M))=r$.

(b) If $d(XY)<r$ than $\text{BASE}(X)=\text{BASE}(Y)$ and $\text{BASE}(X)$ is the base of least level among the bases of small loops which include X; also, all vertices on $p(\text{BASE}(X)\to X)$ have the same base.

The reduced graph: Let G(M) be the subgraph of the original graph G(V,E) defined as follows: The set of vertices V_M is given by $V_M=\{X \mid \text{BASE}(X)=M\}$ and the set of edges is given by $E_M=\{XY \mid X,Y\in V_M \text{ and } XY\in E\}$.

The basic idea of the second stage is to shrink G(M) into M where all edges of E_M are deleted and if parallel edges between M and M' are created, all but one are deleted. The resulting graph is called the reduced graph and is denoted by $\tilde{G}$.

(Notice that the process of shrinking the graph in our algorithm resembles the process of blossoms' shrinking performed by Edmonds [1]. But there are some distinctions between the two processes: Edmonds shrinks each loop when it is discovered. In our algorithm only those loops which cause troubles, namely the small loops, are shrunk. The shrinking in Edmonds' algorithm is performed separately for each loop, thus each vertex may participate in many shrinks, while in our second stage each vertex and each edge is shrunk only once and our bookkeeping is much simpler.)

The algorithm of the second stage.

1. (Initialization). $j\leftarrow 0$; $\text{VLIST}\leftarrow\phi$ (VLIST is the list of all vertices of $\tilde{G}$).

FIND VERTICES OF $\tilde{G}$:

2. (Search the vertices of LIST(j)). If all vertices of LIST(j) have been searched go to 4; else let X be a vertex in LIST(j) which has not been searched yet.

3. If BASE(X) is undefined return to 2. Else define BASE(X) by (2.2). If $\text{BASE}(X)=X$ add X at the end of VLIST and open a new list BASLIST(X) and put X as its only item. (BASLIST(X) is the list of all vertices V in G such that $\text{BASE}(V)=X$). Else, add X to the end of BASLIST(BASE(X)). Return to 2.

4. If $j=r$ go to 5 (VLIST contains all vertices of $\tilde{G}$ listed in nondecreasing order of their levels). Else assign: $j\leftarrow j+1$ and return to 2.

FIND EDGES OF $\tilde{G}$:

5. (Search the vertices of VLIST). If all vertices of VLIST have been searched go to 8; else let M be the first vertex in VLIST which has not been searched yet. Open a list $\text{ELIST}(M)=\phi$ (ELIST(M) is the list of all edges of $\tilde{G}$ which are incident to M). Assign $V\leftarrow M$.

6. (Search the edges VU in G). If all the edges of G which are incident to V have been searched go to 7; else let VU be an edge of G which has not been searched yet. If $\text{BASE}(U)=M$ repeat 6 (VU reduces to a self-loop). If the last edge in ELIST(BASE(U)) is (BASE(U),M) repeat 6 (VU reduces to an edge which is parallel to another edge in $\tilde{G}$). Else, add the edge (BASE(U),M) at the end of ELIST(BASE(U)). Assign: $\text{SOURCE}(\text{BASE}(U),M)\leftarrow UV$ and repeat 6.

7. If V is the last element of BASLIST(M) then return to 5; else, let V_1 be the next element on BASLIST(M). Assign: $V\leftarrow V_1$ and return to 6.

8. (The vertices of $\tilde{G}$ are listed in VLIST according to nondecreasing order of their levels. For each vertex M in VLIST the edges MN of $\tilde{G}$ are listed in ELIST(M) according to nondecreasing order of the levels of N, i.e. according to nondecreasing order of their values). For each vertex M in VLIST invert the order of the edges in ELIST(M), (these edges are now listed in non-increasing order of their values) and proceed to the third stage.

Validity and complexity of the second stage.

The Complexity. The second stage consists in fact of two parts, where in the first part (steps 2-4) we search the vertices of G and in the second part we search its edges (steps 5-7). The number of steps per each vertex and edge is constant, thus the complexity of the second stage is $0(m+n)$.

The validity. Lemma 2.2: The set of the matched edges in $\tilde{G}$ is exactly the set of the matched edges UV, of G, such that $d(UV)=r$.

Corollary 2.2: The set of the exposed vertices in $\tilde{G}$ is exactly the set of the exposed vertices in G which lie at the ends of minimum augmenting paths.

Proof: Recall that an exposed vertex A which lies in G at the end of a minimum augmenting path is represented by a matched edge $A^{(1)}A^{(2)}$ where $\ell(A^{(1)})=0$ and $\ell(A^{(2)})=r$, thus $d(A^{(1)}A^{(2)})=r$.

Lemma 2.3: Let XY be a free edge in $\tilde{G}$ and let UV be its source in G. Then UV is a free edge in G such that $d(UV)=r$.

Proof: UV is a free edge in G. If $d(UV)<r$ then BASE(U)=BASE(V) and UV would not be a source of an edge in $\tilde{G}$.

Lemma 2.4: Let XY be a free edge in G and UV its source. Then the path $p(X\to U)\cdot p(Y\leftarrow V)$ is an alternating path in G which contains $\ell-1$ matched **edges** and ℓ free edges where $\ell=r-\ell(XY)$.

Proof: The path $p(X\to U)\cdot p(Y\leftarrow V)$ is alternating (by the definition) and the two edges at its ends are free (by the definition of X=BASE(U)). Thus it contains $\ell-1$ matched edges and ℓ free edges. The length of the segment $p(X\to U)$ is $\ell(U)-\ell(X)$, i.e. it contains $\ell(U)-\ell(X)$ free edges. The segment $p(Y\leftarrow V)$ contains $\ell(V)-\ell(Y)$ free edges. Including the edge UV there are $\ell=\ell(U)-\ell(X)+\ell(V)-\ell(Y)+1$ free edges thus $\ell=\ell(UV)+1-\ell(XY)=d(UV)-\ell(XY)=r-\ell(XY)$.

QED

Corollary 2.3: Let π be a legal augmenting path in $\tilde{G}$ which connects the exposed vertices A and B. Replace each free edge XY on π by the path $p(X\to U)\cdot p(Y\leftarrow V)$ where UV=SOURCE(XY). Then the result is a legal augmenting path in G which connects A and B (this augmenting path is referred to as the augmenting path corresponding to π.).

Proof: By Lemma 2.4 the resulting path is an augmenting path. If X_1 and X_2 are two different vertices on π, then $p(X_1\to U_1)\in G(X_1)$ while $p(X_2\to U_2)\in G(X_2)$, thus these segments are disjoint and the augmenting path in G is also legal.

Lemma 2.5: Let X be a vertex f-lying on an augmenting path $\pi(A,B)$ in $\tilde{G}$. Then the length of the segment $p(A\to X)$ on the corresponding augmenting path in G is $\ell(X)$.

Proof: By induction on $\ell(X)$, and Lemma 2.4.

Corollary 2.4: Let π be a legal augmenting path in $\tilde{G}$. Then the corresponding augmenting path in G is legal and minimum.

Corollary 2.5: Any set of disjoint augmenting paths in $\tilde{G}$ corresponds to a set of disjoint minimum augmenting paths in G.

Theorem 2.1: Let $p(A\to B)$ be a minimum augmenting path in G which connects the exposed vertices A and B and passes through a vertex W. Then $p(A\to B)$ must also pass through BASE(W), and all the vertices on $p(A\to B)$ between W and BASE(W) belong to G(BASE(W)).

Proof: Let us outline the proof. Assume that $p(A\to B)$ is not passing through M=BASE(W). Let X be the first vertex on $p(A\to B)$ such that BASE(X)=M, then $X\neq M$. Let Y be the vertex which precedes X on $p(A\to B)$, then YX is a free edge. Similarly let T be the last vertex on $p(A\to B)$ such that BASE(T)=M and let S be the vertex which follows T on $p(A\to B)$, then TS is a free edge. Since $BASE(Y)\neq M$ thus $d(XY)=r$. Similarly $d(TS)=r$.

Denote the length of a path p in G by $|p|$. Assume that $|p(X\to B)|>\ell(X)$. Since $|p(A\to B)-p(X\to B)|\geq\ell(Y)+1$ then

$|p(A\to B)|\geq\ell(Y)+1+|p(X\to B)|>\ell(X)+\ell(Y)+1=d(XY)=r$ (a contradiction). Therefore $|p(X\to B)|=\ell(X)$.

By concatenating $p(X\to B)$ to $p(0\to MATE(X))$ we find that $p(0\to M)$ and $p(X\leftarrow B)$ have a common head. Similarly $p(0\to M)$ and $p(A\to B)-p(T\to B)$ have a common head. Thus $p(A\leftarrow B)$ and $p(A\to B)$ have a common head which leads to contradiction.

Corollary 2.6: Let p be a minimum augmenting path in G. Then by the shrinking process it reduces to an augmenting path in $\tilde{G}$.

Corollary 2.7: Let p_1 and p_2 be two disjoint minimum augmenting paths in G. Then their reductions are two disjoint augmenting paths in $\tilde{G}$.

Corollary 2.8: Let S be a maximal set of disjoint augmenting paths in $\tilde{G}$. Then the set of the corresponding disjoint minimum augmenting paths in G is also maximal.

Thus instead of looking for a maximal set of disjoint minimum augmenting paths in the original graph G, we look for a maximal set of disjoint augmenting paths in the reduced graph $\tilde{G}$. This is done in the third stage.

3. The Third Stage

General description.

The goal of the third stage is to find a maximal set of disjoint augmenting paths in the reduced graph. This is achieved by performing a search of the free edges in the graph, similar to the process of the first stage, trying to build alternating paths leading to the vertices. (We denote such an alternating path which leads to the vertex Z by $rp(0\to Z)$). In order to record those paths we use the same method as used in the first stage, namely the 1-link and the 2-link. However, while in the first stage the length of each alternating path must be minimum, here it is arbitrary. Thus, we do not define levels for the vertices but use the levels which were defined in the first stage (by Lemma 2.5 these levels are in fact the places of vertices on the corresponding augmenting paths). The fact that the alternating paths, in the reduced graph, may not be minimum, enables to check the legality of a 2-link before it is defined. In fact, when we identify a bridge XY we search immediately both paths $rp(0\leftarrow X)$ and $rp(0\leftarrow Y)$, assigning a 2-link (YX or XY) to the vertices along the paths which have no link yet. This search which is called a loop-search is carried out simultaneously on both paths, and it is done step by step according to the levels which were defined in the first stage. As in the first stage, we use the

variable TOP (denoted here by RTOP) in order to avoid repeated search of an edge; when we meet a vertex Z we skip to RTOP(Z) to continue the search from there. We shall show later that if XY is a bridge then $rp(0\rightarrow X)\cdot rp(0\leftarrow Y)$ contains a loop. Let M be the base of this loop. As one can see, by performing the loop-search we shall always reach RTOP(MATE(M)) in the same step from both directions of the loop. This terminates the loop-search and no more 2-links which point to the bridges XY or YX are assigned. Therefore, when a vertex is given a link (either 1-link or 2-link) we are sure it is well-defined and the case of a potential link does not happen in the third stage. The search of the edges in the third stage always proceeds from a low level to a higher one, as one can conclude from the following Lemma:

Lemma 3.1: Let XY be a free edge in the reduced graph and Z=MATE(Y). Then $\ell(Z)>\ell(X)$.

Proof: Let UV=SOURCE(XY) (see second stage). Then $\ell(X)\leq\ell(U)$, $\ell(Y)\leq\ell(V)$ and thus $\ell(X)+\ell(Y)\leq\ell(U)+\ell(V)=d(UV)-1<r=d(YZ)=\ell(Y)+\ell(Z)$. Therefore $\ell(X)<\ell(Z)$.

QED

Corollary 3.1: Let π be an alternating path which connects the vertices X and S in the reduced graph such that X is incident to a free edge of π and S is incident to a matched edge of π. Then $\ell(S)>\ell(X)$.

HLFS method. In the first stage we wanted to find minimum augmenting paths and thus used the BFS method. In the third stage the requirement is to find disjoint paths and for that purpose we use the DFS method (Depth First Search) or rather the HLFS method (Highest Level First Search): Assume that the edge XY is being searched. If this search is not a loop-search then X is a vertex for which a link has already been defined and it is the vertex with the highest level such that not all the free edges XY have already been searched. Assume that the search started from a certain 1-exposed vertex A. The HLFS means that we shall not search an edge BV (where B is another 1-exposed vertex) until we either find an augmenting path emerging from A or find that such a path does not exist. Thus, the search of the edges is performed in parts, where in the i-th part we try to find an augmenting path emerging from the i-th 1-exposed vertex. As we shall see, all vertices and edges which have been searched during the i-th part are not relevant any more to the following parts, and they are ignored. This fact assures that the set of augmenting paths found in the third stage is really maximal, and that no edge must be searched more than once (in each direction). Thus, the cost of the search of the edges is $0(m_1)$ where m_1 is the number of edges of the reduced graph.

Because of the HLFS method and the search of the edges by parts, when a bridge XY is searched, both $rp(0\rightarrow X)$ and $rp(0\rightarrow Y)$ emerge from the same 1-exposed vertex and therefore $rp(0\rightarrow X)\cdot rp(0\leftarrow Y)$ must contain a loop. Thus a loop-search always determines a loop, as we noted above. Let M be the base of this loop. After the loop-search is completed, all the vertices which lie on the loop will have the same RTOP as MATE(M). thus, for each vertex X which lies on the loop the following relation holds: RTOP(X)=RTOP(MATE(X)). A new meaning can be given to the RTOP variable, namely: RTOP(X) is the mate of the vertex of minimum level among all the bases of loops on which X is lying and which have already been searched. Since the search is performed in the HLFS method we can also define MATE(RTOP(X)) as a vertex which is f-lying on $rp(0\rightarrow X)$ whose level is the highest such that its mate has no link yet. As in the first stage we define RT(V) to be the set of all vertices Z such that RTOP(Z)=V. Also, as in the first stage, n operations of merging sets $RT(V_1)$ and $RT(V_2)$ may be required. However, because we are not interested in a minimum path, no operations of finding a minimum in RT(V) or changing the value of a vertex are required (the term "value of a vertex" is not even defined here). Thus, an efficient algorithm for handling the operations on the class of sets {RT(V)} have a complexity of $0(n_1 g(n_1))$ where n_1 is the number of vertices of the reduced graph and $g(n_1)$ is the reciprocal Ackerman function [6]. The complexity of the third stage is therefore $0(m_1+n_1 g(n_1))$.

The data structure. From the second stage we have VLIST which contains the vertices of the reduced graph ordered according to their levels. For each of these vertices Z we have a list ELIST(Z) of the edges which are incident in the reduced graph to Z, where these edges are listed in non-increasing order of their values (i.e., in non-increasing order of their other end points). For each vertex Z we use an additional variable called LIFE(Z): In the beginning of the third stage for each vertex Z, LIFE(Z)=n+1. If vertex Z is reached during the search of the i-th part we assign LIFE(Z)=i. Thus, during the search of the i-th part we ignore all vertices Z for which LIFE(Z)<i.

In order to handle the HLFS we define for each of the levels j=0,1.....r-1,r a list RLIST(j), which contains all vertices Z whose level is j, Z has already been reached but not all free edges ZW have been searched yet.

The algorithm of the third stage.

The algorithm of the third stage consists of a main procedure and a subroutine SRCHLOOP which implements the loop-search. As one can see, the following algorithm is in fact a slight improvement of Gabow's algorithm [3].

The main procedure.

1. (Initialization): For each vertex Z in the reduced graph assign: RLINK(Z)$\leftarrow$0; LIFE(Z)$\leftarrow$n+1; RT(Z)={Z}, RTOP(Z)=Z. For each level j(j=0,1...r-1,r) assign RLIST(j)$\leftarrow\phi$. Assign: $i_0\leftarrow 0$.

NEW-PART:

2. Assign $i_0\leftarrow i_0+1$ (The i_0-th part is started). Assign: X$\leftarrow$VLIST(i_0). If $\ell(X)\neq 0$ then proceed to the fourth stage (all the 1-exposed vertices of the reduced graph have already been searched). If LIFE(X)<n+1 return to NEW-PART (the vertex X has already been reached in a former part).

3. Assign: j$\leftarrow$0 (j is the level of the vertex from which the search will continue).

Insert X to RLIST(0). Assign: LIFE(X)$\leftarrow i_0$ LIFE(MATE(X))$\leftarrow i_0$.

4. (Search from vertex X). If all the edges in ELIST(X) have already been searched then remove X from RLIST(j) and go to 6; else let XY be the first edge in ELIST(X) which has not been searched yet. If LIFE(Y)$<i_0$ then repeat 4; else let Z=MATE(Y). If Z=X (the edge XY is matched) repeat 4; else assign: LIFE(Y)$\leftarrow i_0$ LIFE(Z)$\leftarrow i_0$. If RLINK(Y)$\neq 0$ call SRCHLOOP(X,Y) and go to 6.

5. (Search of the edge XY where RLINK(Y)=0). If RLINK(Z)$\neq 0$ return to 4; else assign: RLINK(Z)$\leftarrow$X. j$\leftarrow \ell$(Z). Insert Z to RLIST(j). If j=r return to NEW-PART (the present part is successfully terminated); else assign: X$\leftarrow$Z and go to 4.

6. (Find a new vertex whose level is j to continue the search). If RLIST(j)$=\phi$ go to 7; else let X be the first vertex in RLIST(j). If LIFE(X)$<i_0$ then remove X from RLIST(j) and repeat 6; else return to 4.

7. (For all vertices X whose level is j, all the free edges XY have already been searched. Find a new level to continue the search). If j=0 go to NEW-PART (the present part is terminated without finding an augmenting path: There is no augmenting path passing through vertices whose LIFE is i_0); else assign: j$\leftarrow$j-1 and return to 6.

The subroutine SRCHLOOP(X,Y):

1. Assign: $U_1\leftarrow$RTOP(MATE(X)); $U_2\leftarrow$RTOP(MATE(Y)). (U_1 and U_2 are the new candidates on rp(0$\leftarrow$X) and on rp(0$\leftarrow$Y) to get a 2-link to YX and XY respectively). If $U_1=U_2$ then return to the main procedure (the edges of the loop formed by the bridge XY have already been searched).

2. Assign $V_1\leftarrow$Y, $V_2\leftarrow$X (V_1 and V_2 are the last vertices f-lying on rp(0$\leftarrow$X) and on rp(0$\leftarrow$Y) respectively which have been reached so far during the loop-search).

3. Assign: $A\leftarrow \begin{cases} U_1 & \text{if } \ell(U_1)\leq \ell(U_2) \\ U_2 & \text{if } \ell(U_2)<\ell(U_1) \end{cases}$ (A is a vertex of minimum level on the loop formed by XY whose RLINK is still undefined). (Give a 2-link to A): Assign: $\text{RLINK}(A)\leftarrow \begin{cases} XY & \text{if } A=U_2 \\ YX & \text{if } A=U_1 \end{cases}$; $j_1\leftarrow\ell(A)$ (j_1 is the highest level of a vertex which has been linked through the present activation of SRCHLOOP). Insert A to RLIST(j_1).

4. (Update the sets {RT(V)}). Assign: $B\leftarrow \begin{cases} V_1 & \text{if } A=U_1 \\ V_2 & \text{if } A=U_2 \end{cases}$ (A was reached by searching from B). Merge: RT(A)$\leftarrow$ RT(A)$\cup$RT(B). Assign RT(B)$\leftarrow\phi$.

5. (Update A and B). Assign: B$\leftarrow$A: A$\leftarrow$RTOP(MATE(RLINK(MATE(A))))). If $B=U_1$ assign: $V_1\leftarrow$B, $U_1\leftarrow$A; else assign: $V_2\leftarrow$B, $U_2\leftarrow$A;

6. If $U_1\neq U_2$ return to 3; else ($U_1=U_2$ is the RTOP of the whole loop formed by the bridge XY) assign: j$\leftarrow\max(j,j_1)$ and return to the main procedure.

(Remark: Let M be the base of the loop formed by the bridge XY. It can be shown as a result of the following Lemma 3.2 and Lemma 3.3 that when SRCHLOOP(X,Y) is activated, all the vertices on rp(M$\rightarrow$Y) already have a RLINK, and RTOP(MATE(Y))= RTOP(MATE(M)). Thus, the loop-search is actually performed only on rp(M$\leftarrow$X) and in step 3 of SRCHLOOP we always assign A$\leftarrow U_1$, RLINK(A)$\leftarrow$YX, in step 4 we always assign B$\leftarrow V_1$ and in step 5 we always assign $V_1\leftarrow$B, $U_1\leftarrow$A. Therefore SRCHLOOP(X,Y) can be simplified).

The validity of the third stage

As we already mentioned, the algorithm of the third stage is in fact a slight improvement of Gabow's algorithm [3]. Therefore we shall not prove here that the links which are assigned in the third stage are really defining legal augmenting paths. We only show here that vertices whose LIFE is less than i are not relevant to the i-th part; i.e. we show that there is no augmenting path which is disjoint from previously discovered augmenting paths and which uses vertices whose LIFE is less than i.

Let X be a vertex. If RLINK(X) is not defined yet we call X a passive vertex. If X is included in RLIST(ℓ(X)) we call X an active vertex. If X has already been removed from RLIST(ℓ(X)) we call X a dead vertex.

Lemma 3.2: Let X be an active vertex, MATE(X) be a passive vertex and Z be an active vertex such that $\ell(Z)\geq\ell(X)$ and LIFE(Z)=LIFE(X). If the procedure is not in SRCHLOOP, then the alternating path rp(0$\rightarrow$Z) defined by RLINK(Z) passes through X. (If $\ell(Z)=\ell(X)$ then Z=X).

The proof is by contradiction. Assume that Z is the first vertex which has become active for which the Lemma does not hold. It can be shown that if RLINK(Z)=W then X is lying on rp(0$\rightarrow$W) and thus on rp(0$\rightarrow$Z); and if RLINK(Z)=PQ then X is lying on rp(0$\rightarrow$M), where M is the base of the loop which is on rp(0$\rightarrow$P)$\cdot$rp(0$\leftarrow$Q), and thus X is lying on rp(0$\rightarrow$Z).

Corollary 3.2: Let LIFE(X)=i. If at the end of the i-th part X is still active while MATE(X) is passive, then an augmenting path which passes through X has been found in the i-th part.

Proof: X is active at the end of the i-th part, thus this part has terminated successfully and an augmenting path leading to a

2-exposed vertex $B^{(2)}$ has been found. By Lemma 3.2 X is on $rp(0 \to B^{(2)})$, i.e. X is on the augmenting path.

QED

Lemma 3.3: If X is dead while MATE(X) is still passive, then MATE(X) will remain passive.

Proof: X has a 1-link, thus MATE(X) can have only 2-link. But if RLINK(MATE(X))=PQ then X is f-lying on rp(0→Q) and thus by Corollary 3.1 $\ell(Q)>\ell(X)$, and Q became dead before X. Therefore, QP is searched before X becomes dead. If at this time P has a link then MATE(X) would get a link while X is still not dead. If P has no link then QP and PQ are not bridges.

QED

Lemma 3.4: If X is dead, MATE(X) is passive and LIFE(X)=i, then there exists no augmenting path which passes through X and the LIFE of each of its vertices is greater than or equal to i.

Proof: Assume X is a vertex of highest level f-lying on an augmenting path π, which contradicts the Lemma. Let W be a vertex of highest level, f-lying on π, whose LIFE has become i while X has been active. Let WT be the free edge on π and let U=MATE(W), S=MATE(T). When X becomes dead neither W nor U can be passive: For, let V be the vertex among W and U which has 1-link; by the choice of W, when RLINK(V) is assigned X is active and thus $\ell(V)>\ell(X)$. Therefore V becomes dead before X and by the assumptions on X, MATE(V) cannot be passive. By Corollary 3.1 $\ell(W)>\ell(X)$ and thus W becomes dead before X and WT is searched while X is still active.

Now let us show that, when WT is searched, T cannot be passive. By Corollary 3.1 $\ell(S)>\ell(W)>\ell(X)$. Assume that when WT is searched T is passive. If S is passive then by searching WT LIFE(S) becomes i, contradicting the choice of W. If S is active, since $\ell(S)>\ell(W)$ we cannot be searching WT. If S is dead then the choice of X is wrong. Thus, when WT is searched T already has a link. This implies (by the assumption on π) that when WT is searched LIFE(T)=LIFE(S)=i. By the choice of W, LIFE(S) and LIFE(T) has become i before X becomes active and therefore X is not on rp(0→T). Thus WT is a bridge which forms a loop which contains MATE(X) and this contradicts the assumption that MATE(X) is passive.

QED

Lemma 3.5: If X is active, $\ell(X)\geq\lceil\frac{r}{2}\rceil$, then rp(0→X) passes through all the active vertices V for which $\ell(X)\geq\ell(V)\geq\lceil\frac{r}{2}\rceil$ and LIFE(V)=LIFE(X).

Proof: Assume that X is the first vertex which becomes active for which the Lemma does not hold. First we show that when X becomes active it is in accord with the Lemma: If RLINK(X)=W then the Lemma holds for W and thus for X. Assume RLINK(X) has been assigned during SRCHLOOP(P,Q). Let V be a vertex which is active when SRCHLOOP(P,Q) is activated, $\ell(V)\geq\lceil\frac{r}{2}\rceil$ and LIFE(V)=LIFE(X). Then, $\ell(P)\geq\ell(V)$ and V is on rp(0→P). If RLINK(X)=PQ then V is also on rp(0→X). Assume RLINK(X)=QP and V is not on rp(X→P). In case V is f-lying on rp(0←P) then (by corollary 3.1) $\ell(V)>\ell(X)$ and in case V is f-lying on rp(0→P) then $\ell(V)<\ell(MATE(X))\leq\lceil\frac{r}{2}\rceil$. Therefore if $\ell(X)\geq\ell(V)\geq\lceil\frac{r}{2}\rceil$ then V is on rp(X→P) and thus on rp(0→X). If V becomes active during SRCHLOOP(P,Q) and $\ell(X)\geq\ell(V)$ then V is on rp(0→X).

Next, let V be a vertex for which RLINK(V) is assigned while X is already active, LIFE(V)=LIFE(X) and $\ell(X)\geq\ell(V)\geq\lceil\frac{r}{2}\rceil$. It follows that V cannot get a 1-link and thus MATE(V) has a 1-link. By Lemma 3.3 MATE(V) is still active when RLINK(V) is assigned. $\ell(MATE(V))\leq\lceil\frac{r}{2}\rceil\leq\ell(X)$ and thus by Lemma 3.2 MATE(V) (and V) are lying on rp(0→X).

QED

Corollary 3.3: If during the i-th part an augmenting path is found, then it passes through all vertices V such that LIFE(V)=i, $\ell(V)\geq\lceil\frac{r}{2}\rceil$ and V is active at the end of the i-th part.

Theorem 3.1: Let X be a vertex such that LIFE(X)=i. Then there exists no augmenting path which passes through X and is disjoint from all augmenting paths which have been found in the first i parts.

Proof: By induction on i: Assume X is a vertex of highest level for which the Lemma does not hold. Let us show that at the end of the i-th part both X and MATE(X) have a link. For if one of them is passive then the other cannot be dead by Lemma 3.4, and cannot be active, by Corollary 3.2. By the assumption on X, $\ell(X)\geq\ell(MATE(X))$ and thus $\ell(X)\geq\lceil\frac{r}{2}\rceil$. Thus, by Corollary 3.3 X must be dead at the end of the i-th part. Let XY be the free edge on the augmenting path which is assumed to pass through X and let Z=MATE(Y). By the induction and the assumption on X, at the end of the i-th part LIFE(Y)=LIFE(Z)>i, a contradiction to the fact that XY was searched.

QED

Corollary 3.4: In the i-th part, all the vertices X such that LIFE(X)<i may be ignored.

4. The Fourth Stage

In the fourth stage the augmenting paths which have been found in the third stage are retraced and the matching is improved by converting every free edge which is lying on these paths to a matched one and vice versa. The 2-exposed vertices at the ends of those augmenting paths, are removed from the graph.

The fourth stage is performed as follows: We search the list RLIST(r) which contains all the 2-exposed vertices which are lying at the ends of the augmenting paths which have been found in the third stage. Let B be a vertex in RLIST(r) and let i=LIFE(B). Then the origin of the augmenting path leading to B is a 1-exposed vertex A, which is the i-th vertex

in VLIST. In order to perform the restoration of the whole augmenting path which leads from A to B in the reduced graph we use a recursive procedure RBUILD(W,Z,k) (where k=0,1). This procedure returns an alternating path from W to Z, where W is incident to a free edge of the path and Z is incident to a matched edge of the path. The alternating path from W to Z is a segment of the whole required augmenting path from A to B where their directions coincide if k=1 and are opposite if k=0. Thus in order to retrace the augmenting path from A to B we call RBUILD(A,B,1).

The algorithm of RBUILD(W,Z,k):

1. If W=Z then return Z.

2. If RLINK(Z)=X then do;
   if k=1 then return
   RBUILD(X,W,1)·MATE(Z)·Z;
   else return Z·MATE(Z)·RBUILD(W,X,0).

3. If RLINK(Z)=PQ then do;
   if k=1 then return
   RBUILD(W,P,1)·RBUILD(MATE(Z),Q,0)·Z;
   else return
   Z·RBUILD(MATE(Z),Q,1)·RBUILD(W,P,0).

After the augmenting path from A to B in the reduced graph is retraced, each free edge XY on it must be replaced by the segment p(X→U)·p(Y←V) where UV=SOURCE(X,Y) (see second stage). The paths p(X→U) and p(Y←V) are retraced by the recursive procedure BUILD(W,Z,k) which is similar to RBUILD(W,Z,k) but uses LINK(Z) instead of RLINK(Z). This completes the restoration of the minimum augmenting path from A to B in the original graph. The vertices on this path are rematched and we proceed to the next augmenting path.

During the restoration of an augmenting path, each of the procedures RBUILD and BUILD may be called at most r times and the number of steps per each call is constant. Thus, the number of steps performed in restoring one augmenting path is proportional to r and the complexity of the whole fourth stage is O(n).

5. References

1. Edmonds, J., "Paths, Trees and Flowers"; Canadian J. 1965, Volume 17, pp. 449-467.

2. Witzgall, C. and Zahn, C. T., Jr. "Modification of Edmonds' Maximum Matching Algorithm"; Journal of Research of the National Bureau of Standards, Jan-June 1965, Volume 69B, pp. 91-98.

3. Gabow, H., "An Efficient Implementation of Edmonds' Maximum Matching Algorithm"; June 1972, Technical Report No. 31, Stan-CS, pp. 72-328.

4. Berge, C. "The Theory of Graphs and its Applications"; (English translation from French), John Wiley and Sons Inc., New York; "Two Theorems in Graph Theory" Proceedings of the National Academy of Science 1957, Volume 43, pp. 842-844.

5. Hopcroft, J. E. and Karp, R. M. "An $n^{5/2}$ Algorithm for Maximum Matching in Bipartite Graphs"; SIAM J. on Comp. 2, December 1973, pp. 225-231.

6. Aho, A. V., Hopcroft, J. E., and Ullman, J. D., "The Design and Analysis of Computer Algorithms"; Addison-Wesley Publishing Company, 1974, Chapter 4, pp. 124-163.