

AN $O(n^{2.5})$ ALGORITHM FOR FINDING
A MAXIMUM MATCHING IN A
GENERAL GRAPH

Thesis of the Degree of
DOCTOR OF PHILOSOPHY

by

ODED KARIV

Submitted to the Scientific Council of the Weizmann Institute of Science

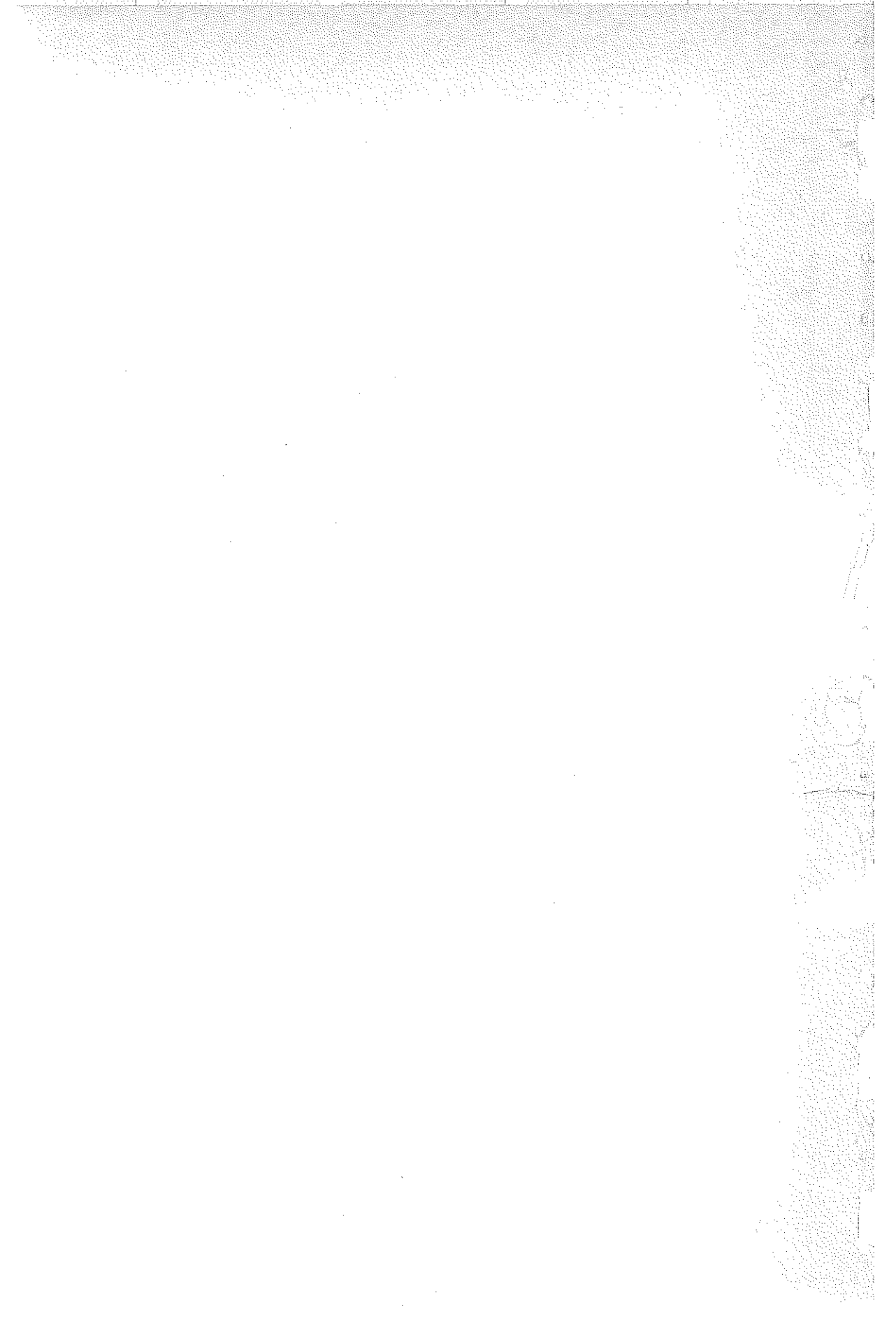
Rehovot

March 1976

SYSTEM NO. מערכת מ'ס'

20129-1

In the memory of my
friend MOSHE VITKIN
who fell during the
Yom Kippur war.



This work was carried out under the supervision of Professor Shimon Even, in the Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel.

Acknowledgements

I wish to express my gratitude to:

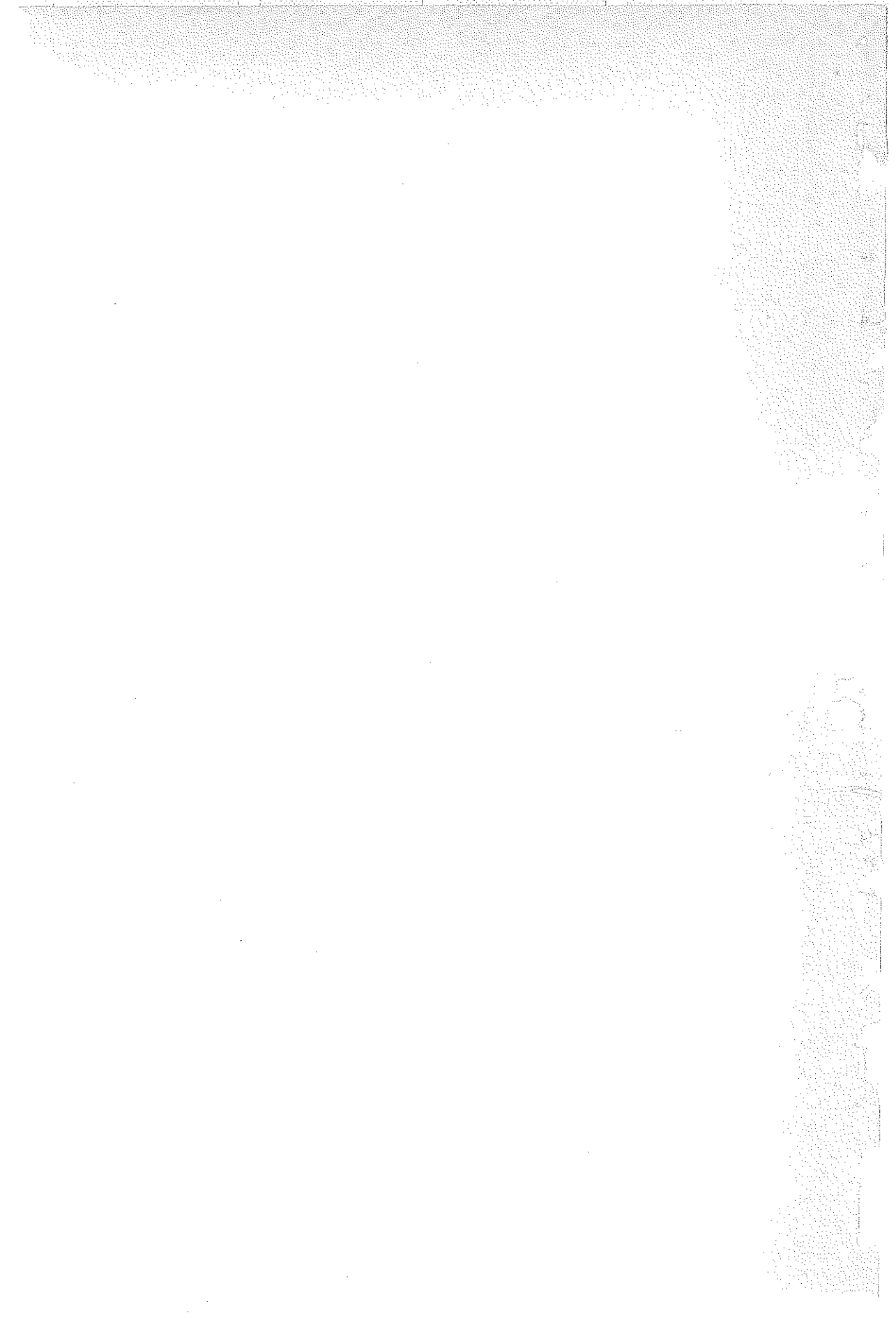
Prof. Shimon Even for introducing me to the subject, for helpful guidance during the course of this work and for critical reading of the manuscript.

Dr. Yehoshua Perl for helpful discussions.

Miss Sara Fliegelman for her devotion and patience while typing the manuscript.

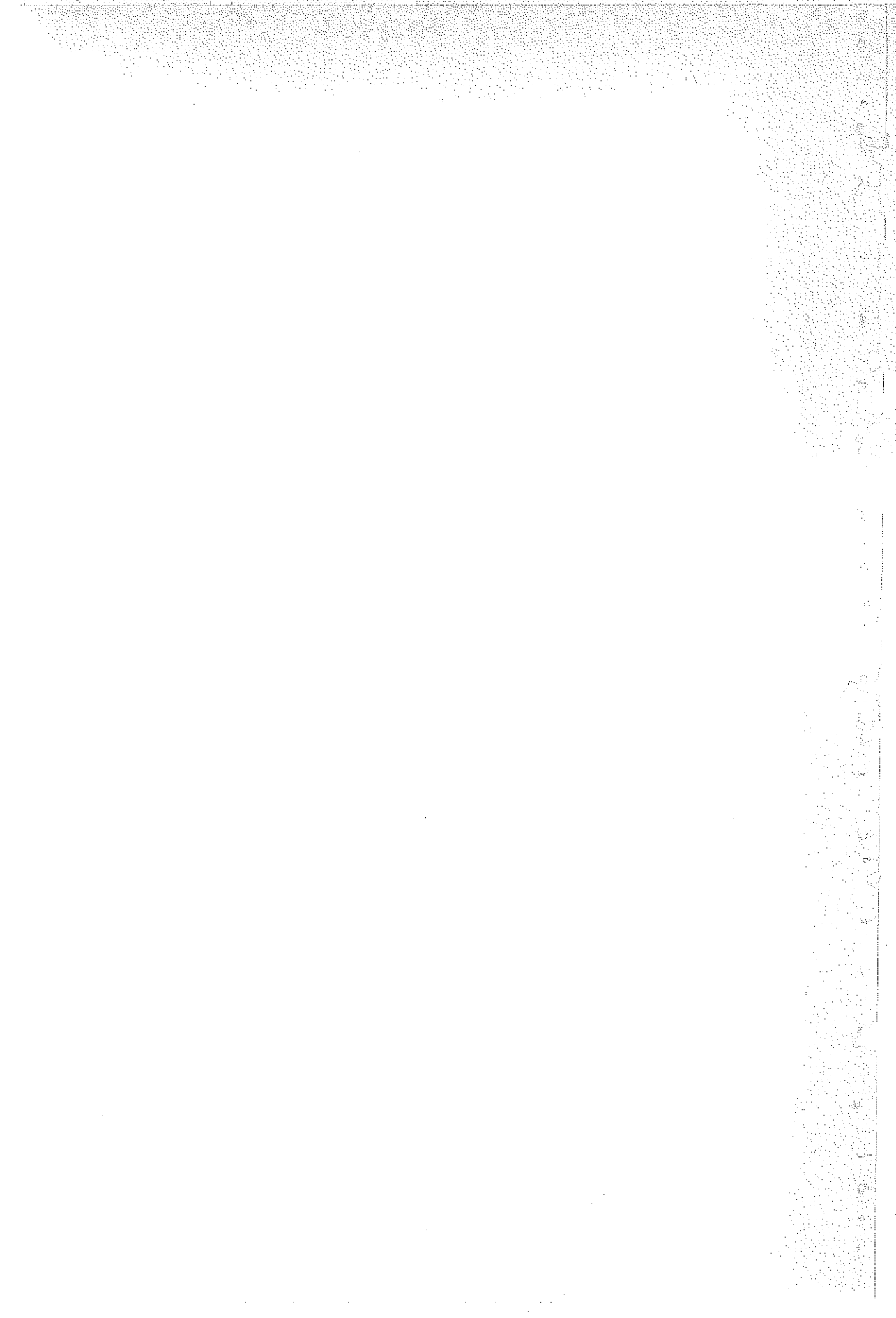
Mr. Yehuda Barbut for his dedicated work of drawing the figures.

Last, but not least, I wish to thank my parents whose love and support enabled this research.

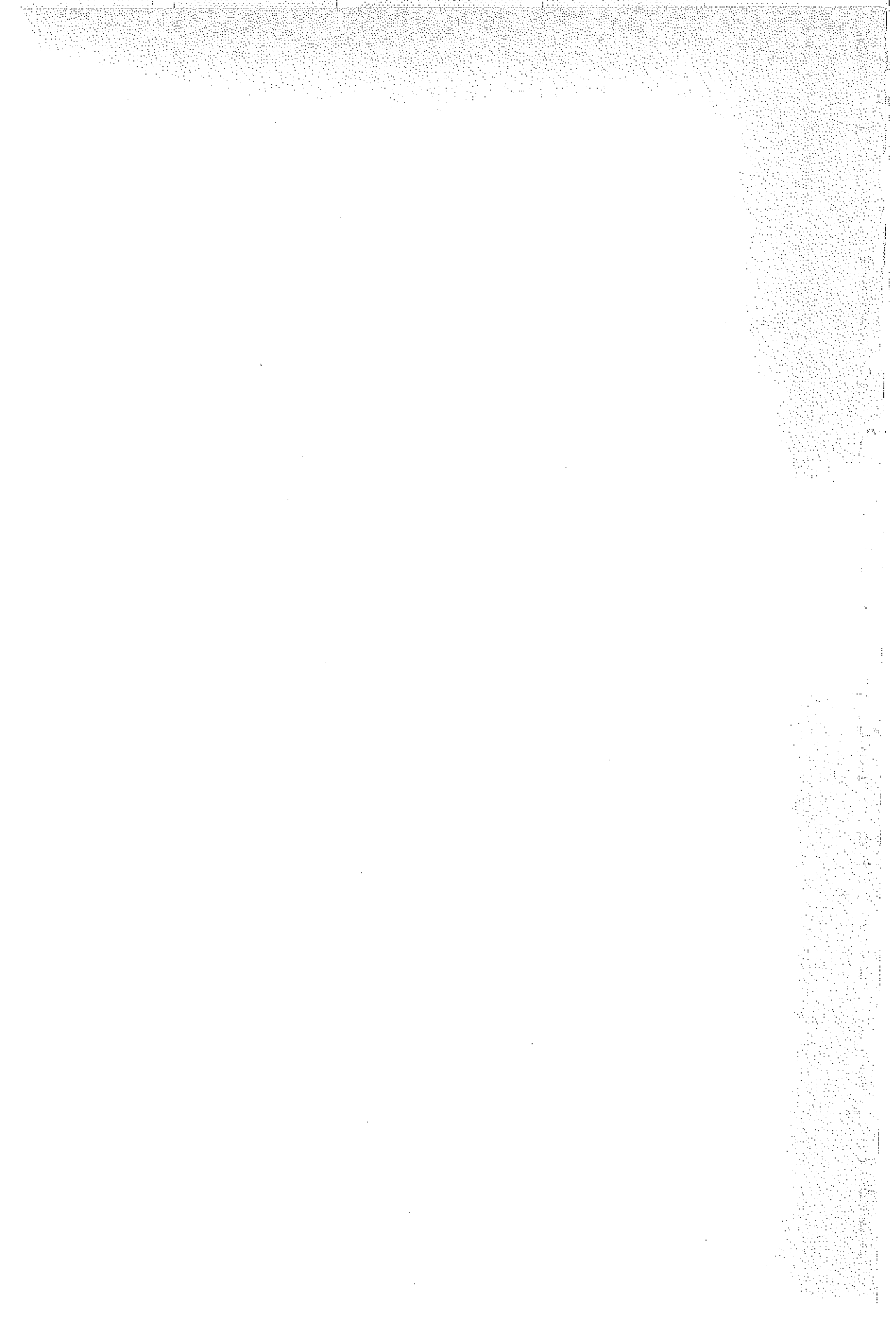


CONTENTS

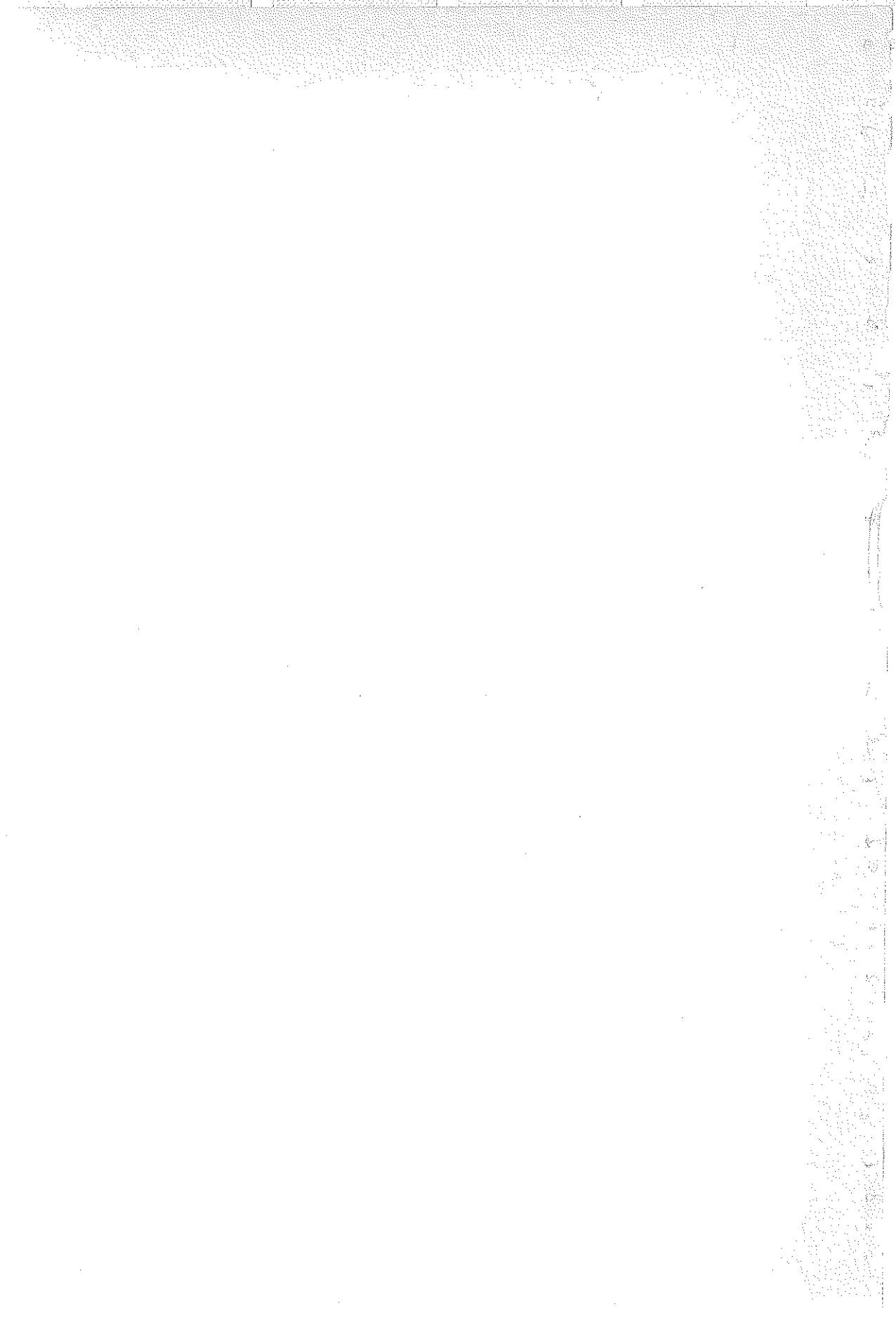
	<u>Page</u>
<u>INTRODUCTION</u>	2- 13
A. Definitions	3- 4
B. General Description	4- 8
C. Description of a Phase	8- 12
D. The Algorithm	12- 13
1. <u>THE FIRST STAGE</u>	14- 79
A. Notations	14- 16
B. The BFS Method	17- 19
C. The Data Structures	19- 33
D. Legality and Minimality	33- 39
E. The Performance of the j -th Substage	39- 45
F. The Algorithm	46- 48
G. The Correctness of the First Stage	49- 73
H. The Complexity of the First Stage and the Implementation of MERGE	73- 79
2. <u>THE SECOND STAGE</u>	80-111
A. General Description	80- 84
B. The BASE of a Vertex	85- 97
C. The Reduced Graph	98-106
D. The Data Structures	106-109
E. The Algorithm	109-110
F. The Complexity of the Second Stage	110-111



	<u>Page</u>
3. <u>THE THIRD STAGE</u>	112-147
A. General Description	112-113
B. The Parts of the Third Stage and the HLFS Method	113-117
C. The Subroutine SRCHLOOP	117-131
D. The Data Structures	131-134
E. The Algorithm	134-137
F. The Correctness of the Third Stage	137-144
G. The Complexity of the Third Stage	144-147
4. <u>THE FOURTH STAGE</u>	148-153
A. General Description	148-151
B. The Algorithm	151-152
C. The Correctness and the Complexity of the Fourth Stage	153
5. <u>IMPLEMENTATION OF THE ALGORITHM AND SPACE REQUIREMENTS</u>	154-167
A. Direct Implementation	154-163
B. Minimum-Space Implementation	163-167
<u>APPENDIX A</u> - Proof of Theorem 1.4(b)	168-193
<u>APPENDIX B</u> - Implementation 4 for Subroutine MERGE	194-201
<u>APPENDIX C</u> - PL/1 Program	202-209
References	210-211



	<u>Page</u>
Fig. A.4	177
Fig. A.5	181
Fig. A.6	185
Fig. A.7	190



Abstract. An efficient algorithm for finding a maximum matching in an arbitrary graph is presented. The algorithm is based on the result of Hopcroft and Karp, i.e., it is performed in phases where in each phase a maximal set of disjoint minimum legal augmenting paths is found and a process of augmentation is carried out along these paths. Each phase is performed in four stages: In the first stage minimum legal augmenting paths are found by using Breadth First Search. In the second stage the irrelevant vertices and edges are removed and the original graph is shrunk. During the third stage, a maximal set of disjoint arbitrary legal augmenting paths is found in the reduced graph, and a process of augmentation is performed on these paths in the fourth stage. The complexity of this algorithm is $O(n^{2.5})$. Other implementations of the data structures lead to complexities $O(m\sqrt{n} \lg n)$ or $O(m\sqrt{n} \lg \lg n)$ [where n and m are the number of vertices and edges of the graph, respectively], or to complexity $O(m\sqrt{n} + n^{1.5+\epsilon})$ [where ϵ is an arbitrary positive value].

INTRODUCTION

Let $G(V,E)$ be an arbitrary undirected graph, whose set of vertices is V , the set of the edges is E , and n and m are the cardinalities of V and E respectively. A matching M in G , is a subset of the edges of G 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 improved version 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. Other authors who solved the problem by $O(n^3)$ algorithms are Balinski [8], Kameda and Munro [9] and Lawler [10].

This work presents an algorithm which finds a maximum matching in general graphs by using at most $O(n^{2.5})$ steps. Some other implementations of the data structures lead to algorithms of complexity $O(m\sqrt{n} \lg \lg n)$ and $O(m\sqrt{n}n^{1.5+\epsilon})$ (where ϵ is an arbitrary positive value). A review of the $O(n^{2.5})$ algorithm has been published in [11].

A. Definitions

Let G be an arbitrary graph, and let M be a matching of this graph. In the following, 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:

A Matched [free] edge - an edge which belongs [does not belong] to the matching M .

An exposed vertex - a vertex which is not incident to any edge of M .

A matched vertex R - a vertex R which is incident to an edge of M (the other end of this edge is denoted by MATE(R)).

A path of G - a sequence of edges of the graph, such that every two consecutive edges have a common vertex. Generally (when it causes no confusion), we denote a path by the concatenation of its vertices.

An alternating path - a path whose edges are alternatingly matched and free. In particular: An alternating path leading to a (matched) vertex R - an alternating path whose first vertex is exposed, the last one is R , and the edge of the path which meets R is matched.

An augmenting path - an alternating path whose first and last vertices are exposed.

The length of an alternating path - the number of the free edges of the path.

A loop - an alternating path where each of its vertices meets exactly two of its edges. If the loop contains an odd number of edges (i.e., if it contains k matched edges and $k+1$ free edges), then we call it an odd loop; we refer to the (unique) vertex of an odd loop in which two free edges of the loop are adjacent, as the base of the loop. If the loop contains an even number of edges, then we call it an even loop (see Fig. 0.1).

An illegal alternating path - an alternating path which contains a loop.

B. General Description

All the algorithms known to us to find a maximum matching are based on Berge's theorem [4] which states that:

A given matching in a graph, is not maximum if and only if there exists in the graph a legal augmenting path (which connects two exposed vertices).

A. Definitions

Let G be an arbitrary graph, and let M be a matching of this graph. In the following, 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:

A Matched [free] edge - an edge which belongs [does not belong] to the matching M .

An exposed vertex - a vertex which is not incident to any edge of M .

A matched vertex R - a vertex R which is incident to an edge of M (the other end of this edge is denoted by MATE(R)).

A path of G - a sequence of edges of the graph, such that every two consecutive edges have a common vertex. Generally (when it causes no confusion), we denote a path by the concatenation of its vertices.

An alternating path - a path whose edges are alternatingly matched and free. In particular: An alternating path leading to a (matched) vertex R - an alternating path whose first vertex is exposed, the last one is R , and the edge of the path which meets R is matched.

An augmenting path - an alternating path whose first and last vertices are exposed.

The length of an alternating path - the number of the free edges of the path.

A loop - an alternating path where each of its vertices meets exactly two of its edges. If the loop contains an odd number of edges (i.e., if it contains k matched edges and $k+1$ free edges), then we call it an odd loop; we refer to the (unique) vertex of an odd loop in which two free edges of the loop are adjacent, as the base of the loop. If the loop contains an even number of edges, then we call it an even loop (see Fig. 0.1).

An illegal alternating path - an alternating path which contains a loop.

B. General Description

All the algorithms known to us to find a maximum matching are based on Berge's theorem [4] which states that:

A given matching in a graph, is not maximum if and only if there exists in the graph a legal augmenting path (which connects two exposed vertices).

If such a path exists, then by converting each of its free edges into matched and vice versa, we improve the matching (i.e., construct a new matching whose cardinality is greater by one). This process is called an augmentation.

A phase. J.E. Hopcroft and R.M. Karp [5] showed that if an algorithm for constructing a maximum matching is carried out in phases, when in each phase the augmentation is simultaneously performed on a maximal set of legal augmenting paths, where these paths are of minimum length and their vertices are disjoint - then the number of phases is at most \sqrt{n} . Thus, if we could find a maximal set of disjoint minimum legal augmenting paths in $O(n^2)$ steps - then we would have an $O(n^{2.5})$ algorithm for constructing a maximum matching in a graph. For the case of bipartite graphs, Hopcroft and Karp succeeded to build an algorithm which performs each phase in $O(m)$ steps, and thus achieved an algorithm which finds a maximum matching in $O(m\sqrt{n})$ steps. Yet, the construction of such an algorithm for the case of general graphs still remained an open question. In this work we describe an algorithm which performs each phase in $O(n^2)$ steps. Some other versions of this algorithm has the complexities $O(m \lg n)$, $O(m \lg \lg n)$ and $O(m+n^{1+\epsilon})$ (where ϵ is an arbitrary positive value).

Difficulties, methods and techniques. The augmenting paths which have to be found in each phase must be disjoint, minimum and legal. The fact that these three requirements must be fulfilled simultaneously causes some problems and difficulties: Generally, in order to

construct disjoint paths in a graph we use the Depth First Search (DFS) method. On the other hand, in order to find minimum paths, we perform a Breadth First Search (BFS) on the graph. However, the most problematic requirement is the legality, i.e., the requirement that the augmenting paths will not contain loops. Indeed, the requirement of minimality excludes the possibility that the augmenting path contains an even loop. For, if an augmenting path p contains an even loop (see Fig. 0.1(a)), then by removing this loop from the path we receive a shorter augmenting path p' (Fig. 0.1(b)). Yet, it is definitely possible that a minimum augmenting path contains an odd loop (see Fig. 0.1(c)).

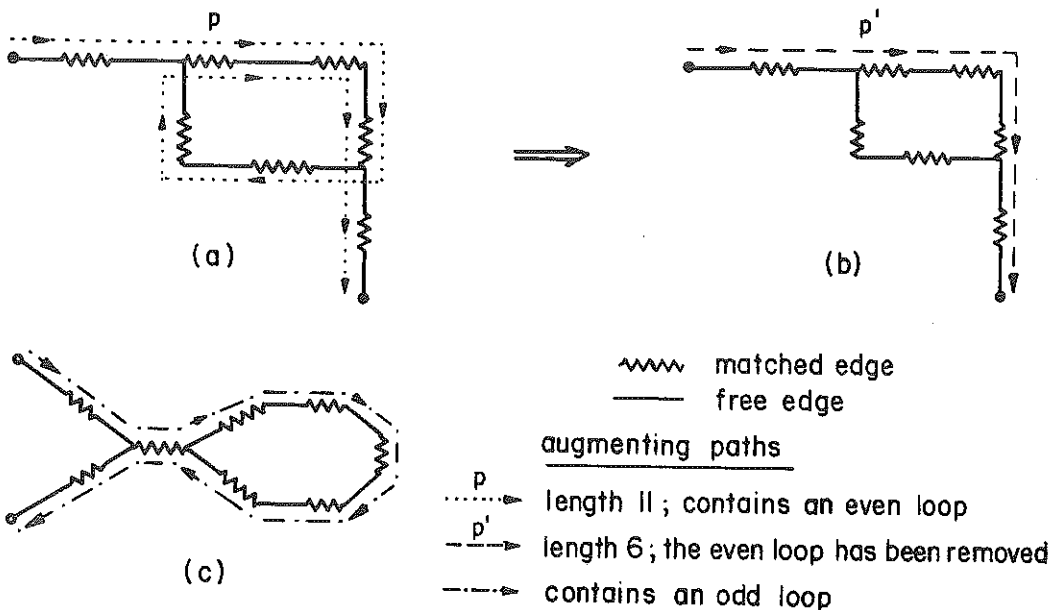


Fig. 0.1 - Loops

[Notice that the case of an odd loop is impossible in bipartite graphs. Therefore no problems of legality arise in the algorithms for bipartite graphs, and the requirement of legality may be omitted. And indeed, each phase in Hopcroft and Karp's algorithm is carried out by performing on the bipartite graph at first a Breadth First Search in order to find all the minimum augmenting paths which exist in the graph, and then a Depth First Search in order to choose out of those paths a maximal set of disjoint paths].

Edmonds [1] overcomes the difficulty of the odd loops by "shrinking" all the loops which are discovered during the performance of his algorithm. Witzgall and Zahn [2], as well as Gabow [3], store these loops in a special data structures. Our algorithm uses in fact all methods and techniques which are used by the other authors: We search the graph according to the BFS method, in order to find minimum alternating paths in the graph. These paths are recorded and stored in data structures which are generalizations of the data structures of Gabow. Some of the odd loops which are found during the search of the graph are shrunk in a process which resembles the blossoms' shrinking of Edmonds. We search the reduced graph (which is received by the shrinking process) according to a certain version of the DFS method, in order to find a maximal set of disjoint augmenting paths. However, all these techniques are used according to a very strict and well-defined system, which assures a maximal efficiency in each stage of the algorithm. Thus, the search of the graph is simultaneously performed from all the exposed vertices of the graph; the data structure which is used to record and store the alternating

paths, also serves to test their legality and to find alternative paths in case the present paths are found to be illegal; the shrinking process is not performed on every odd loop which is found in the graph, but only on those loops which disturb the construction of minimum legal augmenting paths. These methods enable to complete the performance of an entire phase of the algorithm in no more than $O(n^2)$ steps.

C. Description of a Phase

The representation of exposed vertices (notations). In order to have throughout the algorithm a uniform treatment of both the exposed and the matched vertices, it is convenient to replace every exposed vertex A by a matched edge $A_1 A_2$, such that the vertex A_1 meets all the (free) edges which are incident to A , while A_2 meets no free edges. We shall refer to the vertex A_1 [A_2] as a 1-exposed vertex [2-exposed vertex]. By this notation, all the vertices of the graph are matched*. Using this notation, the following definitions have to be reformulated:

An alternating path leading to a vertex R - an alternating path which goes from a 1-exposed vertex to the vertex R , such that its

* Clearly, this notation does not change the order of magnitude of the graph. Thus, in the following, we assume that the number of vertices n includes the 2-exposed vertices, and the number of edges m includes the matched edges $A_1 A_2$.

first edge (which is incident to the 1-exposed vertex) is free, and its last edge (which meets R) is matched.

An augmenting path. An alternating path leading to a 2-exposed vertex (namely, an alternating path which goes from a 1-exposed vertex to a 2-exposed vertex, such that its first edge is free).

Throughout the algorithm, we also use the term of "level" which is defined as follows:

The (well-defined) level of a vertex R - The length of a minimum (legal) alternating path leading to R [See also Section A of Chapter 1]. Clearly, if R is a 2-exposed vertex, then the (well-defined) level of R is the length of a minimum legal augmenting path leading to R .

The initialization of the algorithm. Our algorithm is initialized by defining an arbitrary matching in the graph. [Usually this is the empty matching; namely, all the vertices of the input graph are considered to be 1-exposed, and all the matched edges are of the form $A_1 A_2$ where A_1 and A_2 are 1-exposed and 2-exposed vertices respectively]. For each vertex R of the graph we define a variable MATE(R) which points to the mate of R , according to the initial matching. All the 1-exposed vertices are contained in a list which is called LIST(0) (see Section A of Chapter 1). We also assume that for each vertex of the graph we have the list of all the edges which are incident to it [for a detailed description see Chapter 5].

The structure of a phase. The algorithm which is presented in this dissertation works in phases, according to the results of Hopcroft and Karp [5]. In each phase, we try to find in the graph a maximal set of disjoint minimum legal augmenting paths. If no such augmenting path is found, the present matching is maximum and the algorithm is terminated. Else, an augmentation of the present matching is carried out by converting each free edge on these paths into a matched edge and vice versa. The 2-exposed vertices at the ends of these paths are removed from the graph, and their MATEs are deleted from LIST(0). By the result of Hopcroft and Karp, this process can be repeated at most \sqrt{n} times before maximum matching is reached.

Each phase consists of four stages:

In the first stage the graph is searched according to the BFS method, where the search is simultaneously begun from all the 1-exposed vertices of the graph. For each vertex of the graph we try to find a minimum legal alternating path leading to it, and thus to determine its well-defined level. These alternating paths are recorded and stored in special data structures which enable to verify the legality of the path and to build alternative alternating paths if the present paths are found to be illegal. These data structures also enable a simple restoration of the alternating paths. If a minimum legal alternating path which is leading to a 2-exposed vertex A_2 is found in the graph, then (by the definition) this path is a minimum legal augmenting path, and the level r of the 2-exposed vertex A_2 is the length of the minimum legal augmenting paths which

exist in the graph. In this case the first stage is terminated and we proceed to the second stage. If no legal alternating path which is leading to a 2-exposed vertex is found throughout the search of the graph, then the present matching is maximum and the whole algorithm is terminated. The complexity of the first stage is $O(n^2)$. Other implementations of the data structures lead to complexities of $O(m \lg n)$, $O(m \lg \lg n)$ or $O(m + n^{1+\epsilon})$.

In the second stage we use the levels which were defined for the vertices in the first stage, in order to remove from the graph all the vertices and edges which are not lying on minimum legal augmenting paths (of length r), and to shrink into their bases all the odd loops which are lying on illegal augmenting paths of length less than r . The resulting graph, which is called the reduced graph, has the property that any arbitrary legal augmenting path of this graph, corresponds to a minimum legal augmenting path in the original graph. Moreover, each maximal set of disjoint arbitrary legal augmenting path in the reduced graph corresponds to a maximal set of disjoint minimum legal augmenting paths of the original graph. The complexity of the second stage is $O(m)$.

The properties of the reduced graph are utilized in the third stage where a maximal set of disjoint arbitrary legal augmenting paths is found in the reduced graph. The renouncement of the minimality makes it possible to accomplish this task by performing a certain version of the Depth First Search method on the graph, where no edge of the reduced graph is searched more than once in each direction.

The complexity of this stage is $O(n_r^2)$ or $O(m_r + n_r \lg \lg n_r)$ (depending on the exact implementation of the data structures), where n_r and m_r are the numbers of vertices and edges of the reduced graph respectively.

In the fourth stage, the augmenting paths which have been found in the reduced graph during the third stage are retraced. By restoring the paths of the original graph which correspond to these paths of the reduced graph, we get a maximal set of disjoint minimum legal augmenting paths in the original graph. A process of augmentation is performed on these paths, and the matching is improved. The 2-exposed vertices at the ends of these paths are removed from the graph, and their MATEs are deleted from the list LIST(0). The performance of the present phase is terminated, and we return to the first stage, to start a new one. The complexity of the fourth stage is $O(n)$.

D. The Algorithm

According to the above discussion, a summary description of the algorithm will be the following:

0. Initialization: Input the graph. Define an arbitrary matching. Replace every exposed vertex by a matched edge $A_1 A_2$ where A_1 is a 1-exposed vertex and A_2 is 2-exposed. Insert all the 1-exposed vertices into LIST(0). For each vertex R assign the appropriate value of MATE(R) according to the initial matching (in particular: $\text{MATE}(A_1) \leftarrow A_2$, $\text{MATE}(A_2) \leftarrow A_1$). [see Chapter 5].

1. Perform the First Stage [see Chapter 1]. If the algorithm is terminated go to 5. Else go to 2.
2. Perform the Second Stage [see Chapter 2].
3. Perform the Third Stage [see Chapter 3].
4. Perform the Fourth Stage [see Chapter 4]. Return to 1.
5. Print the variables MATEs (which give the maximum matching) and the list LIST(0) (which gives the list of the exposed vertices).

A PL1 program which is based on this algorithm is described in Appendix C.

1. THE FIRST STAGE

The goal of the first stage is to find for each vertex R a shortest legal alternating path leading to it from a 1-exposed vertex. The length of this minimum path is defined as the level of the vertex R . Therefore, the level of a 2-exposed vertex is the length of a legal minimum augmenting path leading to it. Thus, by finding at the end of the first stage a 2-exposed vertex of minimum level r , we also know the length of the minimum legal augmenting paths in the graph, and the vertices which are lying on these paths.

A. Notations

The basic operation of the first stage is the search through the free edges: During the performance of the first stage, each free edge of the graph is searched at most once in each direction. The order by which these edges are searched is determined by the Breadth First Search (BFS) method, which will be explained later (Section B). By using the edges which have already been searched, we try to construct certain alternating paths in the graph.

Levels and alternating paths. Let $p(0 \rightarrow R)$ denote the shortest alternating path leading to R which has been constructed until now and has not yet been found to be illegal, and let $l(R)$ denote its length. If we can prove that $p(0 \rightarrow R)$ is both minimum and legal, then we refer to $p(0 \rightarrow R)$ as the well-defined al-path leading to R , and

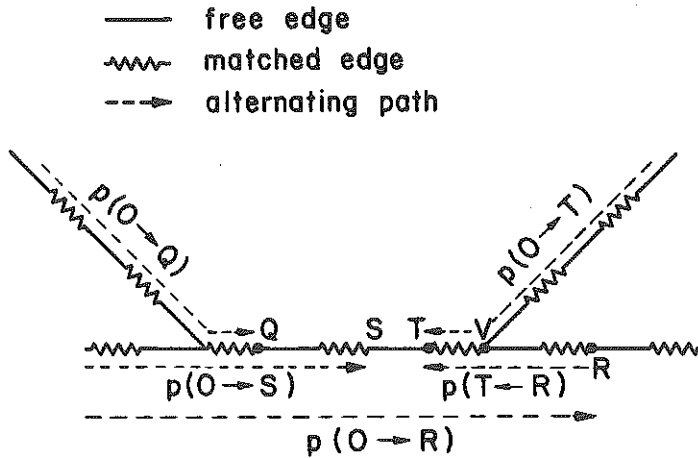
we refer to $\ell(R)$ as the well-defined level of R . In this case R is called a well-defined vertex. If the minimality or the legality of $p(0 \rightarrow R)$ have not been verified yet, we refer to $p(0 \rightarrow R)$ and to $\ell(R)$ as the potential a1-path and the potential level of R respectively.

Let T be a vertex lying on $p(0 \rightarrow R)$. By $p(T \rightarrow R)$ we denote the segment from T to R on the a1-path $p(0 \rightarrow R)$. While referring to this segment in the reverse direction (from R to T) we use the notation $p(T \leftarrow R)$. (Notice that $p(T \leftarrow R)$ is not necessarily the same as $p(R \rightarrow T)$. For example, if T is lying on $p(0 \rightarrow R)$ while R is not lying on $p(0 \rightarrow T)$, then $p(T \leftarrow R)$ is defined while $p(R \rightarrow T)$ does not exist at all. See Fig. 1.1).

Let R be a vertex and let ST be a free edge which is lying on $p(0 \rightarrow R)$ such that $p(0 \rightarrow R) = p(0 \rightarrow S) \cdot p(T \rightarrow R)$. Then $p(0 \rightarrow S)$ is called a head of $p(0 \rightarrow R)$ (see Fig. 1.1).

Let TV be a matched edge which is lying on $p(0 \rightarrow R)$ such that on this path V is closer than T to R . We say that V is f-lying on $p(0 \rightarrow R)$ (see Fig. 1.1).

In the beginning of the first stage all the 1-exposed vertices have a (well-defined) level 0. It is convenient to assume that all the other vertices have an initial dummy (potential) level $\lfloor \frac{n}{2} \rfloor + 1$ (since no alternating path of length greater than $\lfloor \frac{n}{2} \rfloor$ can exist in the graph, this notation causes no confusion). A LIST(i) which contains all the vertices whose level is i , is defined for each of



- (0) V is f-lying on $p(O \rightarrow R)$.
- (1) S is f-lying on $p(O \rightarrow R)$ and $p(O \rightarrow S)$ is a head of $p(O \rightarrow R)$.
- (2) Q is f-lying on $p(O \rightarrow R)$ but $p(O \rightarrow Q)$ is not a head of $p(O \rightarrow R)$.
- (3) T is lying but not f-lying on $p(O \rightarrow R)$ [$p(T \rightarrow R)$ exists].
- (4) R is not lying on $p(O \rightarrow T)$ [$p(R \rightarrow T)$ does not exist].

Fig. 1.1 - Notations

the possible values of the levels: $i=0,1,\dots, \lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor + 1^*$. Thus, in the beginning of the first stage $LIST(0)$ contains all the 1-exposed vertices, $LIST(\lfloor \frac{n}{2} \rfloor + 1)$ contains all the other vertices, and all the other $LISTs$ are empty.

Let ST be an edge. We define the sum $\ell(S) + \ell(T)$ as the level of the edge ST and we denote it by $\ell(ST)$.

* $LIST(0)$ has already been defined during the initialization of the algorithm - see Sections C and D of the Introduction.

B. The BFS Method

The first stage of the algorithm is performed in substages. The goal of the j -th substage is to find all vertices of well-defined level j . Each substage consists of two parts: SEARCH and UPDATE.

During the SEARCH part we search exactly through all the free edges XY which are incident at the vertices X whose (well-defined) level is $j-1$. These edges are searched in the direction from X to Y^* , and attempts are made to construct new alternating paths in the graph by concatenating these edges to the al-paths $p(0 \rightarrow X)$.

Assume that by this process a new alternating path leading to the vertex R has been found: We first test the legality of this path in a way which will be explained later (Section D). If in this test the alternating path is not found to be illegal, then it is recorded in a special data structure which is related to the vertex R (notice that this alternating path may still be found illegal in later tests). Then we check whether this alternating path can define for R a lower level than the present $\ell(R)$, and if the answer is positive we assign this path as the new al-path $p(0 \rightarrow R)$ which is leading to R , and we define its length as the new $\ell(R)$. In particular, if in the j -th substage $\ell(R)=j$, then (as we shall see, Corollary 1.2) the al-path $p(0 \rightarrow R)$ and the level $\ell(R)$ are in

* By saying "XY is searched" we shall always mean "XY is searched in the direction from X to Y ".

fact well-defined (i.e., no more tests are needed to verify their legality, and since we are in the j -th substage, there is also no doubt about their minimality).

Thus, in the beginning of the UPDATE part we declare all vertices whose level is j , to be well-defined. These declarations cause changes in some of the data structures of the graph, and these data structures must therefore be updated. The updating is done by transferring the information which is related to the vertices which are declared to be well-defined to the data structures of some other vertices whose levels are still potential. Assume that during the updating process an information is transferred to the data structure related to some vertex R_2 . This additional information enables us to try to lower the level of R_2 . This information also enables us to test the legality of the present al -path $p(O \rightarrow R_2)$. Moreover, in case $p(O \rightarrow R_2)$ fails this test, we also use the information related to R_2 in order to define for R_2 the least level and a shortest al -path $p(O \rightarrow R_2)$ which presently pass the legality test. The details of all these operations will be described after the data structures are defined and explained (Sections D and E).

The first stage is terminated in one of the following two ways:

(i) If during the r -th substage we find at the first time that some 2-exposed vertices have (well-defined) level r , then legal augmenting paths of (minimum) length r have been found in the

graph. We shall prove (Corollary 1.4) that all vertices whose present levels are greater than r (and therefore are still potential), can not be lying on minimum legal augmenting paths and thus are not relevant for the present phase. Therefore we do not proceed to the next substages to find the well-defined levels of these vertices, but we terminate the performance of the first stage and proceed to the second stage.

(ii) If the $\lfloor \frac{n}{2} \rfloor$ -th substage is terminated and no 2-exposed vertex has been declared to be well-defined, then no legal augmenting path exists in the graph. The present matching is maximum and the whole algorithm is terminated.

C. The Data Structures

Assume that during the first stage we construct an alternating path p_1 of length ℓ_1 leading to vertex R . If $\ell_1 < \ell(R)$ (where $\ell(R)$ is the present level of R), then the new path p_1 replaces the present al-path $p(O \rightarrow R)$ and the level of R is updated; i.e., we assign: $p(O \rightarrow R) \leftarrow p_1$; $\ell(R) \leftarrow \ell_1$. However, we want to keep the information about the old path $p(O \rightarrow R)$ which may still be useful for the construction of other alternating paths. If $\ell_1 > \ell(R)$, then although the present level and al-path of R are not changed, we still want to keep the information about the new alternating path p_1 , because this information may also be used later to construct other alternating paths. Thus, a data structure which enables an

efficient treatment of alternating paths in the graph is required.

For these purposes (efficient recording and tracing of al-paths) we attach to each vertex a variable called LINK and we define the concept of a "bridge" (to be explained). For the purpose of storing and restoring these LINKs we attach to each vertex a list called CHAIN. The list of vertices whose CHAINS contain information about alternating paths leading to the vertex R is accumulated in a set called T(R). The data structure T(R) is also used for the purpose of testing and verifying the legality of the al-paths. (The minimality of these paths is assured by the BFS method). We now describe these data structures.

LINK. In order to efficiently record and retrace the al-paths, we use a data structure suggested by Gabow [3]: To each vertex Z of the graph, a variable called "LINK" is attached. If $\ell(Z) = \lfloor \frac{n}{2} \rfloor + 1$ (i.e., if every al-path leading to Z, considered so far, has been found to be illegal), then $LINK(Z) = 0$. Else, $LINK(Z)$ recursively describes the present al-path $p(0 \rightarrow Z)$. A LINK which defines a well-defined (potential) al-path is called a well-defined (potential) LINK. There are two types of LINKs: 1-link and 2-link. The 1-link points to another vertex ($LINK(Z) = X$), while the 2-link points to a free edge ($LINK(Z) = PQ$).

1-link. A 1-link of a vertex Z is a link which points to another vertex X such that the following conditions are satisfied:

- (i) The free edge XY (where $Y = MATE(Z)$) has already been

searched (thus, by the BFS method X is already well-defined).

(ii) When the assignment $\text{LINK}(Z) \leftarrow X$ is performed, both Y and Z have the initial dummy level $\lfloor \frac{n}{2} \rfloor + 1$ (thus, $\text{LINK}(Y) = \text{LINK}(Z) = 0$).

The a_1 -path $p(0 \rightarrow Z)$ which is defined by $\text{LINK}(Z) = X$ is described by the following concatenation:

$$p(0 \rightarrow Z) = p(0 \rightarrow X) \cdot Y \cdot Z . \quad (1.1)$$

Clearly (1.1) defines an alternating path leading to Z (see Fig. 1.2).

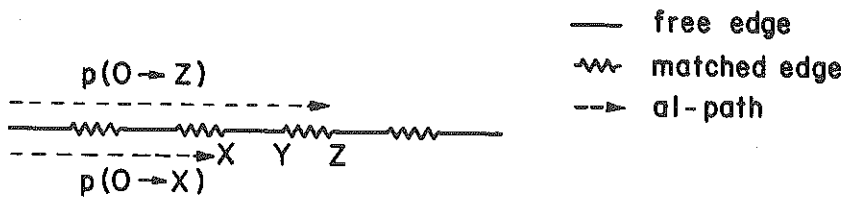


Fig. 1.2 - 1-link

Let YZ be a matched edge such that both Y and Z have the initial level $\lfloor \frac{n}{2} \rfloor + 1$. In the algorithm we perform the 1-link assignment immediately when it becomes possible, i.e., when the first free edge XY is searched. If XY is searched during the j -th substage, then (by the BFS method) X must have a (well-defined) level $j-1$

and thus the length of the a_1 -path as defined by (1.1) is j . Therefore we assign: $LINK(Z) \leftarrow X$, $l(Z) \leftarrow j$. We shall prove (Theorem 1.1) that this assignment defines a well-defined level and $LINK$ for Z . Thus:

(a) [By the algorithm:] The first of Y and Z to get a $LINK$ other than 0 is given a 1-link.

(b) [By Theorem 1.1:] A 1-link is always well-defined.

(c) [By condition (ii) of the definition:] If Z has a 1-link then Y cannot have a 1-link too. Thus, either $LINK(Y)=0$ (and $l(Y) = \lfloor \frac{n}{2} \rfloor + 1$), or Y has a 2-link.

It is convenient to assume that all the 1-exposed vertices have a 1-link to a dummy vertex DUM (this notation causes no confusion).

Bridge. Let ST be a free edge. If $LINK(MATE(T))=S$ then ST is lying on the a_1 -path $p(0 \rightarrow MATE(T))$. If $LINK(MATE(S))=T$ then ST (or rather TS) is lying on the a_1 -path $p(0 \rightarrow MATE(S))$. Hence, if the edge ST has already been searched^{*} and neither $LINK(MATE(T))=S$ nor $LINK(MATE(S))=T$ then ST is neither lying on $p(0 \rightarrow MATE(T))$ nor on $p(0 \rightarrow MATE(S))$ but it connects and bridges these two a_1 -paths. In this case we call ST a bridge (see Fig. 1.3).

* If this condition is omitted, then each free edge would be defined as a bridge in the beginning of the first stage, and the concept of a bridge would not necessarily be permanent.

- > an α 1-path leading to a vertex which has a 1-link
- free edge
- == bridge
- ~ matched edge

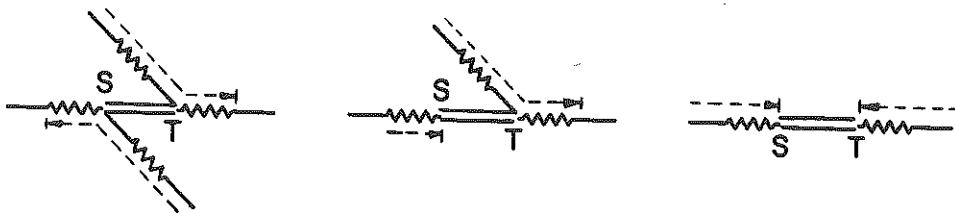


Fig. 1.3 - Bridges

If ST is a bridge, then by the definition it has been searched from S to T , and thus, according to the BFS method, S must already be well-defined. If T is also well-defined, then we refer to ST as a well-defined bridge. If the level of T is still potential, we call ST a potential bridge.

It will be shown (Lemma 1.1) that once ST is searched and found to be a bridge it will remain a bridge throughout the remainder of the first stage (although a potential bridge may of course become well-defined). Thus, when ST is searched, we can decide whether it is a bridge or not. Clearly, if ST is a bridge, then, if and when TS is searched, it becomes a (well-defined) bridge too.

2-link. A 2-link of a vertex R is a link which points to a bridge ST where the following conditions hold:

- (i) ST is a well-defined bridge.
- (ii) $MATE(R)$ has a 1-link.
- (iii) $p(O \rightarrow MATE(R))$ is a head of $p(O \rightarrow T)$.
- (iv) All the vertices on $p(MATE(R) \rightarrow T)$ are well-defined.

In this case, the a_1 -path $p(O \rightarrow R)$ which is defined by $LINK(R) = ST$ is described by the following concatenation:

$$p(O \rightarrow R) = p(O \rightarrow S) \cdot p(R \leftarrow T) . \quad (1.2)$$

Clearly, (1.2) defines an alternating path leading to R (see Fig. 1.4).

- $---\rightarrow$ alternating path
- $---$ free edge
- $===$ bridge
- $\sim\sim\sim$ matched edge

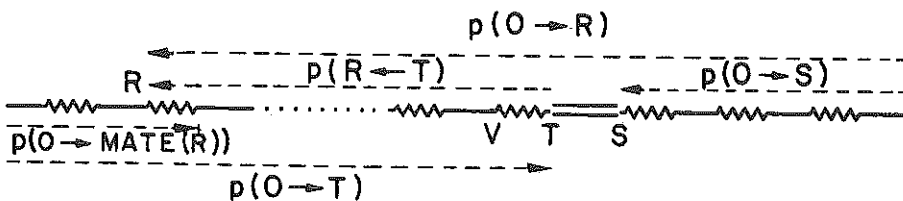


Fig. 1.4 - 2-link

If $p(0 \rightarrow \text{MATE}(R))$ is a head of $p(0 \rightarrow T)$, then the number of free edges on $p(\text{MATE}(R) \rightarrow T)$ is: $\lambda(T) - \lambda(\text{MATE}(R))$. Using this number, it is easy to see that if $\text{LINK}(R) = ST$ then:

$$\lambda(R) = \lambda(ST) + 1 - \lambda(\text{MATE}(R)). \quad (1.3)$$

Thus, in order to define a minimal level for R , a bridge ST of minimum level must be found (notice that $\text{MATE}(R)$ has a 1-link and therefore $\lambda(\text{MATE}(R))$ is already well-defined and cannot be changed).

Notice that in opposite to the case of a 1-link, generally neither the minimality nor the legality of an a1-path $p(0 \rightarrow R)$ which is defined through a 2-link are verified, unless a further check is carried out. For example, if R is lying on $p(0 \rightarrow S)$, then the a1-path $p(0 \rightarrow R)$ as defined by (1.2) contains an odd loop and is therefore illegal (see Fig. 1.5).

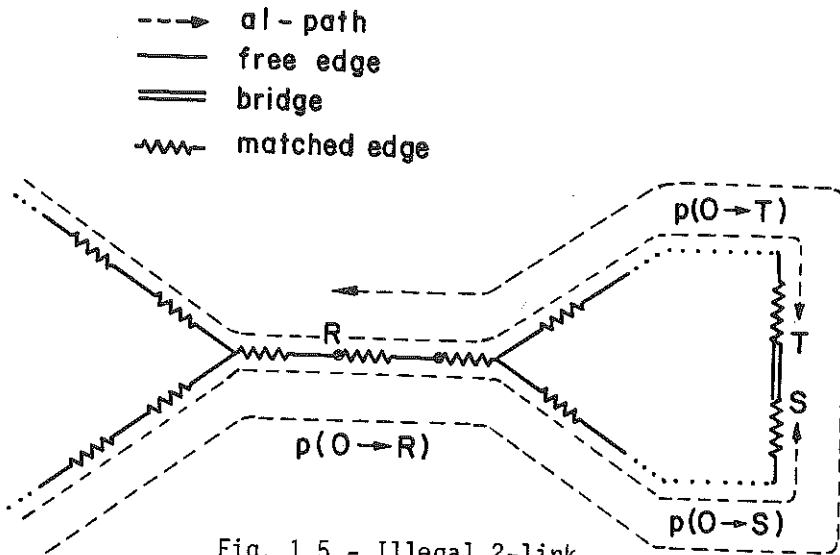


Fig. 1.5 - Illegal 2-link

The minimality is not verified because $\ell(R)$ depends on $\ell(ST)$ and according to the BFS method it may happen that although $\ell(S_1 T_1) < \ell(S_2 T_2)$, yet $\ell(S_2) < \ell(S_1)$ and therefore the bridge $S_2 T_2$ is searched and assigned as $LINK(R)$ before the bridge $S_1 T_1$ is searched. Clearly, the assignment $LINK(R) \leftarrow S_2 T_2$ does not define a minimum level for R .

Thus, an al-path which is defined by a 2-link is always considered a potential al-path until its minimality and legality are verified. The 2-link considered in these circumstances is a potential LINK.

The legality of an al-path $p(O \rightarrow R)$ which is defined by a 2-link is verified by a "legality test", which is performed on $LINK(R)$ (this test is described in Section D). If $LINK(R)$ fails the legality test (i.e., if $p(O \rightarrow R)$ is found to be illegal), then the bridge which is presently assigned as $LINK(R)$, must be replaced by another bridge which can be assigned as $LINK(R)$ and has not failed the legality test so far. The minimality of $p(O \rightarrow R)$ is verified by using the following procedure: Each time a new assignment $LINK(R) \leftarrow ST$ becomes possible according to the definition, we check whether this assignment defines for R a level which is lower than the present $\ell(R)$, and if the answer is positive we perform these changes on $LINK(R)$ and $\ell(R)$. Also, each time the present $LINK(R)$ is found to be illegal, we choose a bridge of minimal level among the bridges which can be assigned as $LINK(R)$ and presently pass the legality test, and we assign this bridge as the new $LINK(R)$.

Therefore a list of all the bridges which can presently be assigned as $LINK(R)$ is required. For this purpose we use two data structures: The bridges of the graph are stored in disjoint lists called "CHAINS", and the list of all CHAINS whose bridges can be assigned as $LINK(R)$ is accumulated in a set called $T(R)$.

CHAIN. For each vertex in the graph a separate list called CHAIN is defined: Let T be a vertex, then $CHAIN(T)$ is an ordered list of bridges of the form $S_i T$ which is constructed as follows:

In the beginning of the first stage all the CHAINS are empty. Each time a free edge ST is searched and found to be a bridge (either potential or well-defined), it is attached to the end of the list $CHAIN(T)^*$. Thus the bridges of $CHAIN(T)$ are arranged in the same order by which they have been searched. Therefore a bridge $S_i T$ precedes another bridge $S_j T$ in $CHAIN(T)$ only if $S_i T$ has been searched before $S_j T$. By the BFS method this implies that $\ell(S_i) \leq \ell(S_j)$, or $\ell(S_i T) \leq \ell(S_j T)$. Thus, the bridges of $CHAIN(T)$ are arranged in nondecreasing order of their levels, and in particular, the first bridge has always a minimum level in $CHAIN(T)$.

Each time an assignment $LINK(R)=ST$ fails the legality test, ST is deleted from $CHAIN(T)$. It will be proven (Theorem 1.4) that a bridge which has been failed the legality test cannot define a legal level for any of the vertices whose levels are still potential.

* In fact, if ST is a well-defined bridge, it is inserted into $CHAIN(T)$ only if it presently passes the legality test.

Thus, the deletion of ST from $CHAIN(T)$ in this case is safe.

Since each bridge S_iT of $CHAIN(T)$ has already been searched, S_i must be well-defined. Therefore S_iT is a well-defined bridge if and only if T has a well-defined level. It follows that either all the bridges of $CHAIN(T)$ are well-defined, or none.

Therefore, by the definition of the 2-link, we see that if one bridge S_iT of $CHAIN(T)$ can be assigned as $LINK(R)$, then all the bridges of $CHAIN(T)$ can be assigned as $LINK(R)$. (Notice that Equation (1.3) and the construction of $CHAIN(T)$ imply that out of all the bridges of $CHAIN(T)$, the first bridge defines for R a minimal level).

The list of all the CHAINS whose bridges can be assigned as $LINK(R)$ is accumulated in a data structure called $T(R)$.

$T(R)$. Let R be a vertex such that $MATE(R)$ has a 1-link (i.e., if $LINK(R) \neq 0$ then R has a 2-link). Assume that $LINK(R)$ is still not well-defined; then we want to have a set $T(R)$ which includes exactly all the CHAINS whose bridges can be assigned as $LINK(R)$.

For this purpose we attach to each vertex T in the graph a variable called $TAIL(T)$ which is defined as follows*:

* $TAIL(T)$ in our algorithm is a generalization of $TOP(LINK(T))$ in Gabow's algorithm [3]. In fact, this variable is not essential for the performance of the algorithm, and it is introduced only to

(a) If T is not yet well-defined then $TAIL(T)=MATE(T)$.

(b) If T is well-defined, then $TAIL(T)$ is the vertex closest to T on $p(0 \rightarrow T)$ whose $LINK$ is still potential.

Using the variable $TAIL$ we now attach to each vertex R a set $T(R)$ of vertices, which is defined as follows:

$T(R)$ is the set of all vertices T such that $TAIL(T)=R$.

Clearly all the sets $T(R)$ are disjoint and their union contains all vertices of the graph.

We now prove that if $MATE(R)$ has a 1-link and R is not well-defined, then $T(R)$ is exactly the set of all the vertices, such that the bridges of their $CHAINS$ can presently be assigned as $LINK(R)$.

First we prove the following lemma:

Lemma 1.0. If Z is f-lying on $p(0 \rightarrow T)$ (where $T \neq Z$) , then Z is well-defined.

[Thus, if T is well-defined, $TAIL(T)$ is lying but not f-lying on $p(0 \rightarrow T)$].

Proof of the lemma. Consider the sequence of vertices T_1, T_2, \dots, T_k which is defined as follows:

enable a better understanding of the sets $T(R)$.

$$T_0 = T \quad T_i = \begin{cases} W & \text{if LINK}(T_{i-1}) = W \\ P & \text{if LINK}(T_{i-1}) = PQ \end{cases} \quad \text{LINK}(T_k) = \text{DUM} \\ 1 \leq i \leq k$$

By the definition of 1-link and 2-link, for each i , $1 \leq i \leq k$, T_i is well-defined, and $p(0 \rightarrow T_i)$ is a head of $p(0 \rightarrow T_{i-1})$. Therefore, for each i , $1 \leq i \leq k$, $p(0 \rightarrow T_i)$ is a head of $p(0 \rightarrow T)$.

Let Z be a vertex which is f -lying on $p(0 \rightarrow T)$, and let j be the greatest index such that $T_j \neq Z$, and on $p(0 \rightarrow T)$, T_j is lying between Z and T . Clearly $0 \leq j \leq k$ and Z is f -lying on $p(0 \rightarrow T_j)$. Assume that T_j has a 1-link, then $T_{j+1} = \text{LINK}(T_j)$. If $T_{j+1} \neq Z$, then T_{j+1} contradicts the choice of T_j . Thus, $T_{j+1} = Z$, and therefore Z is well-defined. Assume that T_j has a 2-link, $\text{LINK}(T_j) = PQ$, then $T_{j+1} = P$. If $Z = T_{j+1}$ then Z is well-defined; else, by the choice of T_j , Z is lying on $p(\text{MATE}(T_{j+1}) \rightarrow T_j)$, and thus, by the definition of a 2-link, Z is well-defined.

Q.E.D.

Theorem 1.0

(a) If $\text{MATE}(R)$ is not well-defined, then $T(R) = \{\text{MATE}(R)\}$.

(b) If $\text{MATE}(R)$ is well-defined but R is not well-defined yet (thus $\text{MATE}(R)$ has a 1-link), then $T(R)$ if and only if $p(0 \rightarrow \text{MATE}(R))$ is a head of $p(0 \rightarrow T)$ and all the vertices on $p(\text{MATE}(R) \rightarrow T)$ are well-defined.

(c) If both $\text{MATE}(R)$ and R are well-defined, then $T(R) = \emptyset$.

Proof.

(a) If $MATE(R)$ is not well-defined, then by the definition $TAIL(MATE(R))=R$ and thus $MATE(R) \in T(R)$. Assume that $T(R)$ contains a vertex V , where $V \neq MATE(R)$. Then $TAIL(V)=R$, and by the definition R is the vertex closest to V on $p(O \rightarrow V)$ whose $LINK$ is still potential. Since $V \neq R$ (for, by the definition $TAIL(V) \neq V$), then by Lemma 1.0, R cannot be f -lying on $p(O \rightarrow V)$ and thus $MATE(R)$ is f -lying on $p(O \rightarrow V)$. But since $MATE(R)$ is not well-defined and on $p(O \rightarrow V)$ it is closer than R to V , the assumption $R=TAIL(V)$ is contradicted. Thus, $T(R) = \{MATE(R)\}$.

(b) (i) [If]. Assume that $p(O \rightarrow MATE(R))$ is a head of $p(O \rightarrow T)$ and all the vertices on $p(MATE(R) \rightarrow T)$ are well-defined. Since R is not well-defined, it is the vertex closest to T on $p(O \rightarrow T)$ whose $LINK$ is still potential. Thus, $TAIL(T)=R$ and $T \in T(R)$.

(ii) [Only if]. Assume that $T \in T(R)$, thus R is the vertex closest to T on $p(O \rightarrow T)$ whose $LINK$ is still potential. Since $T \neq R$ (for, by the definition $TAIL(T) \neq T$), then by Lemma 1.0 R is not f -lying on $p(O \rightarrow T)$. Thus, $MATE(R)$ is f -lying on $p(O \rightarrow T)$, and therefore, by the choice of R , all the vertices on $p(MATE(R) \rightarrow T)$ are well-defined. We still have to prove that $p(O \rightarrow MATE(R))$ is a head of $p(O \rightarrow T)$. This claim is trivial if $T=MATE(R)$. Assume that $T \neq MATE(R)$ and let W be the vertex closest to R on $p(MATE(R) \rightarrow T)$ such that $p(O \rightarrow W)$ is a head of $p(O \rightarrow T)$. If $W \neq MATE(R)$, then W must have a 2-link (otherwise $LINK(W)$ would contradict the choice of W). Let $LINK(W)=PQ$, then $p(O \rightarrow P)$ is a head of $p(O \rightarrow W)$ and

thus a head of $p(0 \rightarrow T)$. Hence, by the choice of W , R cannot be lying on $p(0 \rightarrow P)$, but must be lying on $p(W \rightarrow Q)$. But in this case, according to the definition of 2-link, R must be well-defined - and this is a contradiction. Thus, $W = \text{MATE}(R)$, and $p(0 \rightarrow \text{MATE}(R))$ is therefore a head of $p(0 \rightarrow T)$.

(c) If both $\text{MATE}(R)$ and R are well-defined, then for no vertex T , $\text{TAIL}(T) = R$. Thus $T(R) = \emptyset$.

Q.E.D.

Corollary 1.0

If $\text{MATE}(R)$ has a 1-link and R is not well-defined yet, then $T(R)$ is exactly the set of all the vertices such that the bridges of their CHAINS can presently be assigned as $\text{LINK}(R)$.

Proof. (i) If ST is a bridge in $\text{CHAIN}(T)$ which can be assigned as $\text{LINK}(R)$, then by the definition of 2-link, $p(0 \rightarrow \text{MATE}(R))$ is a head of $p(0 \rightarrow T)$ and all the vertices on $p(\text{MATE}(R) \rightarrow T)$ are well-defined. Thus, by case (b) of Theorem 1.0 $T \in T(R)$.

(ii) If $T \in T(R)$ then by case (b) of Theorem 1.0 $p(0 \rightarrow \text{MATE}(R))$ is a head of $p(0 \rightarrow T)$ and all the vertices on $p(\text{MATE}(R) \rightarrow T)$ are well-defined. In particular, T is well-defined, and thus each bridge ST in $\text{CHAIN}(T)$ is a well-defined bridge. Therefore, by the 2-link definition, ST can be assigned as $\text{LINK}(R)$.

Q.E.D.

Therefore, if $\text{MATE}(R)$ has a 1-link and R is still not well-defined, then the set $\{ST \mid ST \in \text{CHAIN}(T), T \in T(R)\}$ is exactly the set

of all bridges which can presently be assigned as $LINK(R)$, and have not yet failed the legality test. Thus, in order to find a minimal 2-link for R , we have to find a bridge of minimum level in this set. [In fact, by the construction of the CHAINS, it is sufficient to find a bridge of minimum level in the subset $\{ST | ST \text{ is first in } CHAIN(T), T \in T(R)\}$.]

According to the definition, changes of the variables TAILS and the sets $T(R)$ occur if and only if a vertex R_1 which has a 2-link is declared to be well-defined. [By Theorem 1.3, in this case all the vertices of $T(R_1)$ are transferred to another set $T(R_2)$, and the set $T(R_1)$ becomes empty; also, the set $T(MATE(R_1))$, which includes only the vertex R_1 , becomes empty, and R_1 is inserted into another set $T(R_3)$.]. Thus, at the end of the SEARCH part of the j -th substage, when the vertices of level j are declared to be well-defined, the appropriate sets $T(R)$ must be updated. The updating process, which is performed during the UPDATE part of the j -th substage, is described in Section E.

D. Legality and Minimality

Since a 1-link is always well-defined (Theorem 1.1) we discuss here only the case of a 2-link.

Legality. In Section G we prove the following theorem (Theorem 1.4):

- (a) Let $LINK(R)=ST$. If $SET(R)$, then the 2-link $LINK(R)=ST$

is illegal, and the bridge ST cannot be used later to define a legal 2-link for any other vertex whose $LINK$ is still potential.

(b) Let R be a vertex such that at the end of the $SEARCH$ part of the j -th substage R is not yet declared to be well-defined. Let ST be a bridge which can be assigned as $LINK(R)$ such that this assignment defines for R a level $\leq j$. If $LINK(R)=ST$ is illegal, then $SET(R)$.

Since the relation $SET(R)$ means that the bridges of $CHAIN(S)$ [and in particular, the bridge TS] can be assigned as $LINK(R)$, we can roughly reformulate Theorem 1.4 as follows: "The assignment $LINK(R)=ST$ is illegal if and only if TS can also be assigned as $LINK(R)$ ". Clearly, if the assignment $LINK(R)=ST$ defines for R a level j , then (according to Equation (1.3)), the same level is defined for R also by the assignment $LINK(R)=TS$. It follows by the BFS method and by the definition of $T(R)$, that if $LINK(R)=ST$ is illegal, then during the $SEARCH$ part of the j -th substage both S and T belong to the set $T(R)$, [part (b) of the theorem].

Therefore we define the legality test of a bridge ST to be: "Do S and T belong to the same set $T(R)$?" If the answer to this question is positive, then (by part (a) of Theorem 1.4), ST cannot be used to define a legal $LINK$ for any of the other vertices whose $LINK$ is still potential, and therefore ST is no more useful and can safely be deleted from its $CHAIN$. Thus, each time a well-defined bridge is searched, it is inserted into the appropriate $CHAIN$

only if it passes the legality test^{*}. Also, each time the present LINK(R) fails the legality test and thus is found to be illegal, we assign some other bridge as the new LINK(R) only if it presently passes the legality test.

Apparently, according to part (b) of Theorem 1.4, if LINK(R)=ST defines for R a level j, it suffices to test the legality of ST at the end of the SEARCH part of the j-th substage (just before R is declared to be well-defined). However, due to complexity considerations which will be explained in Section H, it is more convenient to test the legality of ST just before the assignment LINK(R) \leftarrow ST is performed, and to repeat this test each time T(R) is changed. If LINK(R) fails one of these tests it is replaced by another bridge which presently passes the test and can be assigned as LINK(R) (if no such a bridge exists, we assign LINK(R) \leftarrow 0). This procedure assures us that the present LINK(R) passes the legality test. Therefore, if at the end of the SEARCH part of the j-th substage the present LINK(R) defines for R a level j, then (by part (b) of Theorem 1.4) LINK(R) is legal and no further tests of the legality are required.

* The legality test is useless for a potential bridge: If ST is a potential bridge, then T is not well-defined and thus TAIL(T)=MATE(T), or T(MATE(T))={T}. Since S \neq T, S cannot belong to the same set as T, and therefore a potential bridge always passes the legality test.

Minimality. In order to verify the minimality of $\lambda(R)$, we use a procedure which assures that the present $\text{LINK}(R)$ defines for R the minimum legal level which has become possible so far. By this procedure the minimality of the a_1 -path which is defined by $\text{LINK}(R)$ is checked each time one of the following two cases happens:

(i) A new assignment $\text{LINK}(R) \leftarrow ST$ becomes possible by the definition.

(ii) The present $\text{LINK}(R)$ is found to be illegal and another bridge is assigned as $\text{LINK}(R)$.

We now discuss these two cases:

(i) Since a 2-link can be assigned to R , $\text{MATE}(R)$ must have a 1-link and is therefore well-defined. According to Corollary 1.0 of Section C, the set $\{S'T' | S'T' \in \text{CHAIN}(T'), T' \in T(R)\}$ contains exactly all bridges which can presently be assigned as $\text{LINK}(R)$, and have not yet failed the legality test. Thus, a new assignment $\text{LINK}(R) \leftarrow ST$ becomes possible by the definition, if and only if the bridge ST is inserted into the set $\{S'T' | S'T' \in \text{CHAIN}(T'), T' \in T(R)\}$. This event can happen in one of the following two ways: Either ST is a new well-defined bridge which is searched, passes the legality test and is inserted into $\text{CHAIN}(T)$, or the set $T(R)$ is changed and the vertex T becomes a member of it. Thus, in order to verify the minimality of $\lambda(R)$, we have to keep the following two rules:

(a) Each time a well-defined bridge ST is inserted into $\text{CHAIN}(T)$, we check by Eq. (1.3) whether the assignment $\text{LINK}(R) \leftarrow ST$ can lower the present $\lambda(R)$, and we perform these changes on $\text{LINK}(R)$ and $\lambda(R)$ in case the answer is positive.

(b) Each time the set $T(R)$ is changed, we find in the set $\{S'T' | S'T' \in \text{CHAIN}(T'), T' \in T(R)\}$ a bridge of minimal level which presently passes the legality test, and we assign this bridge as $\text{LINK}(R)$ (if no such a bridge exists, we assign $\text{LINK}(R) \leftarrow 0$, $\ell(R) \leftarrow \lfloor \frac{n}{2} \rfloor + 1$).

(ii) Each time the present $\text{LINK}(R)$ is found to be illegal, another bridge is assigned as $\text{LINK}(R)$. In order to verify the minimality of the α -path which is defined by the new $\text{LINK}(R)$, we have to make sure that the new $\text{LINK}(R)$ has a minimal level among the bridges which can presently be assigned as $\text{LINK}(R)$ and pass the legality test. Notice that according to the procedure of verifying the legality, we perform a legality test on the present $\text{LINK}(R)$ only when the set $T(R)$ is changed. Therefore case (ii) is covered by rule (b) of case (i).

In section G, throughout the proving of Theorem 1.5, we show that the following claim is true: "If there exists in the graph a legal alternating path of length j which is leading to R , then at the end of the SEARCH part of the j -th substage, the set $\{S'T' | S'T' \in \text{CHAIN}(T'), T' \in T(R)\}$ contains at least one bridge which can define for R a legal level $\leq j$ ". This claim implies that if at the end of the SEARCH part of the j -th substage $\ell(R) = j$, then it is minimum. (For, assume that there exists in the graph a legal alternating path of length i , $i < j$, which is leading to R . Then at the end of the SEARCH part of the i -th substage the set $\{S'T' | S'T' \in \text{CHAIN}(T'), T' \in T(R)\}$ contains a bridge $S_0 T_0$ which defines

for R a legal level $\leq i$. According to our procedure we would find this bridge and would assign for R a legal level $\lambda(R)$ where $\lambda(R) \leq i < j$ — in contradiction to the fact that $\lambda(R) = j$. Thus, if in the j -th substage $\lambda(R) = j$, then $\lambda(R)$ is minimum, and no further verifications are needed.

The combined procedure. Both the procedure for verifying the legality and the procedure for verifying the minimality, require tests which are performed exactly each time a new well-defined bridge is searched and each time a set $T(R)$ is changed. Moreover, if $\lambda(R) = j$, then according to both procedures no tests are required after the SEARCH part of the j -th substage. Thus, we can combine these two procedures into one procedure which simultaneously verifies both the legality and the minimality of $\lambda(R)$. This procedure has three rules:

(a) Each time a well-defined bridge ST is searched and passes the legality test, attach it to the end of $CHAIN(T)$; let $R \leftarrow TAIL(T)$; by using Eq. (1.3) check whether the assignment $LINK(R) \leftarrow ST$ can lower the present $\lambda(R)$, and if the answer is positive, perform these changes on $LINK(R)$ and $\lambda(R)$.

(b) Each time a set $T(R)$ is changed, check the set $\{ST | ST \in CHAIN(T), T \in T(R), S \notin T(R)\}$; if this set is empty, then assign: $LINK(R) \leftarrow G$, $\lambda(R) \leftarrow \lfloor \frac{n}{2} \rfloor + 1$; else, let $S_0 T_0$ be a bridge of minimum level in this set, and assign: $LINK(R) \leftarrow S_0 T_0$, $\lambda(R) \leftarrow \lambda(S_0 T_0) + 1 - \lambda(MATE(R))$.

(c) If at the end of the SEARCH part of the j -th substage $\ell(R)=j$, then $\ell(R)$, $\text{LINK}(R)$ and $p(0 \rightarrow R)$ are well-defined.

E. The Performance of the j -th Substage

The SEARCH part. During the SEARCH part of the j -th substage, all the free edges that are incident at vertices whose (well-defined) level is $j-1$ are searched. Let X be a vertex of (well-defined) level $j-1$, XY be a free edge, and $Z=\text{MATE}(Y)$. The edge XY is searched from X to Y , and an attempt is made to concatenate Y and Z to $p(0 \rightarrow X)$. Four different cases may arise (see Fig. 1.6):

(a) Both Y and Z are not well-defined (Fig. 1.6(a)). Thus, both vertices must have the dummy level $\lfloor \frac{n}{2} \rfloor + 1$. In this case the conditions for the 1-link assignment are satisfied, and therefore we assign: $\text{LINK}(Z) \leftarrow X$; $\ell(Z) \leftarrow j$. The a_1 -path $p(0 \rightarrow Z)$ (which is constructed by concatenating Y and Z to $p(0 \rightarrow X)$) is minimum and legal (Theorem 1.1), and therefore Z is declared to be well-defined. (By Theorem 1.3, no set $T(R)$ is changed by this declaration).

(b) Y is not well-defined, while Z is well-defined (Fig. 1.6 (b)). Thus, Z must have a 1-link. Since $\text{LINK}(Z) \neq X$ (for, Z has a 1-link before XY is searched), and $\text{LINK}(\text{MATE}(X)) \neq Y$ (for, Y has no well-defined level, and thus YX has not been searched yet), then XY is a bridge. Since Y has no well-defined level, XY is a potential bridge, and thus it cannot yet be used as a 2-link; we

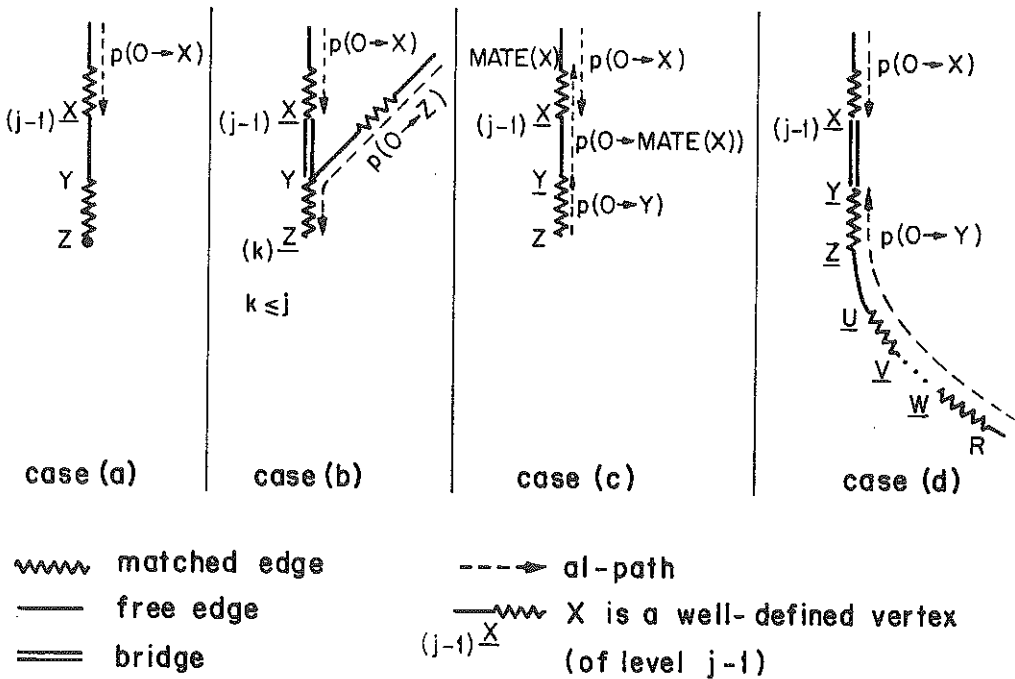


Fig. 1.6 - The SEARCH part of the j -th substage

attach XY to the end of $\text{CHAIN}(Y)$ without testing its legality.

(c) Y is well-defined and $\text{LINK}(\text{MATE}(X))=Y$ (Fig. 1.6(c)).

Clearly XY is not a bridge, and we cannot perform the assignment $\text{LINK}(Z) \leftarrow X$ either. Thus, nothing is done.

(d) Y is well-defined and $\text{LINK}(\text{MATE}(X)) \neq Y$. In this case XY

is a bridge (for, $\text{LINK}(Z) \neq X$). Since Y is well-defined, XY is a well-defined bridge, and thus we perform on it the legality test. If it fails the test - nothing is done. Else, we attach XY to the end of $\text{CHAIN}(Y)$. Let R be $\text{TAIL}(Y)$ (thus, R is the vertex closest to Y on $p(0 \rightarrow Y)$ whose LINK is still potential). We check whether the assignment $\text{LINK}(R) \leftarrow XY$ lowers $\ell(R)$, and perform these changes on $\text{LINK}(R)$ and $\ell(R)$ in case the answer is positive (Fig. 1.6(d)).

Thus, at the end of the SEARCH part of the j -th substage, exactly those vertices whose levels are less than j , and those vertices with a 1-link and level j , are declared to be well-defined.

The UPDATE part. During the UPDATE part of the j -th substage we search through the vertices which have a 2-link and (potential) level j . We declare these vertices to be well-defined and perform the changes which are required by these declarations on the variables TAIL and the sets $T(R)$.

First, we search through the vertices which have a (potential) level j and we look for a 2-exposed vertex. If such a vertex exists, then a minimum legal augmenting path of length j is found in the graph. By Corollary 1.4 there is no need to find the well-defined levels of the vertices whose present levels are still potential (i.e., are greater than j). Thus, we proceed to the second stage of the algorithm.

Assume that no 2-exposed vertex of level j exists in the graph, and let R be a vertex such that $\text{LINK}(R) = ST$ and $\ell(R) = j$.

Clearly $MATE(R)$ has a (well-defined) 1-link. Therefore, (according to case (c) of Theorem 1.0 in Section C), after R is declared to be well-defined, $T(R)$ becomes empty. Let V be a vertex such that before R is declared to be well-defined $V \in T(R)$ (i.e., $TAIL(V)=R$). According to case (b) of Theorem 1.0 in Section C, $p(0 \rightarrow MATE(R))$ is a head of $p(0 \rightarrow V)$ and all the vertices on $p(MATE(R) \rightarrow V)$ are well-defined. After R is declared to be well-defined, $TAIL(V)$ is changed. It is not difficult to see that the new $TAIL(V)$ is the same as the new $TAIL(MATE(R))$. Thus, let R_2 be $TAIL(MATE(R))$ after R is declared to be well-defined [by Theorem 1.3 $R_2 = TAIL(LINK(MATE(R)))$; see Fig. 1.7]. Then, all the vertices which have been in $T(R)$ before R is declared to be well-defined, must be transferred to the set $T(R_2)$; namely:

$$T(R_2) \leftarrow T(R_2) \cup T(R), \quad T(R) \leftarrow \emptyset.$$

Since $T(R_2)$ is changed, (then by the discussion of Section D), we now search the set $\{S'T' \mid S'T' \in CHAIN(T'), T' \in T(R_2), S' \notin T(R_2)\}$ for a bridge $S_2 T_2$ of minimum level. If the set is empty, then we assign $LINK(R_2) \leftarrow 0$, $\ell(R_2) \leftarrow \lfloor \frac{n}{2} \rfloor + 1$. Else, we assign: $LINK(R_2) \leftarrow S_2 T_2$, $\ell(R_2) \leftarrow \ell(S_2 T_2) + 1 - \ell(MATE(R_2))$.

The declaration of R as well-defined changes $T(MATE(R))$ too: Before R is declared well-defined, $T(MATE(R))$ contains exactly the vertex R (case (a) of Theorem 1.0 in Section C). Thus, $TAIL(R)=MATE(R)$. After R is declared to be well-defined, $TAIL(R)$ becomes a vertex R_3 which is the vertex closest to R on $p(0 \rightarrow R)$ whose $LINK$ is still potential. [In Theorem 1.3 we show that

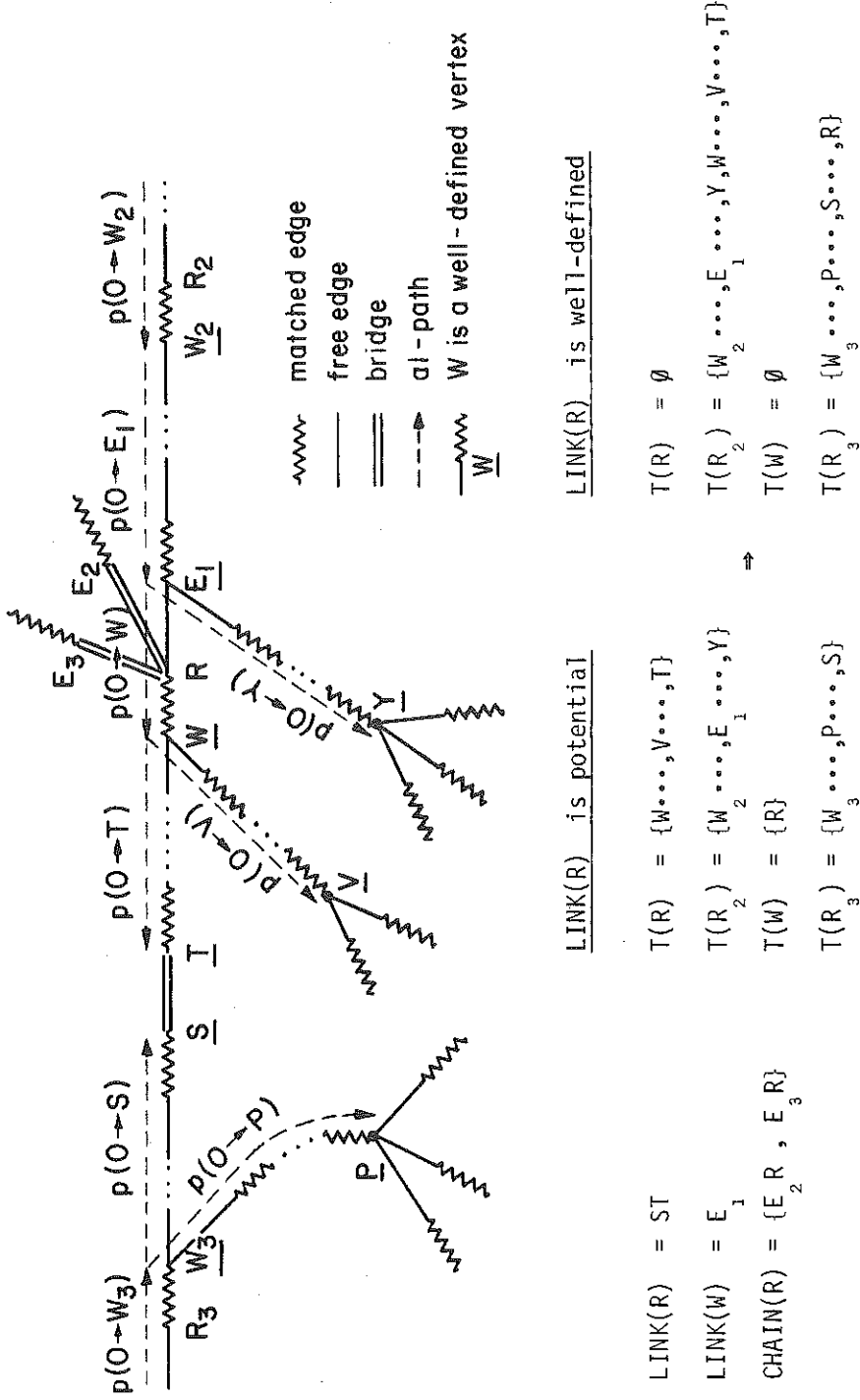


Fig. 1.7 - The UPDATE part of the j-th substage

$R_3 = \text{TAIL}(S)$, where $\text{LINK}(R) = \text{ST}$. Thus, when R is declared to be well-defined, we have to perform the operations:

$$T(R_3) \leftarrow T(R_3) \cup T(\text{MATE}(R)) , T(\text{MATE}(R)) \leftarrow \emptyset .$$

Again, since $T(R_3)$ is changed, we have to look in the set $\{S'T' \mid S'T' \in \text{CHAIN}(T'), T' \in T(R_3), S' \notin T(R_3)\}$ for a bridge $S_3 T_3$ of minimum level. If this set is empty, we assign: $\text{LINK}(R_3) \leftarrow 0$, $\ell(R_3) \leftarrow \lfloor \frac{n}{2} \rfloor + 1$. Else, we assign: $\text{LINK}(R_3) \leftarrow S_3 T_3$, $\ell(R_3) \leftarrow \ell(S_3 T_3) + 1 - \ell(\text{MATE}(R))$.

Subroutine MERGE. We define a subroutine $\text{MERGE}(R_2, R)$ which performs the following operations:

(a) $T(R_2) \leftarrow T(R_2) \cup T(R)$; $T(R) \leftarrow \emptyset$.

(b) Find in $\{S'T' \mid S'T' \in \text{CHAIN}(T'), T' \in T(R_2), S' \notin T(R_2)\}$ a bridge $S_2 T_2$ of minimum level. If the set is empty, then assign: $\text{LINK}(R_2) \leftarrow 0$, $\ell(R_2) \leftarrow \lfloor \frac{n}{2} \rfloor + 1$. Else, assign: $\text{LINK}(R_2) \leftarrow S_2 T_2$, $\ell(R_2) \leftarrow \ell(S_2 T_2) + 1 - \ell(\text{MATE}(R_2))$.

Since, (by the construction) the first bridge in $\text{CHAIN}(T')$ has always a minimum level in the CHAIN , we may replace operation (b) in MERGE by the following rule:

(b') Find in $\{S'T' \mid S'T' \text{ is first in } \text{CHAIN}(T'), T' \in T(R_2)\}$ a bridge $S_2 T_2$ of minimum level. If the set is empty, then assign: $\text{LINK}(R_2) \leftarrow 0$, $\ell(R_2) \leftarrow \lfloor \frac{n}{2} \rfloor + 1$. Else, perform the legality test on $S_2 T_2$: If $S_2 \in T(R_2)$ then delete $S_2 T_2$ from $\text{CHAIN}(T_2)$ and repeat (b'). Else, assign: $\text{LINK}(R_2) \leftarrow S_2 T_2$, $\ell(R_2) \leftarrow \ell(S_2 T_2) + 1 - \ell(\text{MATE}(R_2))$.

Using the subroutine MERGE , we can describe the performance of the UPDATE part of the j -th substage as follows:

(i) If there is in the graph a 2-exposed vertex whose level is j , then assign $r \leftarrow j$ [r is the length of a minimum legal augmenting path in the graph] and proceed to the second stage.

(ii) Else, for each vertex R , such that $\text{LINK}(R)=ST$ and $\ell(R)=j$, do the following operations: Declare R to be well-defined; call the subroutine $\text{MERGE}(R_2, R)$ [where R_2 is the vertex such that $\text{LINK}(\text{MATE}(R)) \in T(R_2)$]; Call the subroutine $\text{MERGE}(R_3, \text{MATE}(R))$ [where R_3 is the vertex such that $\text{SET}(R_3)$].

In the description of MERGE we ignored the variable TAIL . In fact this variable is not essential for the performance of the algorithm (this variable was introduced only to enable a better understanding of the sets $T(R)$). Some implementations of MERGE use this variable and it is updated when the operation $T(R_2) \leftarrow T(R_2) \cup T(R)$ is performed. Other implementations replace $\text{TAIL}(V)$ by the instruction "Find the vertex R such that $V \in T(R)$ " which is performed only when it is really required. Since different implementations of MERGE and different structures of $T(R)$ lead to different complexities of the algorithm, we shall discuss this subject in Section H: "The Complexity of the First Stage and the Implementation of MERGE ".

F. The Algorithm

0. [Initialization].

(a) For $i=1,2,\dots,\lfloor \frac{n}{2} \rfloor$ assign: $LIST(i) \leftarrow \emptyset$.

(b) For each vertex R assign: $CHAIN(R) \leftarrow \emptyset$; $T(R) \leftarrow \{MATE(R)\}$.

(c) For each 1-exposed vertex B [B belongs to $LIST(0)$] ,
assign: $\ell(B) \leftarrow 0$; $LINK(B) \leftarrow DUM$.

(d) For each vertex R which is not 1-exposed, insert R
into $LIST(\lfloor \frac{n}{2} \rfloor + 1)$ and assign: $LINK(R) \leftarrow 0$; $\ell(R) \leftarrow \lfloor \frac{n}{2} \rfloor + 1$.

(e) $j \leftarrow 1$.

SUBSTAGE:

SEARCH:

1. [Search through vertices.] If $LIST(j-1)$ contains no vertex which has not yet been searched, then go to UPDATE ; else let X be a vertex which has not yet been searched.
2. [Search through edges.] If all the free edges XY have already been searched then return to SEARCH ; else, let XY be a free edge which has not yet been searched.
3. Let $Z \leftarrow MATE(Y)$. If $\ell(Y) < j$ or Y has a 1-link, then [Y is well-defined] go to 6.
4. [Y is not well-defined.] If $LINK(Z) \neq 0$ then [XY is a

(potential) bridge] go to 5; else, assign [a 1-link assignment]: $\text{LINK}(Z) \leftarrow X$; $\ell(Z) \leftarrow j$ [Z is declared to be well-defined]; delete Z from $\text{LIST}(\lfloor \frac{n}{2} \rfloor + 1)$ and insert Z into $\text{LIST}(j)$; return to 2.

5. [XY is a (potential) bridge.] Attach XY to the end of $\text{CHAIN}(Y)$ and return to 2.
6. [Y is well-defined.] If $\text{LINK}(\text{MATE}(X)) = Y$ then [XY is not a bridge] return to 2; else [XY is a well-defined bridge], find the vertex R such that $Y \in \text{ET}(R)$. If $X \in \text{ET}(R)$ then [XY closes a loop] return to 2.
7. [XY is a (well-defined) bridge which presently does not close a loop.] Attach XY to the end of $\text{CHAIN}(Y)$. Assign: $\ell_1 \leftarrow \ell(XY) + 1 - \ell(\text{MATE}(R))$. If $\ell_1 \geq \ell(R)$ then [XY does not define for R a lower level than the present $\ell(R)$] return to 2.
8. [A 2-link assignment.] Delete R from $\text{LIST}(\ell(R))$; insert R into $\text{LIST}(\ell_1)$; assign: $\ell(R) \leftarrow \ell_1$; $\text{LINK}(R) \leftarrow XY$; return to 2.

UPDATE:

9. [Search through $\text{LIST}(j)$].

(a) If $\text{LIST}(j)$ contains a 2-exposed vertex, then assign:
 $r \leftarrow j$ and proceed to the second stage [r is the length

of a minimum legal augmenting path in the graph].

(b) If $LIST(j)$ contains no vertex which has a 2-link, then go to END OF SUBSTAGE .

10. If $LIST(j)$ contains no vertex which has a 2-link and has not yet been declared to be well-defined, then go to END OF SUBSTAGE . Else, let R be a vertex such that R has a 2-link and R has not yet been declared to be well-defined. Let $ST \leftarrow LINK(R)$.

11. [R is declared to be well-defined.] Find the vertex R_2 such that $LINK(MATE(R)) \in T(R_2)$. Find the vertex R_3 such that $SET(R_3)$.

12. Call $MERGE^*(R_2, R)$; call $MERGE(R_3, MATE(R))$; return to 10.

END OF SUBSTAGE.

13. If $j < \lfloor \frac{n}{2} \rfloor$ then assign: $j \leftarrow j+1$ and go to SUBSTAGE .

14. [The present matching is maximum.] PRINT the variables MATE [which give the matching] and the list $LIST(0)$ [which gives the exposed vertices]. HALT .

* For the implementation of MERGE , see Section H: "The Complexity of the First Stage and the Implementation of MERGE".

G. The Correctness of the First Stage

In this section we prove that the algorithm of the first stage, as described in Section F, is correct; i.e., that the goals of the first stage are achieved by this algorithm. In other words, we prove the following main theorem:

Main Theorem.

If during the performance of the algorithm we reach SUBSTAGE for the j -th time, then for each $i < j$, LIST(i) contains exactly the vertices of legal level i , and for each $R \in \text{LIST}(i)$, LINK(R) defines a legal minimum a -path of length i , leading to R .

The inductive hypothesis. The main theorem is proved by induction on j . In the inductive hypothesis we also assume that each time SUBSTAGE is reached, the sets $T(R)$ as constructed by the algorithm, are the same as implied by the definition.

Proof of the Main Theorem.

- (a) The inductive hypothesis holds for $j=1$ (by step 0 of the algorithm).
- (b) The inductive step for j is proved by the following sequence of lemmas and theorems.

Theorem 1.1.

A 1-link assignment is well-defined.

Proof. A 1-link assignment is performed in step 4, which is reached from step 3 in case Y is not well-defined (Y has no 1-link and $\ell(Y) \geq j$), and only in case $\text{LINK}(Z)=0$. Clearly Y cannot have a 2-link (otherwise Z must have a 1-link) and therefore $\text{LINK}(Y)=0$. Thus the 1-link assignment in step 4 of the algorithm is in accordance with the definition. By the inductive hypothesis X has a well-defined level $j-1$, and $p(0 \rightarrow X)$ is a legal a_1 -path of length $j-1$. Assume that the matched edge YZ is lying on $p(0 \rightarrow X)$: If Z is f -lying on $p(0 \rightarrow X)$ then the segment $p(0 \rightarrow Z)$ of the a_1 -path $p(0 \rightarrow X)$ is a legal alternating path of length less than $j-1$ which is leading to Z , and thus (by the inductive hypothesis), Z must have a well-defined level less than $j-1$, and $\text{LINK}(Z) \neq 0$. A similar contradiction is achieved if we assume that Y is f -lying on $p(0 \rightarrow X)$. Thus, the edge YZ is disjoint from $p(0 \rightarrow X)$, and therefore the a_1 -path $p(0 \rightarrow X) \cdot Y \cdot Z$ (which is defined by $\text{LINK}(Z)=X$) is legal. Since the length of this path is j , it is also a minimum a_1 -path leading to Z (otherwise, by the inductive hypothesis, $\text{LINK}(Z)$ would not be 0). Thus, $\text{LINK}(Z)=X$ and $\ell(Z)=j$ are well-defined.

Q.E.D.

Lemma 1.1.

(a) If a free edge ST is searched and found to be a bridge,

then it will remain a bridge throughout the remainder of the first stage.

(b) If, when ST is searched, it is not found to be a bridge, then it will never become a bridge.

Proof. (a) Assume that ST is searched and found to be a bridge. When ST is searched, at least one of the vertices T and $MATE(T)$ must have a LINK other than 0 (otherwise, we would reach step 4 of the algorithm and would assign $LINK(MATE(T)) \leftarrow S$. In this case ST would not be a bridge). Therefore, when ST is searched, either T or $MATE(T)$ must have a 1-link, which (by Theorem 1.1) is well-defined and therefore is never changed. If T has a 1-link, then $MATE(T)$ will never have a 1-link too, and thus $LINK(MATE(T))=S$ will never hold. If $MATE(T)$ has a 1-link, then since presently $LINK(MATE(T)) \neq S$, this will remain true throughout the remainder of the first stage. Thus, if ST is found to be a bridge, then $LINK(MATE(T))=S$ will never hold.

Since ST is searched, S must have a well-defined LINK. If S has a 1-link, then $MATE(S)$ will never have a 1-link too, and thus $LINK(MATE(S))=T$ will never hold. If S has a 2-link, then $MATE(S)$ has a 1-link which (by Theorem 1.1) is well-defined and therefore is never changed. Since presently $LINK(MATE(S)) \neq T$, this will remain true throughout the remainder of the first stage. Thus, if ST is found to be a bridge, then $LINK(MATE(S))=T$ will never hold.

Therefore, if ST is searched and found to be a bridge, then

ST will remain a bridge throughout the remainder of the first stage.

(b) Assume that ST is searched and is not found to be a bridge. Then either $\text{LINK}(\text{MATE}(T))=S$ or $\text{LINK}(\text{MATE}(S))=T$. Since by Theorem 1.1 a 1-link is well-defined and is never changed, ST will never become a bridge.

Q.E.D.

Theorem 1.2.

For each Y, CHAIN(Y) contains exactly all bridges of the form XY except those which have failed the legality test. These bridges are listed in nondecreasing order of their levels.

Proof. (a) New bridges are inserted into CHAIN(Y) only in step 5 and step 7 of the algorithm. Step 5 is reached when XY is being searched and $\text{LINK}(Z) \neq 0$ (where $Z=\text{MATE}(Y)$). Clearly in this case $\text{LINK}(Z) \neq X$. Also, when step 5 is reached, $\ell(Y) \geq j$ (by step 3) and therefore the edge YX has not been searched yet, and thus $\text{LINK}(\text{MATE}(X)) \neq Y$. Therefore, when step 5 is reached, XY is found to be a bridge, and by Lemma 1.1 it will remain a bridge throughout the first stage. Step 7 is reached when $\ell(Y) < j$ or Y has a 1-link (step 3) and when $\text{LINK}(\text{MATE}(X)) \neq Y$ (step 6). If Y has a 1-link, then Z will never have a 1-link and thus $\text{LINK}(Z) \neq X$. If Y has no 1-link, then $\ell(Y) < j$ and thus Y must have a 2-link. This implies that when XY is searched, Z already have a 1-link and thus $\text{LINK}(Z) \neq X$. Since by step 6 $\text{LINK}(\text{MATE}(X)) \neq Y$, then XY is

found to be a bridge, and by Lemma 1.1 it will remain a bridge throughout the first stage. Thus, $\text{CHAIN}(Y)$ contains only bridges of the form XY .

(b) Assume that XY is searched (step 2 of the algorithm). If we do not reach step 5 or step 7 of the algorithm, then either we assign $\text{LINK}(Z)+X$ (in step 4) or we find that $\text{LINK}(\text{MATE}(X))=Y$, or we find that X and Y belong to the same set $T(R)$ (in step 6). In the first two cases XY is not a bridge and therefore (by Lemma 1.1) it will never become one. In the third case XY is a bridge which fails the legality test. Thus, in step 5 and 7 we insert into $\text{CHAIN}(Y)$ all the bridges of the form XY , except those which have failed the legality test in step 6.

(c) A bridge XY is deleted from $\text{CHAIN}(Y)$ if and only if it fails the legality test during the performance of subroutine MERGE (see Section E). Therefore, $\text{CHAIN}(Y)$ contains exactly all bridges of the form XY which have not failed the legality test.

(d) Each time a new bridge is inserted into $\text{CHAIN}(Y)$ (step 5 and step 7), it is attached to the end of CHAIN . Therefore, the bridges of $\text{CHAIN}(Y)$ are listed according to the order by which they have been searched. Thus, a bridge $X_i Y$ precedes another bridge $X_j Y$ in $\text{CHAIN}(Y)$ only if $X_i Y$ has been searched before $X_j Y$, i.e., only if $\ell(X_i) \leq \ell(X_j)$ or $\ell(X_i Y) \leq \ell(X_j Y)$. Clearly, the deletions of bridges from $\text{CHAIN}(Y)$ do not disturb this order.

Q.E.D.

Theorem 1.3.

For each vertex E , the set $T(E)$ as constructed by the algorithm, is in agreement with the definition of $T(E)$.

Proof. According to the inductive hypothesis, when SUBSTAGE is reached in the beginning of the j -th substage, the sets $T(E)$ are in agreement with the definition. Throughout the j -th substage, changes of the sets $T(E)$ occur only in subroutine MERGE which is activated in step 12 of the algorithm. Thus, we have to prove the following two claims:

(i) The changes of the sets $T(E)$ as implied by the activation of subroutine MERGE in step 12 of the algorithm are in accordance with the definition.

(ii) All the changes of the sets $T(E)$ which are forced by the definition, are correctly performed by subroutine MERGE in step 12 of the algorithm.

Proof of Claim (i). Step 12 in the algorithm is reached immediately after R is declared to be well-defined. During step 12, subroutine MERGE is activated twice.

Consider first the call of $MERGE(R_2, R)$, where R_2 is the vertex such that $LINK(MATE(R)) \in T(R_2)$ (i.e., $TAIL(LINK(MATE(R))) = R_2$): During the performance of $MERGE(R_2, R)$ two changes are made on the sets $T(E)$: $T(R_2) \leftarrow T(R_2) \cup T(R)$ and $T(R) \leftarrow \emptyset$. Let V be a vertex such that before R has been declared to be well-defined, $V \in T(R)$, i.e.,

$TAIL(V)=R$ (see Fig. 1.7). Assume that V is not well-defined, then by the definition of $TAIL$, $V=MATE(R)$ - which is a contradiction (for $MATE(R)$ has a 1-link and therefore is well-defined). Thus, V is well-defined, and R is the vertex closest to V on $p(0 \rightarrow V)$ whose $LINK$ is still potential. Denote $E_1 \leftarrow LINK(MATE(R))$, then E_1 is well-defined (by the definition of 1-link), and thus R_2 (where $R_2 = TAIL(E_1)$) is the vertex closest to E_1 on $p(0 \rightarrow E_1)$ whose $LINK$ is still potential. Since $p(0 \rightarrow V) = p(0 \rightarrow E_1) \cdot p(R \rightarrow V)$, then after R is declared to be well-defined R_2 is the vertex closest to V on $p(0 \rightarrow V)$ whose $LINK$ is still potential. Therefore, after R is declared to be well-defined, $TAIL(V)=R_2$ and thus the merging $T(R_2) \leftarrow T(R_2) \cup T(R)$ is correct. Clearly, (by case (c) of Theorem 1.0 in Section C), the assignment $T(R) \leftarrow \emptyset$ is correct too.

Consider the call of $MERGE(R_3, MATE(R))$, where R_3 is the vertex such that $SET(R_3)$ (i.e., $TAIL(S)=R_3$), and $LINK(R)=ST$ (see Fig. 1.7): According to case (a) of Theorem 1.0 in Section C, before R is declared to be well-defined, $T(MATE(R))=\{R\}$, i.e., $TAIL(R)=MATE(R)$. After R is declared to be well-defined, $TAIL(R)$ is the vertex closest to R on $p(0 \rightarrow R)$ whose $LINK$ is still potential. Since $LINK(R)=ST$, then $p(0 \rightarrow R) = p(0 \rightarrow S) \cdot p(R \rightarrow T)$ and all the vertices on $p(R \rightarrow T)$ are well-defined. Thus, after R is declared to be well-defined, $TAIL(R)$ is the vertex closest to S on $p(0 \rightarrow S)$ whose $LINK$ is still potential, namely $TAIL(R)=TAIL(S)$, i.e., $TAIL(R)=R_3$. Therefore the operation $T(R_3) \leftarrow T(R_3) \cup T(MATE(R))$ is correct. Clearly, (by case (c) of Theorem 1.0 in Section C), the assignment $T(MATE(R)) \leftarrow \emptyset$ is correct too.

Proof of Claim (ii). Assume that according to the definition, the set $T(E)$ must be changed.

(a) Assume that before the change has occurred, $MATE(E)$ was not well-defined. Then by case (a) of Theorem 1.0 in Section C, before the change has occurred $T(E) = \{MATE(E)\}$. The only change that can happen in this case is that the LINK of $MATE(E)$ is declared to be well-defined.

Assume that $MATE(E)$ has a 1-link. Then, by case (b) of Theorem 1.0 in Section C, after $MATE(E)$ is declared to be well-defined, a vertex V belongs to $T(E)$ if and only if $p(0 \rightarrow MATE(E))$ is a head of $p(0 \rightarrow V)$ and all the vertices on $p(MATE(E) \rightarrow V)$ are well-defined. Since we are in the j -th substage, $MATE(E)$ has a well-defined level j . Thus, if $V \neq MATE(E)$ then V must have a well-defined level greater than j - and this is impossible during the j -th substage. Thus, the only vertex in $T(E)$ is $MATE(E)$, and therefore no change has to be made. And indeed nothing is done in this case in the algorithm.

Assume that $MATE(E)$ has a 2-link. Then, by case (c) of Theorem 1.0, after $MATE(E)$ is declared to be well-defined $T(E)$ becomes empty. And indeed, when $MATE(E)$ is declared to be well-defined, subroutine $MERGE(R_3, E)$ is called, and $T(E) \leftarrow \emptyset$ is performed.

(b) Assume that before the change has occurred, $MATE(E)$ was well-defined but E was still potential. Then two changes of $T(E)$ can happen: E becomes well-defined or, by the definition, a new

vertex V must be inserted into $T(E)$.

If E becomes well-defined, then, by case (c) of Theorem 1.0, $T(E)$ becomes empty. And indeed, when E is declared to be well-defined, subroutine $MERGE(R_2, E)$ is called, and $T(E) \leftarrow \emptyset$ is performed.

Assume that, by the definition, a new vertex V must be inserted into $T(E)$, and let R be the vertex such that before the change has occurred $V \in T(R)$ (i.e., $TAIL(V)=R$). If, before the change has occurred, V was not well-defined, then $R=MATE(V)$, and a change of $T(R)$ can take place only if the 2-link of V is declared to be well-defined (see case (a) of this proof). And indeed, in this case we call $MERGE(R_3, R)$, which transfers V into the correct set $T(E)$ [as we have already shown in the proof of Claim (i)]. If, before the change has occurred, V was well-defined, then R was the vertex closest to V on $p(0 \rightarrow V)$ whose LINK was still potential, and the change has occurred by declaring the 2-link of R to be well-defined. And indeed, in this case we call $MERGE(R_2, R)$ which transfers the vertices of $T(R)$ into the correct set $T(E)$ [as we have already shown in the proof of Claim (i)].

(c) Assume that both $MATE(E)$ and E are well-defined. Then (by case (c) of Theorem 1.0 in Section C) $T(E)=\emptyset$, and no change can occur.

Q.E.D.

Corollary 1.1.

The 2-link assignments in the algorithm are in accordance with the definition of 2-link.

Proof. The 2-link assignments are performed in the algorithm in step 8 and during the performance of subroutine MERGE . In both cases the assignments are of the form $LINK(R) \leftarrow ST$, where $T \in T(R)$, and $ST \in CHAIN(T)$. By Corollary 1.0 in Section C, all the bridges of the set $\{S'T' \mid S'T' \in CHAIN(T'), T' \in T(R)\}$ can be assigned as $LINK(R)$ according to the definition of 2-link. By Theorem 1.3 the set $T(R)$ as constructed by the algorithm is in agreement with the definition of $T(R)$. Therefore the assignment $LINK(R) \leftarrow ST$ is in accordance with the definition.

Q.E.D.

Theorem 1.4.

(a) Let $LINK(R) = ST$. If $S \in T(R)$, then the 2-link $LINK(R) = ST$ is illegal, and the bridge ST cannot be used later to define a legal 2-link for any other vertex whose $LINK$ is still potential.

(b) Let R be a vertex such that at the end of the SEARCH part of the j -th substage R is not yet declared to be well-defined. Let ST be a bridge which can be assigned as $LINK(R)$ such that this assignment defines for R a level $\leq j$. If $LINK(R) = ST$ is illegal, then $S \in T(R)$.

Proof. (a) Assume that $LINK(R) = ST$ and $S \in T(R)$. By case (b) of Theorem 1.0 in Section C, $p(O \rightarrow MATE(R))$ is a head of $p(O \rightarrow S)$. On

the other hand, the a_1 -path which is defined by $LINK(R)=ST$ is $p(0 \rightarrow R) = p(0 \rightarrow S) \cdot p(R \leftarrow T)$, and thus $p(0 \rightarrow S)$ is a head of $p(0 \rightarrow R)$. Therefore, $p(0 \rightarrow MATE(R))$ is a head of $p(0 \rightarrow R)$, which means that $p(0 \rightarrow R)$ contains an odd loop and is therefore illegal.

Since $LINK(R)=ST$, by the definition of 2-link $p(0 \rightarrow MATE(R))$ is a head of $p(0 \rightarrow T)$ and all the vertices on $p(MATE(R) \rightarrow T)$ are well-defined. Assume that ST can be used later to define a 2-link for a vertex E whose $LINK$ is still potential. By the definition of 2-link, for this to happen, $p(0 \rightarrow MATE(E))$ must be a head of $p(0 \rightarrow T)$. Thus, either $p(0 \rightarrow MATE(E))$ is a head of $p(0 \rightarrow MATE(R))$ or vice versa. Since presently E is not well-defined while all vertices on $p(MATE(R) \rightarrow T)$ are well-defined, $p(0 \rightarrow MATE(E))$ is a head of $p(0 \rightarrow MATE(R))$. On the other hand $p(0 \rightarrow MATE(R))$ is a head of $p(0 \rightarrow S)$ (as implied by $SET(R)$). Therefore, $p(0 \rightarrow MATE(E))$ is a head of $p(0 \rightarrow S)$. If we assign $LINK(E) \leftarrow ST$, then $p(0 \rightarrow S)$ becomes a head of $p(0 \rightarrow E)$. In this case, $p(0 \rightarrow MATE(E))$ becomes a head of $p(0 \rightarrow E)$ and thus the assignment $LINK(E) \leftarrow ST$ is illegal.

Q.E.D.

(b) The proof of part (b) of Theorem 1.4 is described in Appendix A.

Corollary 1.2.

If at the end of the SEARCH part of the j -th substage $l(R)=j$, then R is a vertex of minimum legal level j .

Proof. If R has a 1-link, then (by Theorem 1.1) R is well-defined, i.e., has a minimum legal level j .

Assume that R has a 2-link. Consider the last time $T(R)$ has been changed before the j -th substage. A change of $T(R)$ may occur only during the performance of subroutine MERGE. When this happens we also find a bridge $S_0 T_0$ of minimum level in the set $\{S'T' \mid S'T' \in \text{CHAIN}(T'), T' \in T(R), S' \notin T(R)\}$, and we assign this bridge as $\text{LINK}(R)$.

If at the end of the SEARCH part of the j -th substage still $\text{LINK}(R) = S_0 T_0$, then $S_0 \notin T(R)$ (for, $T(R)$ has not been changed since $S_0 T_0$ was assigned as $\text{LINK}(R)$). Therefore, by Theorem 1.4 $\text{LINK}(R)$ is legal. If at the end of the SEARCH part of the j -th substage $\text{LINK}(R) \neq S_0 T_0$, then the present $\text{LINK}(R)$ was assigned in step 8 of the algorithm. Let $\text{LINK}(R) = ST$. Since step 8 could be reached from steps 6 and 7 only in case $S \notin T(R)$, and since $T(R)$ has not been changed after ST was assigned as $\text{LINK}(R)$ - by Theorem 1.4, $\text{LINK}(R)$ is legal.

Thus, in either case, $\text{LINK}(R)$ is legal. If R has a legal level less than j , then by the inductive hypothesis $\ell(R) < j$. Hence $\ell(R) = j$ implies that j is the minimum legal level of R .

Q.E.D.

Theorem 1.5.

If there exists in the graph a legal alternating path of length j which is leading to R , then at the end of the SEARCH part of the j -th substage $\ell(R) \leq j$.

Proof. Let p_1 be a legal alternating path of length j which is leading from a 1-exposed vertex A_1 to the vertex R . Clearly all vertices which are f -lying on p_1 (except R) have levels less than j , and therefore (by the inductive hypothesis) are already well-defined in the beginning of the j -th substage. Let $W = \text{MATE}(R)$, and let U be the vertex which precedes W on p_1 (thus, U is f -lying on p_1 , and UW is a free edge on p_1). Since $\ell(U) < j$, the edge UW has been searched before the end of the SEARCH part of the j -th substage. If when UW is searched W has no well-defined level, then either R has a 1-link, or (according to step 4 of the algorithm), it is given a 1-link. In both cases, at the end of the SEARCH part of the j -th substage, R has a well-defined level not greater than j , and thus the theorem holds. Therefore, assume that when UW is searched W is already well-defined. Thus, at the end of the SEARCH part of the j -th substage, W has a 1-link while R is not yet well-defined. It follows that $\text{MATE}(R) \in T(R)$, i.e., $T(R) \neq \emptyset$.

Claim 1. If at the end of the SEARCH part of the j -th substage, the set $\{S'T' \mid S'T' \in \text{CHAIN}(T'), T' \in T(R), S' \notin T(R)\}$ contains at least one bridge $S_0 T_0$ which can define for R a level not greater

than j , then $\ell(R) \leq j$.

Proof of Claim 1. Consider the last time $T(R)$ has been changed before the j -th substage. Such a change may occur only during a performance of the subroutine MERGE. By the definition of MERGE, after $T(R)$ is changed we find a bridge $S_2 T_2$ of minimum level in the set $\{S' T' \mid S' T' \in \text{CHAIN}(T'), T' \in T(R), S' \notin T(R)\}$, and we assign: $\text{LINK}(R) \leftarrow S_2 T_2$, $\ell(R) \leftarrow \ell_2$ (where $\ell_2 = \ell(S_2 T_2) + 1 - \ell(\text{MATE}(R))$). After this assignment of $\ell(R)$ is performed, $\ell(R)$ can be changed only by step 8 of the algorithm, in which case it becomes lower. Therefore, if $\ell_2 \leq j$, then at the end of the SEARCH part of the j -th substage $\ell(R) \leq j$.

Assume that $\ell_2 > j$. At the end of the SEARCH part of the j -th substage $T_0 \in T(R)$ and $S_0 \notin T(R)$. If $S_0 T_0$ is searched before the assignment $\ell(R) \leftarrow \ell_2$ is performed, then, when this assignment is performed, $S_0 T_0$ is already contained in the set $\{S' T' \mid S' T' \in \text{CHAIN}(T'), T' \in T(R), S' \notin T(R)\}$, and thus we would assign $\text{LINK}(R) \leftarrow S_0 T_0$, $\ell(R) \leftarrow j$ — in contradiction to the assumption that $\ell_2 > j$. Therefore, $S_0 T_0$ is searched after the last change of $T(R)$ occurs. It follows that when $S_0 T_0$ is searched, T_0 is already contained in $T(R)$ and thus by step 8 of the algorithm we assign: $\text{LINK}(R) \leftarrow S_0 T_0$, $\ell(R) \leftarrow j$. Since any further assignment of $\ell(R)$ by step 8 can only lower $\ell(R)$, it follows that at the end of the SEARCH part of the j -th substage $\ell(R) \leq j$.

Q.E.D.

Claim 2. Let YZ be a matched [free] edge on p_1 , such that at the end of the SEARCH part of the j -th substage Y and Z are already well-defined and $Y \in T(R)$. Then either $\ell(R) < j$, or $\ell(Y) + \ell(Z) - \ell(W) \leq j$ [$\ell(Y) + \ell(Z) + 1 - \ell(W) \leq j$], where $W = \text{MATE}(R)$.

Proof of Claim 2. Denote by $V_1 \xrightarrow{1} V_2$ the segment of p_1 from V_1 to V_2 . Denote by $V_1 \xrightarrow{2} V_2$ the segment of $p(O \rightarrow Y)$ from V_1 to V_2 . Let $|V_1 \xrightarrow{1} V_2|$ be the length of a segment $V_1 \xrightarrow{1} V_2$.

Let B_1 be the 1-exposed vertex from which $p(O \rightarrow Y)$ starts (thus, $p(O \rightarrow Y)$ is $B_1 \xrightarrow{2} Y$). By case (b) of Theorem 1.0 in Section C, $B_1 \xrightarrow{2} W$ is a head of $B_1 \xrightarrow{2} Y$. Denote by V_1 this vertex of Y and Z which is f -lying on p_1 , and let V_2 be the other vertex. Clearly $\ell(V_1) \leq |A_1 \xrightarrow{1} V_1|$. If $V_2 \xrightarrow{1} W$ and $B_1 \xrightarrow{2} W$ are disjoint (except for the vertex W), then $B_1 \xrightarrow{2} W \xrightarrow{1} V_2$ is a legal alternating path leading to V_2 , and thus $\ell(V_2) \leq |B_1 \xrightarrow{2} W| + |W \xrightarrow{1} V_2| = \ell(W) + |W \xrightarrow{1} V_2|$. Therefore: $\ell(Y) + \ell(Z) - \ell(W) = \ell(V_1) + \ell(V_2) - \ell(W) \leq |A_1 \xrightarrow{1} V_1| + |V_2 \xrightarrow{1} W| = |A_1 \xrightarrow{1} R| = j$ [if YZ is a free edge:

$\ell(Y) + \ell(Z) + 1 - \ell(W) \leq |A_1 \xrightarrow{1} V_1| + 1 + |V_2 \xrightarrow{1} W| = |A_1 \xrightarrow{1} R| = j$]. Assume that $V_2 \xrightarrow{1} W$ and $B_1 \xrightarrow{2} W$ are not disjoint, and let L be the first vertex which is f -lying on $B_1 \xrightarrow{2} W$ and is common to $V_2 \xrightarrow{1} W$. If L is not f -lying on $V_2 \xrightarrow{1} W$, then $B_1 \xrightarrow{2} L \xrightarrow{1} V_2$ is a legal alternating path leading to V_2 , and thus:

$\ell(V_2) \leq |B_1 \xrightarrow{2} L| + |L \xrightarrow{1} V_2| < |B_1 \xrightarrow{2} W| + |W \xrightarrow{1} V_2| = \ell(W) + |W \xrightarrow{1} V_2|$, and again we get $\ell(Y) + \ell(Z) - \ell(W) \leq j$ [if YZ is a free edge:

$\ell(Y) + \ell(Z) + 1 - \ell(W) \leq j$].

Assume that L is f -lyign on $V_2 \xrightarrow{1} W$, then $B_1 \xrightarrow{2} L \xrightarrow{1} R$ is a legal alternating path leading to R . If $V_1 = Y$ (i.e., if Y is f -lying on p_1) then:

$$|B_1 \xrightarrow{2} L| + |L \xrightarrow{1} R| < |B_1 \xrightarrow{2} Y| + |Y \xrightarrow{1} R| \leq |A_1 \xrightarrow{1} Y| + |Y \xrightarrow{1} R| = |A_1 \xrightarrow{1} R| = j.$$

Thus, there exists a legal alternating path leading to R of length less than j , and therefore, by the inductive hypothesis $\ell(R) < j$.

Assume that $V_1 = Z$ (i.e., Z is f -lying on p_1). If

$$|W \xrightarrow{2} Y| \leq |W \xrightarrow{1} Y| \quad \text{then} \quad \ell(Y) = |B_1 \xrightarrow{2} W \xrightarrow{2} Y| \leq |B_1 \xrightarrow{2} W| + |W \xrightarrow{1} Y| = \ell(W) + |W \xrightarrow{1} Y|.$$

Since $\ell(Z) \leq |A_1 \xrightarrow{1} Z|$ it follows that

$$\ell(Y) + \ell(Z) - \ell(W) \leq |A_1 \xrightarrow{1} Z| + |Y \xrightarrow{1} W| = |A_1 \xrightarrow{1} R| = j, \quad [\text{if } YZ \text{ is a free}$$

edge: $\ell(Y) + \ell(Z) + 1 - \ell(W) \leq |A_1 \xrightarrow{1} Z| + 1 + |Y \xrightarrow{1} W| = |A_1 \xrightarrow{1} R| = j$]. If

$|W \xrightarrow{2} Y| > |W \xrightarrow{1} Y|$ then:

$$|B_1 \xrightarrow{2} L| + |L \xrightarrow{1} R| < |B_1 \xrightarrow{2} W| + |Y \xrightarrow{1} W| < |B_1 \xrightarrow{2} W| + |W \xrightarrow{2} Y| = |B_1 \xrightarrow{2} Y| = \ell(Y).$$

Since Y is well-defined, $\ell(Y) \leq j$. It follows that $B_1 \xrightarrow{2} L \xrightarrow{1} R$ is a legal alternating path of length less than j which is leading to R , and thus, by the inductive hypothesis, $\ell(R) < j$.

Q.E.D.

Claim 3. Let YZ be a matched edge on p_1 , such that at the end of the SEARCH part of the j -th substage Y and Z are already well-defined and $Y \in T(R)$ while $Z \notin T(R)$. Then $\ell(R) \leq j$.

Proof of Claim 3. Assume that $\ell(R) \geq j$, then by Claim 2

$\ell(Y) + \ell(Z) - \ell(W) \leq j$. Since at the end of the SEARCH part of the j -th substage both Y and Z are well-defined, one of them has a 2-link

to a bridge $S_0 T_0$.

Assume that $S_0 T_0 = \text{LINK}(Y)$. By the definition of 2-link, all the vertices on $p(T_0 \rightarrow Y)$ are well-defined. Therefore, at the end of the SEARCH part of the j -th substage, both T_0 and $\text{MATE}(T_0)$ are well-defined, and thus $\ell(T_0) \leq j$. By the algorithm, vertices of 2-link and level j are declared to be well-defined only during the UPDATE part of the j -th substage, and thus if $\ell(T_0) = j$ then T_0 must have a 1-link. However, in this case, $\text{MATE}(T_0)$ has a 2-link and $\ell(\text{MATE}(T_0)) \geq j$, and thus $\text{MATE}(T_0)$ cannot be well-defined—in contradiction to the requirement that all vertices on $p(T_0 \rightarrow Y)$ are well-defined. Hence $\ell(T_0) < j$, and $T_0 S_0$ has been searched and inserted into $\text{CHAIN}(S_0)$ before the end of the SEARCH part of the j -th substage. Since $\text{LINK}(Y) = S_0 T_0$, $p(O \rightarrow S_0)$ is a head of $p(O \rightarrow Y)$ and all the vertices on $p(S_0 \rightarrow Y)$ are well-defined - and thus $Y \in T(R)$ implies that $S_0 \in T(R)$. On the other hand (by $\text{LINK}(Y) = S_0 T_0$), $p(O \rightarrow Z)$ is a head of $p(O \rightarrow T_0)$, and all the vertices on $p(Z \rightarrow T_0)$ are well-defined. Therefore, T_0 and Z belong to the same set $T(R')$, and in particular $T_0 \notin T(R)$. It follows that at the end of the SEARCH part of the j -th substage, the bridge $T_0 S_0$ belongs to the set $\{S'T' \mid S'T' \in \text{CHAIN}(T'), T' \in T(R), S' \notin T(R)\}$.

By Eq. (1.3), $\ell(Y) = \ell(S_0 T_0) + 1 - \ell(Z)$, and thus $\ell(S_0 T_0) + 1 = \ell(Y) + \ell(Z)$. Therefore, the bridge $T_0 S_0$ can define for R a level ℓ_1 where: $\ell_1 = \ell(S_0 T_0) + 1 - \ell(W) = \ell(Y) + \ell(Z) - \ell(W) \leq j$. Hence, by Claim 1, $\ell(R) \leq j$.

Assume that $S_0 T_0 = \text{LINK}(Z)$. Since $\ell(S_0) < \ell(Z) \leq j$, the bridge

$S_0 T_0$ has been searched and inserted into $CHAIN(T_0)$ before the end of the SEARCH part of the j -th substage. Since $LINK(Z)=S_0 T_0$, $p(0 \rightarrow S_0)$ is a head of $p(0 \rightarrow Z)$ and all the vertices on $p(S_0 \rightarrow Z)$ are well-defined. Therefore S_0 and Z belong to the same set $T(R')$, and in particular $S_0 \notin T(R)$. On the other hand, since $p(0 \rightarrow Y)$ is a head of $p(0 \rightarrow T_0)$ and all vertices on $p(Y \rightarrow T_0)$ are well-defined, the relation $Y \in T(R)$ implies that $T_0 \in T(R)$. Thus, at the end of the SEARCH part of the j -th substage the bridge $S_0 T_0$ belongs to the set $\{S'T' \mid S'T' \in CHAIN(T'), T' \in T(R), S' \notin T(R)\}$, and it can be assigned as $LINK(R)$. By Eq. (1.3), this assignment defines for R a level ℓ_1 where: $\ell_1 = \ell(S_0 T_0) + 1 - \ell(W) = \ell(Y) + \ell(Z) - \ell(W) \leq j$. Hence, by Claim 1, $\ell(R) \leq j$.

Q.E.D.

Proof of Theorem 1.5. Let Q be the vertex closest to A_1 on p_1 , such that Q is not f -lying on p_1 , and at the end of the SEARCH part of the j -th substage $Q \in T(R)$ (such a vertex exists - e.g. $MATE(R)$). Clearly Q is well-defined, and thus $Q \neq MATE(A_1)$ (for, if $MATE(A_1)$ is already well-defined, then the algorithm would be terminated in step 9(a) before the j -th substage begins). Let P be the vertex which precedes Q on p_1 (i.e., P is f -lying on p_1 , and PQ is a free edge on p_1). Since $A_1 \xrightarrow{1} P$ is a legal alternating path of length less than j which is leading to P , then by the inductive hypothesis P is well-defined and $\ell(P) < j$. Therefore PQ has been searched before the end of the SEARCH part of the j -th substage.

Assume that at the end of the SEARCH part of the j -th substage $P \notin T(R)$. By the definition of $T(R)$, $MATE(P)$ is already well-defined and by the choice of Q , $MATE(P) \notin T(R)$. Therefore by Claim 3, $\ell(R) \leq j$.

Assume that at the end of the SEARCH part of the j -th substage, $P \notin T(R)$. If $LINK(MATE(P))=Q$, then $MATE(P)$ is well-defined and belongs to the same set as Q , i.e., $MATE(P) \in T(R)$. Thus, by Claim 3, $\ell(R) \leq j$. If $LINK(MATE(Q))=P$, then $MATE(Q)$ is well-defined and belongs to the same set $T(R')$ as P , i.e., $MATE(Q) \notin T(R)$. Again, by Claim 3, $\ell(R) \leq j$. If $LINK(MATE(P)) \neq Q$ and $LINK(MATE(Q)) \neq P$, then PQ is a bridge. Since PQ has been searched before the end of the SEARCH part of the j -th substage and $P \notin T(R)$ while $Q \in T(R)$, it follows that PQ belongs to the set $\{S'T' | S'T' \in CHAIN(T'), T' \in T(R), S' \notin T(R)\}$. Thus, PQ can define for R a legal level ℓ_1 , where (by Eq. (1.3)) $\ell_1 = \ell(P) + \ell(Q) + 1 - \ell(W)$. By Claim 2, either $\ell(R) < j$ or $\ell_1 \leq j$, where the latter case implies (by Claim 1) that $\ell(R) \leq j$.

Q.E.D.

Corollary 1.3

At the end of the SEARCH part of the j -th substage, for each $i \leq j$, $LIST(i)$ contains exactly all vertices of minimum legal level i .

Proof. If $i < j$, then by the inductive hypothesis, in the beginning of the j -th substage, $LIST(i)$ contains exactly all vertices

of minimum legal level i . During the SEARCH part of the algorithm, a vertex R can be deleted from $LIST(i)$ only in step 8, when a new 2-link XY is assigned to R . For this to happen, Y must belong to $T(R)$. However, by the definition of $T(R)$ and by Theorem 1.3, if R has a 2-link, and it belongs to $LIST(i)$ (where $i < j$), then $T(R) = \emptyset$. Thus, during the SEARCH part no vertex R is deleted from $LIST(i)$.

Assume that during the SEARCH part of the j -th substage a vertex R is inserted into $LIST(i)$ (where $i < j$). This can happen if a level i is defined for R . Clearly this level is illegal, and thus by Theorem 1.1 this level cannot be defined by a 1-link. Assume that the level i is defined for R by a 2-link XY . Since no changes of the set $T(R)$ occur during the SEARCH part of the algorithm, it follows by Theorem 1.4 that $X \in T(R)$. However, in this case, by step 6 of the algorithm, the assignment $LINK(R) \leftarrow XY$ cannot be performed, and thus the level i is not defined for R and R is not inserted into $LIST(i)$.

Thus, if $i < j$, no changes of $LIST(i)$ occur during the SEARCH part of the j -th substage, and at the end of this part $LIST(i)$ contains exactly all vertices of minimum legal level i .

Consider now $LIST(j)$: Corollary 1.2 implies that at the end of SEARCH part of the j -th substage, all the vertices which are contained in $LIST(j)$ are vertices of minimum legal level j . On the other hand, if a vertex R has a minimum legal level j , then by Theorem 1.5,

at the end of the SEARCH part of the j -th substage $\ell(R)=j$ and thus $R \in \text{LIST}(j)$. Therefore, at the end of the SEARCH part of the j -th substage $\text{LIST}(j)$ contains exactly all vertices of minimum legal level j .

Q.E.D.

Theorem 1.6

At the end of the j -th substage, for each $i \leq j$, $\text{LIST}(i)$ contains exactly all vertices of minimum legal level i .

Proof. By Corollary 1.3, at the end of the SEARCH part of the j -th substage, for each $i \leq j$, $\text{LIST}(i)$ contains exactly all vertices of minimum legal level i . Throughout the UPDATE part, changes of $\text{LIST}(i)$ can occur only if during an activation of $\text{MERGE}(R_2, R)$ [or $\text{MERGE}(R_3, \text{MATE}(R))$], the level of R_2 [or R_3] is changed as the result of the operation $T(R_2) \leftarrow T(R_2) \cup T(R)$ [or $T(R_3) \leftarrow T(R_3) \cup T(\text{MATE}(R))$]. By the definition of MERGE, in this case R_2 [or R_3] has a 2-link and $T(R_2) \neq \emptyset$ [$T(R_3) \neq \emptyset$]. However, if $i < j$, then for each vertex R in $\text{LIST}(i)$ such that R has a 2-link, $T(R) = \emptyset$. It follows that changes of $\text{LIST}(i)$ ($i < j$) can happen during the UPDATE part of the j -th substage only if a vertex of higher level is inserted into $\text{LIST}(i)$ or if a vertex is deleted from $\text{LIST}(j)$.

Let $i < j$, and assume that during the UPDATE part of the j -th substage a new vertex R_2 is inserted into $\text{LIST}(i)$ by $\text{MERGE}(R_2, R)$ such that $\text{LINK}(R_2) = \text{ST}$. Before MERGE is called,

belongs to the set $T(R)$ where $\ell(R)=j$, and thus ST can be assigned as $LINK(R)$. However, by the definition of R_2 , if ST defines for R_2 a level i , then it defines for R a level less than i , which is (by Corollary 1.3) illegal. Therefore, by Theorem 1.4, at the end of the SEARCH part of the j -th substage $SET(R)$, and thus after the operation $T(R_2) \leftarrow T(R_2) \cup T(R)$ is performed, $SET(R_2)$. It follows that ST fails the legality test which is performed on it during the performance of $MERGE(R_2, R)$, and thus it cannot be assigned as $LINK(R_2)$. Therefore, no vertex R_2 is inserted into $LIST(i)$ by $MERGE(R_2, R)$.

Assume that a vertex R_3 is inserted into $LIST(i)$ by $MERGE(R_3, MATE(R))$. By the definition, $T(MATE(R)) = \{R\}$, and thus the level i is defined for R_3 by a bridge of the form ER . By Eq. (1.3), $i = \ell(R_3) = \ell(R) + \ell(E) + 1 - \ell(MATE(R_3))$. When the bridge ER is searched, $MATE(R)$ already has a 1-link (for otherwise we would assign $LINK(MATE(R)) \leftarrow E$ and ER would not be a bridge), and thus $\ell(E) + 1 \geq \ell(MATE(R))$. Therefore $i = \ell(R_3) \geq \ell(R) + \ell(MATE(R)) - \ell(MATE(R_3))$. On the other hand, by step 11 of the algorithm, $LINK(R) = ST$ where $SET(R_3)$. Thus (by Eq. (1.3)): $\ell(R) + \ell(MATE(R)) = \ell(ST) + 1$. It follows that $i \geq \ell(ST) + 1 - \ell(MATE(R_3))$. Since at the end of the SEARCH part $T \in T(R)$, it follows that at the end of the SEARCH part of the j -th substage, the bridge TS belongs to the set $\{S'T' \mid S'T' \in CHAIN(T'), T' \in T(R_3), S' \notin T(R_3)\}$ and it can define for R_3 a level not greater than i . Therefore by Claim 1 of Theorem 1.5, at the end of the SEARCH part $\ell(R_3) \leq i$ — in contradiction to our assumption on R_3 . Therefore, no vertex R_3 is inserted into

LIST(i) by $MERGE(R_3, MATE(R))$.

Hence, no vertex is inserted into LIST(i) during the UPDATE part of the j-th substage.

Assume that a vertex R_2 is deleted from LIST(j) during the performance of $MERGE(R_2, R)$ in the UPDATE part. Let $ST = LINK(R_2)$ when MERGE is called. Thus (by Corollary 1.0 in Section C) $T \in T(R_2)$. Since after the performance of $T(R_2) \leftarrow T(R_2) \cup T(R)$ ST fails the legality test, when MERGE is called $SET(R)$. Therefore, at the end of the SEARCH part TS can define for R a legal level ℓ_1 , and since $\ell(R) = j$, $\ell_1 \geq j$. On the other hand, by the definition of MERGE, $LINK(MATE(R)) \in T(R_2)$ and thus (by case (b) of Theorem 1.0 in Section C), $p(0 \rightarrow MATE(R_2))$ is a head of $p(0 \rightarrow LINK(MATE(R)))$. It follows that $p(0 \rightarrow MATE(R_2))$ is a head of $p(0 \rightarrow MATE(R))$ and thus (since $R_2 \neq R$), $\ell(MATE(R)) > \ell(MATE(R_2))$. Therefore (by Eq. (1.3)):

$$\ell(R_2) = \ell(ST) + 1 - \ell(MATE(R_2)) > \ell(ST) + 1 - \ell(MATE(R)) = \ell_1 \geq j.$$

This inequality contradicts the fact that $\ell(R_2) = j$. Hence, no vertex can be deleted from LIST(j) during the performance of $MERGE(R_2, R)$.

Assume that a vertex R_3 is deleted from LIST(j) during the performance of $MERGE(R_3, MATE(R))$. Let $ST = LINK(R_3)$ when $MERGE(R_3, MATE(R))$ is called. Since after the performance of $T(R_3) \leftarrow T(R_3) \cup T(MATE(R))$, ST fails the legality test, when MERGE is called $SET(MATE(R))$. However, when MERGE is called, $T(MATE(R)) = \{R\}$, and thus $S = R$. Therefore, the assignment $LINK(R_3) = ST$ defines for R_3 a level greater than $\ell(R)$, i.e.,

greater than j . This contradicts the fact that $\ell(R_3)=j$. Hence no vertex can be deleted from $LIST(j)$ during the performance of $MERGE(R_3, MATE(R))$.

Therefore no change can be performed on $LIST(j)$ during the UPDATE part of the j -th substage and thus, at the end of the j -th substage $LIST(j)$ contains exactly all vertices of minimum legal level j .

Q.E.D.

Theorem 1.6 proves the inductive hypothesis for j , and thus the proof of the Main Theorem is completed.

Corollary 1.4.

If the first stage is terminated in step 9(a) of the algorithm, then all the vertices whose present levels are greater than r , are not lying on any minimum legal augmenting path in the graph, and therefore can be ignored throughout the remainder of the present phase (i.e., during the second, third and fourth stages).

Proof. Let R be a vertex such that when the first stage is terminated $\ell(R) > r$. If R is lying on a minimum legal augmenting path $p(A \rightarrow B)$ then it is either f -lying on $p(A \rightarrow B)$ or f -lying on $p(A \leftarrow B)$. Assume that R is f -lying on $p(A \rightarrow B)$ [$p(A \leftarrow B)$], then the path $p(A \rightarrow R)$ [$p(R \leftarrow B)$] is a legal alternating path leading to R whose length is not greater than r . Thus, by Theorem 1.5, at the end of the SEARCH part of the r -th substage, $\ell(R) \leq r$. This is a

contradiction to the fact that $\ell(R) > r$. Hence R is not lying on any minimum legal augmenting path in the graph, and therefore can be ignored throughout the remainder of the present phase.

Q.E.D.

Corollary 1.5.

If the first stage is terminated in step 14 of the algorithm, then the present matching is maximum.

Proof. If the present matching is not maximum, then there must exist in the graph a minimum legal augmenting path $p(A \rightarrow B)$. Clearly the length r of this path can not exceed $\lfloor \frac{n}{2} \rfloor$. Therefore, by Theorem 1.5, at the end of the SEARCH part of the r -th substage $\ell(B) = r$, and thus the algorithm would be terminated in step 9(a). This contradiction proves that there is no legal augmenting path in the graph, and thus the present matching is maximum.

Q.E.D.

H. The Complexity of the First Stage and the Implementation of MERGE

We consider separately the complexities of the SEARCH part and of the UPDATE part in the algorithm.

Throughout the whole performance of the first stage, each free edge of the graph is searched (during the SEARCH part) at most once in each direction. A constant number of operations are performed per

each such a search. Thus, the complexity of the SEARCH part is at most $O(m)$ (where m is the number of edges in the graph).

During the first stage, subroutine MERGE is called exactly twice per each vertex whose 2-link is declared to be well-defined. Since there are at most $\lfloor \frac{n}{2} \rfloor$ vertices which have a 2-link, subroutine MERGE is called at most n times throughout the whole first stage.

In this section we show three implementations of subroutine MERGE : The first implementation leads to a total complexity of $O(n^2)$ for the first stage. The second implementation leads to a complexity of $O(m \lg n)$ for the first stage. The third implementation leads to a complexity of $O(m \lg \lg n)$ for the first stage. Another implementation, which leads to a complexity of $O(m+n^{1+\epsilon})$ (where ϵ is an arbitrary positive value), is described in Appendix B. Since the complexity of the first stage determines the complexity of the whole phase, those implementations lead to matching algorithms of complexities $O(n^{2.5})$, $O(m\sqrt{n} \lg n)$, $O(m\sqrt{n} \lg \lg n)$ and $O(m\sqrt{n}+n^{1.5+\epsilon})$ respectively.

The first two implementations are simpler than the other two. While the first is efficient in the case of dense graphs, the second implementation is most efficient in the case of sparse graphs (e.g. planar graphs).

Implementation 1. In this implementation each set $T(R)$ is represented by a simple list. Two variables are used to describe

this list: $FRST(R)$ points to the first vertex in the list $T(R)$ (if $T(R)=\emptyset$ then $FRST(R)=0$); $NEXT(V)$ (where $V \in T(R)$) points to the vertex which follows V in $T(R)$ (if V is the last vertex in $T(R)$, then $NEXT(V)=0$). We also use in this implementation the variable $TAIL$ which is attached to each vertex V of the graph and points to the set $T(R)$ which contains V : $TAIL(V)=R \Leftrightarrow V \in T(R)$. In the beginning of the first stage we assign for each vertex R of the graph: $TAIL(R) \leftarrow MATE(R)$, $FRST(R) \leftarrow MATE(R)$, $NEXT(R) \leftarrow 0$. The variables $TAIL$, $FRST$ and $NEXT$ are changed only when the sets $T(R')$ are changed, i.e., during the performance of subroutine $MERGE$.

The algorithm of the subroutine $MERGE(R_2, R)$ goes as follows:

[Merge $T(R_2)$ and $T(R)$]

1. For each vertex V of $T(R)$ assign: $TAIL(V) \leftarrow R_2$.
2. Let V_1 be the last vertex of $T(R_2)$, then assign:
 $NEXT(V_1) \leftarrow FRST(R)$.
3. Assign $FRST(R) \leftarrow 0$.
4. Delete R_2 from $LIST(\mathcal{L}(R_2))$. Assign: $\mathcal{L}(R_2) \leftarrow \lfloor \frac{n}{2} \rfloor + 1$,
 $LINK(R_2) \leftarrow 0$. Let $T \leftarrow FRST(R_2)$.

[Search through the vertices of $T(R_2)$]

5. If $CHAIN(T) = \emptyset$ then go to 8. Else let ST be the first bridge of $CHAIN(T)$.

6. If $\text{TAIL}(S)=R_2$ then delete ST from $\text{CHAIN}(T)$ and return to 5.
7. If $\ell(R_2) > \ell(ST) + 1 - \ell(\text{MATE}(R_2))$, then assign:
 $\ell(R_2) \leftarrow \ell(ST) + 1 - \ell(\text{MATE}(R_2))$, $\text{LINK}(R_2) \leftarrow ST$.
8. If $\text{NEXT}(T) \neq 0$ then assign: $T \leftarrow \text{NEXT}(T)$ and return to 5.
9. Insert R_2 into $\text{LIST}(\ell(R_2))$.

One can see that this algorithm implements subroutine MERGE as it is defined by rules (a) and (b') in Section E.

By Theorem 1.4, the bridges which are removed from the CHAINS in step 6 of the algorithm are no more useful and their deletion is safe.

Except for the deletion of bridges in steps 5-6, a constant number of operations is performed per each vertex of $T(R_2) \cup T(R)$ in every call of MERGE. Thus, except for steps 5-6, the complexity of each performance of MERGE is $O(n)$ and the total complexity of activating subroutine MERGE during the first stage is therefore $O(n^2)$. Since the number of bridges is less than m (the number of edges in the graph), steps 5-6 can be performed not more than m times throughout the whole first stage. Thus, the complexity of steps 5-6 is $O(m)$.

Therefore, the total complexity of the first stage in the case of Implementation 1 is $O(n^2)$.

Implementation 2. In Implementation 2, each set $T(R)$ is represented by a 2-3 tree [6]. Since the number of vertices in a set $T(R)$ is at most n , the depth of the tree cannot exceed $\lg n$.

For each vertex V of the graph we attach a variable called $VALUE$ which is defined as follows: If V is not well-defined yet, or if $CHAIN(V)$ is empty, then $VALUE(V)=n$; else $VALUE(V)=\ell(UV)$, where UV is the first bridge in $CHAIN(V)$. Thus, in the beginning of the first stage we assign for each vertex V : $VALUE(V)=n$. $VALUE(V)$ is changed only in one of the following three cases: When $CHAIN(V)=\emptyset$ and a well-defined bridge is inserted into $CHAIN(V)$ (step 7 of the algorithm); when $CHAIN(V)\neq\emptyset$ and V has a 2-link which is declared to be well-defined (step 11 of the algorithm); when the first bridge of $CHAIN(V)$ is deleted from the $CHAIN$ (during the performance of subroutine $MERGE$). Clearly the operations which may be required in steps 7 and 11 of the algorithm have no influence on its complexity.

The algorithm of the subroutine $MERGE(R_2, R)$ in this implementation, goes as follows (the underlined instructions are the instructions on 2-3 tree as defined in [6]):

1. MERGE $T(R_2) \leftarrow T(R_2) \cup T(R)$.
2. Delete R_2 from $LIST(\ell(R_2))$.
3. Find in $T(R_2)$ a vertex T of MINIMUM $VALUE$.
4. If $VALUE(T)=n$ then assign: $\ell(R_2) \leftarrow \lfloor \frac{n}{2} \rfloor + 1$, $LINK(R_2) \leftarrow 0$ and

go to 7. Else, let ST be the first bridge in $CHAIN(T)$.

5. FIND the vertex R_1 such that $SET(R_1)$. If $R_1 = R_2$ then DELETE T from $T(R_2)$. Delete ST from $CHAIN(T)$ and update $VALUE(T)$. INSERT T into $T(R_2)$. Return to 3.
6. [$SET(R_2)$.] Assign: $l(R_2) \leftarrow l(ST) + 1 - l(MATE(R_2))$, $LINK(R_2) \leftarrow ST$.
7. Insert R_2 into $LIST(l(R_2))$.

Clearly the implementation of MERGE by this algorithm is in accordance with the definition of MERGE.

Throughout the whole activation of subroutine MERGE in the first stage, each of the instructions MINIMUM (step 3), FIND, DELETE and INSERT (step 5) is required at most $m+n$ times (once per each deletion of a bridge ST from $CHAIN(T)$ and once for each call of MERGE). The instruction MERGE (step 1) is required at most n times (once per each call of subroutine MERGE). Since the complexity of each of those instructions is $O(\lg n)$, the total complexity of this algorithm is $O(m \lg n)$.

Thus, Implementation 2 determines for the first stage complexity $O(m \lg n)$.

Implementation 3. Implementation 3 is in fact similar to Implementation 2, except that the representation of the sets $T(R)$ by 2-3 trees is replaced by the data structure which was suggested by van Emde Boas [7]. According to his algorithm, each of the

instructions MINIMUM , FIND , DELETE , INSERT and MERGE has complexity $O(\lg \lg n)$. Therefore the total complexity of the first stage in this implementation is $O(m \lg \lg n)$.

2. THE SECOND STAGE

A. General Description

The final goal of every phase of the algorithm is to find a maximal set of disjoint minimum legal augmenting paths in the graph. In the second stage of the phase, we reduce the original graph by removing those edges which are not lying on any minimum legal augmenting paths, and by shrinking some of the edges which are lying on illegal augmenting paths. The result of this process is a reduced graph which has the following property:

Every maximal set of disjoint arbitrary legal augmenting paths in the reduced graph corresponds to a maximal set of disjoint minimum legal augmenting paths in the original graph.

Thus, instead of looking for a maximal set of disjoint minimum augmenting paths in the original graph, it is sufficient to look for a maximal set of disjoint arbitrary (legal) augmenting path in the reduced graph. Clearly, the renouncement of the minimality requirement makes the search of a maximal set of disjoint paths, simpler.

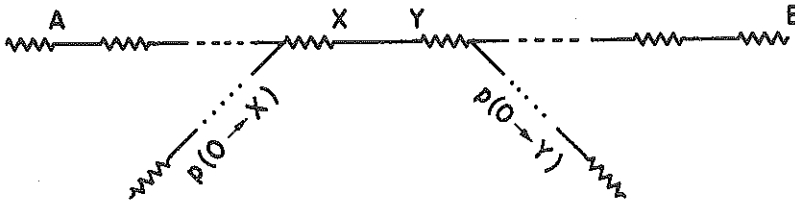
The irrelevant vertices and edges. In the beginning of the second stage we know the length r of the minimum legal augmenting paths in the graph (step 9(a) of the algorithm of the first stage). For each i , $i \leq r$, we also know the set $LIST(i)$ of all the vertices whose levels are i , and by using the variables $LINK$ we can retrace minimum legal a_1 -paths leading to these vertices (Main

Theorem of Chapter 1). The other vertices (whose levels are greater than r), cannot be lying on any minimum legal augmenting path in the graph, and therefore can be ignored throughout the remainder of the present phase (Corollary 1.4 of Chapter 1).

Let XY be a free [matched] edge, and let $p(A \rightarrow B)$ be a legal augmenting path which is passing through XY (such that X is f-lying on $p(A \rightarrow B)$). The augmenting path $p(A \rightarrow B)$ consists in fact of two legal alternating paths, $p(A \rightarrow X)$ which is leading to X and $p(Y \leftarrow B)$ which is leading to Y (see Fig. 2.1). Since by the definition $\ell(X) \leq |p(A \rightarrow X)|^*$ and $\ell(Y) \leq |p(Y \leftarrow B)|$, it follows that $\ell(X) + \ell(Y) + 1 \leq |p(A \rightarrow B)|$ [or, if XY is matched, $\ell(X) + \ell(Y) \leq |p(A \rightarrow B)| - 1$].

Define:

— free edge
 ~~~~ matched edge



$$\ell(X) = |p(O \rightarrow X)| \leq |p(A \rightarrow X)|$$

$$\ell(Y) = |p(O \rightarrow Y)| \leq |p(Y \leftarrow B)|$$

---


$$d(XY) = \ell(X) + \ell(Y) + 1 \leq |p(A \rightarrow B)|$$

Fig. 2.1 -  $d(XY)$

\*  $|p(A \rightarrow X)|$  is the length of the alternating path  $p(A \rightarrow X)$ .

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

Then, the length of a legal augmenting path which is passing through  $XY$  must be at least  $d(XY)$ . It follows that if  $d(XY) > r$  then  $XY$  is not lying on any minimum legal augmenting path in the graph and therefore can be ignored throughout the remainder of the present phase. In particular, if  $XY$  is a matched edge, then neither  $X$  nor  $Y$  can be lying on a minimum legal augmenting path and thus these vertices can be ignored too.

Small loop. Let  $XY$  be an edge (free or matched), such that  $d(XY) < r$ . Since the length of the augmenting path  $p(O \rightarrow X) \cdot p(O \leftarrow Y)$  is  $d(XY)$  (i.e., less than  $r$ ), this path cannot be legal, but must contain an odd loop (see Fig. 2.2). Such a loop, which is formed by an augmenting path whose length is less than  $r$ , is called a small loop. An edge  $XY$  is lying on a small loop if and only if  $d(XY) < r$ . A vertex  $Y$  is lying on a small loop if and only if the matched edge  $YZ$  (where  $Z = \text{MATE}(Y)$ ) is lying on a small loop, i.e., if and only if  $d(YZ) < r$ . However,  $Y$  can simultaneously be lying on some different small loops (e.g., in Fig. 2.2,  $Y$  is lying on the two small loops which are formed by the edges  $XY$  and  $ZU$ ).

Let  $Y$  be a vertex and let  $N$  be the last vertex on  $p(O \rightarrow Y)$  such that  $N$  is not lying on a small loop. Then  $N$  is  $f$ -lying on  $p(O \rightarrow Y)$  and the matched edge  $MN$  (where  $M = \text{MATE}(N)$ ) satisfies  $d(MN) \geq r$ . In Section B we define  $N$  as BASE(Y), and we prove that



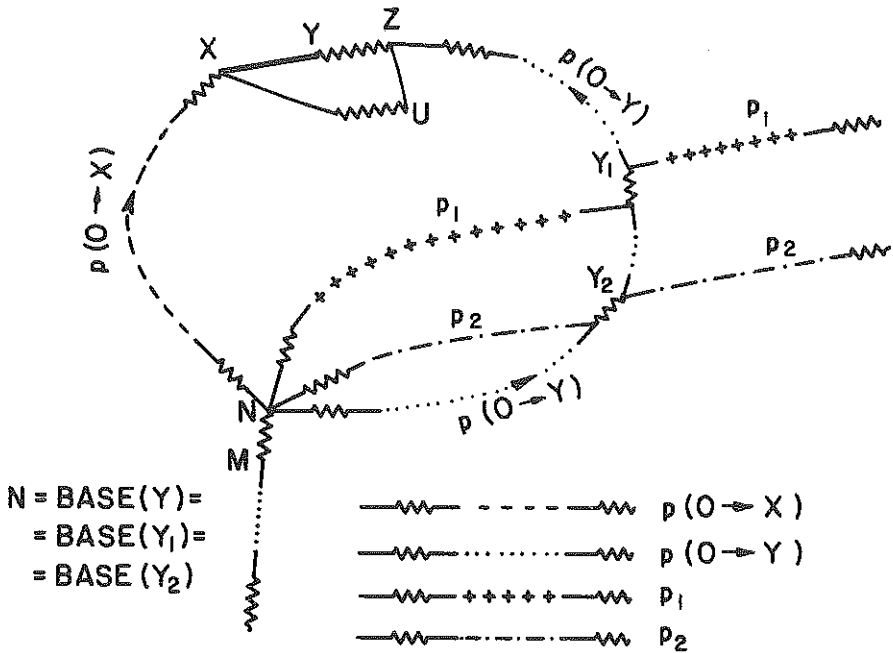


Fig. 2.2 -  $d(XY) < r$

if a minimum legal augmenting path is passing through  $Y$  then it must also pass through  $N$ . Thus, if two vertices  $Y_1$  and  $Y_2$  have the same  $\text{BASE}$ , then there exist no two disjoint minimum legal augmenting paths  $p_1$  and  $p_2$ , such that  $p_1$  is passing through  $Y_1$  and  $p_2$  is passing through  $Y_2$ . Therefore, for the purpose of finding a maximal set of disjoint minimum legal augmenting path in the graph, we may think of  $Y_1$  and  $Y_2$  as if they are identical with their  $\text{BASE}$ .

Thus, the basic idea of the second stage is to remove from the graph all the vertices and edges which are not lying on a minimum

legal augmenting path (i.e., all edges  $XY$  such that  $d(XY) > r$ ), and for every vertex  $Y$  which is lying on a small loop to identify  $Y$  with its BASE. The self-loops which are created by this process are removed from the graph, and if parallel edges are created, all of them but one are deleted. The graph  $\tilde{G}$  which is formed as the result of this process is called the reduced graph. In Section C we prove that  $\tilde{G}$  has the property that every maximal set of (arbitrary) disjoint legal augmenting paths in  $\tilde{G}$  corresponds to a maximal set of disjoint minimum legal augmenting paths in the original graph  $G$ .

[Notice that the process of identifying every vertex  $Y$  which is lying on a small loop with its BASE is in fact a process of shrinking the small loops, and is therefore similar to the process of blossoms' shrinking which is performed by Edmonds [1]. However, there are two principal distinctions between these two processes: While Edmonds shrinks every loop when it is discovered, we shrink only those loops which cause troubles, namely the small loops. Also, in Edmonds' algorithm the shrinking is performed immediately when the loop is discovered and separately for each loop, and thus a vertex which is lying on some loops may participate in many shrinks. In our algorithm each vertex and each edge is shrunk only once and therefore our bookkeeping is simpler and the complexity of the algorithm is lower].

## B. The BASE of a Vertex

Let  $Y$  be a vertex and let  $Z=MATE(Y)$ . Define:

$$BASE(Y) = \begin{cases} Y & \text{if } d(YZ) \geq r \\ BASE(V) & \text{if } d(YZ) < r \text{ and } LINK(Y)=V \\ BASE(S) & \text{if } d(YZ) < r \text{ and } LINK(Y)=ST \end{cases} \quad (2.2)$$

The following Lemma 2.1 also proves that for each  $Y$ ,  $BASE(Y)$  is properly defined.

### Lemma 2.1

Let  $N=BASE(Y)$  and  $M=MATE(N)$ . Then  $N$  is the last vertex on  $p(0 \rightarrow Y)$  such that  $d(MN) \geq r$ .

Proof. Let  $Z=MATE(Y)$ . The lemma is trivial if  $d(YZ) \geq r$ . Thus, assume that  $d(YZ) < r$ . Let  $A_1$  be the 1-exposed vertex from which  $p(0 \rightarrow Y)$  starts, and let  $A_2$  be the 2-exposed vertex such that  $A_2=MATE(A_1)$ . By the definition  $\ell(A_1)=0$  while by the algorithm of the first stage  $\ell(A_2) \geq r$ . Thus,  $A_2 A_1$  is a matched edge on  $p(0 \rightarrow Y)$  such that  $d(A_2 A_1) \geq r$ . Let  $M_1 N_1$  be the last matched edge on  $p(0 \rightarrow Y)$  such that  $d(M_1 N_1) \geq r$ , and  $N_1$  is f-lying on  $p(0 \rightarrow Y)$ . We have to show that  $N_1=BASE(Y)$ .

Consider the sequence of vertices  $Y_0, Y_1, \dots$  which is defined as follows:

$$Y_0 = Y ; \quad Y_i = \begin{cases} V & \text{if LINK}(Y_{i-1})=V \\ S & \text{if LINK}(Y_{i-1})=ST \end{cases} \quad i \geq 1$$

Clearly, for each  $i \geq 1$ ,  $p(0 \rightarrow Y_i)$  is a head of  $p(0 \rightarrow Y_{i-1})$  and thus a head of  $p(0 \rightarrow Y)$ .

Let  $j$  be the greatest index such that  $N_1$  is lying (and thus f-lying) on  $p(0 \rightarrow Y_j)$ . Assume that  $Y_j \neq N_1$ . Then  $Y_j$  cannot have a 1-link (for otherwise  $\text{LINK}(Y_j)$  would contradict the choice of  $Y_j$ ). Thus,  $Y_j$  has a 2-link. Let  $ST = \text{LINK}(Y_j)$ , then  $S = Y_{j+1}$ . By the choice of  $Y_j$ ,  $N_1$  is not lying on  $p(0 \rightarrow S)$ , but it is lying on  $p(T \rightarrow Y_j)$ . Denote by  $d_1$  the number of free edges on  $p(S \rightarrow Y_j)$  between  $S$  and  $N_1$  and by  $d_2$  the number of free edges on the segment  $p(M_1 \rightarrow Y_j)$ . Since  $p(0 \rightarrow S)$  is a head of  $p(0 \rightarrow Y_j)$ ,  $\ell(Y_j) = \ell(S) + d_1 + d_2$ . For the same reason:  $\ell(N_1) \leq \ell(S) + d_1$ . Since  $\text{LINK}(Y_j) = ST$ ,  $p(0 \rightarrow \text{MATE}(Y_j))$  is a head of  $p(0 \rightarrow T)$  and thus  $\ell(M_1) \leq \ell(\text{MATE}(Y_j)) + d_2$ . Therefore:

$$\begin{aligned} r \leq d(M_1 N_1) &= \ell(M_1) + \ell(N_1) \leq \ell(S) + d_1 + d_2 + \ell(\text{MATE}(Y_j)) \\ &= \ell(Y_j) + \ell(\text{MATE}(Y_j)) \end{aligned}$$

in contradiction to the choice of  $M_1 N_1$ . Hence  $Y_j = N_1$ .

By the choice of  $N_1$  and  $Y_j$ , for each  $i$ ,  $0 \leq i \leq j-1$ , there exists:  $\text{BASE}(Y_i) = \text{BASE}(Y_{i+1})$ . Therefore,  $\text{BASE}(Y) = \text{BASE}(Y_j) = \text{BASE}(N_1)$ . However, by the definition,  $\text{BASE}(N_1) = N_1$  and thus  $\text{BASE}(Y) = N_1$ .

Q.E.D.

Corollary 2.1

Let  $N = \text{BASE}(Y)$  and  $M = \text{MATE}(N)$ . Then  $p(O \rightarrow N)$  is the longest head of  $p(O \rightarrow Y)$  such that  $d(MN) \geq r$ .

Lemma 2.2

Let  $YZ$  be a matched edge such that  $d(YZ) < r$ . Then for each edge  $ST$  on  $p(\text{BASE}(Y) \rightarrow Y)$ ,  $d(ST) < r$ .

Proof. By Lemma 2.1, the claim is true in case  $ST$  is a matched edge. Thus, let  $ST$  be a free edge, and without loss of generality assume that  $S$  is  $f$ -lying on  $p(O \rightarrow Y)$ .

Consider the sequence of vertices  $Y_0, Y_1, \dots, Y_k$  which is defined as follows:

$$Y_0 = Y ; \quad Y_k = \text{BASE}(Y) ; \quad Y_i = \begin{cases} V & \text{if } \text{LINK}(Y_{i-1}) = V \\ U & \text{if } \text{LINK}(Y_{i-1}) = UW \end{cases} \quad 1 \leq i \leq k$$

Clearly for each  $i$ ,  $1 \leq i \leq k$ ,  $p(O \rightarrow Y_i)$  is a head of  $p(O \rightarrow Y_{i-1})$ , and thus a head of  $p(O \rightarrow Y)$ .

For each  $i$ ,  $0 \leq i \leq k$ , let  $Z_i = \text{MATE}(Y_i)$ . Let  $j$  be the least index such that  $ST$  is lying on  $p(Y_j \rightarrow Y)$ . If  $Y_j = \text{LINK}(Y_{j-1})$  then  $\ell(Y_{j-1}) = \ell(Y_j) + 1$ . On the other hand, in this case the choice of  $Y_j$  implies that  $Y_j = S$ ,  $Z_{j-1} = T$ . Thus, by Lemma 2.1:

$$\begin{aligned} d(ST) &= \ell(S) + \ell(T) + 1 = \ell(Y_j) + \ell(Z_{j-1}) + 1 \\ &= \ell(Y_{j-1}) + \ell(Z_{j-1}) = d(Y_{j-1}Z_{j-1}) < r . \end{aligned}$$

If  $Y_j \neq \text{LINK}(Y_{j-1})$ , then  $\text{LINK}(Y_{j-1}) = Y_j W$ . Denote by  $d_1$  the number of free edges on  $p(Y_j \rightarrow Y_{j-1})$  between  $Y_j$  and  $S$ , and by  $d_2$  the number of free edges on the segment  $p(T \rightarrow Y_{j-1})$ . Since  $p(O \rightarrow Y_j)$  is a head of  $p(O \rightarrow Y_{j-1})$ ,  $\ell(Y_{j-1}) = \ell(Y_j) + d_1 + d_2 + 1$ . For the same reason:  $\ell(S) \leq \ell(Y_j) + d_1$ . Since  $\text{LINK}(Y_{j-1}) = Y_j W$ ,  $p(O \rightarrow Z_{j-1})$  is a head of  $p(O \rightarrow W)$  and thus:  $\ell(T) \leq \ell(Z_{j-1}) + d_2$ . Therefore:

$$\begin{aligned} d(ST) &= \ell(S) + \ell(T) + 1 \leq \ell(Y_j) + \ell(Z_{j-1}) + d_1 + d_2 + 1 \\ &= \ell(Y_{j-1}) + \ell(Z_{j-1}) = d(Y_{j-1} Z_{j-1}) < r. \end{aligned}$$

Q.E.D.

### Corollary 2.2

Let  $YZ$  be a matched edge such that  $d(YZ) < r$ . Let  $N = \text{BASE}(Y)$  and  $M = \text{MATE}(N)$ . Then  $N$  has a 1-link and  $\ell(M) > \ell(N)$ .

Proof. Let  $N = \text{BASE}(Y)$ ,  $M = \text{MATE}(N)$  and  $P$  be the vertex which follows  $N$  on  $p(N \rightarrow Y)$ . By Lemma 2.1:  $r \leq d(MN) = \ell(M) + \ell(N)$ . By Lemma 2.2:  $d(PN) = \ell(P) + \ell(N) + 1 < r$ . Thus,  $\ell(P) + 1 < \ell(M)$ . If  $\ell(N) \geq \ell(M)$ , then when the edge  $PN$  is searched (during substage  $\ell(P) + 1$  of the first stage), neither  $M$  nor  $N$  are already well-defined. In this case, by the algorithm of the first stage (step 4) we assign:  $\text{LINK}(M) \leftarrow P$ ,  $\ell(M) \leftarrow \ell(P) + 1$ , in contradiction to  $\ell(M) > \ell(P) + 1$ . Thus,  $\ell(M) > \ell(N)$  and  $N$  must have a 1-link.

Q.E.D.

Lemma 2.3

Let  $ST$  be a bridge such that  $d(ST) < r$ . Let  $N = \text{BASE}(T)$  and  $M = \text{MATE}(N)$ . If  $\ell(T) < \ell(M)$  then  $p(O \rightarrow N)$  is a head of  $p(O \rightarrow S)$ .

Proof. Let  $\ell_1 = \ell(S) + \ell(T) + 1 - \ell(N)$ . Since by Corollary 2.1,  $p(O \rightarrow N)$  is a head of  $p(O \rightarrow T)$ ,  $\ell(T) \geq \ell(N)$ . Thus,  $\ell_1 \geq \ell(S) + 1$ . On the other hand, by Lemma 2.1  $d(MN) \geq r$ , and thus:

$$\ell_1 = d(ST) - \ell(N) < r - \ell(N) \leq d(MN) - \ell(N) = \ell(M).$$

Therefore,  $\ell(S) + 1 < \ell(M)$ , and in the first stage the bridge  $ST$  has been searched before the beginning of the  $\ell(M)$ -th substage.

Let  $W$  be the vertex such that in the beginning of the  $\ell(M)$ -th substage  $T \in T(W)$ . Since  $\ell(T) < \ell(M)$ ,  $T$  is already well-defined and thus  $W$  is the last vertex on  $p(O \rightarrow T)$  such that  $W$  is not yet well-defined. Hence,  $W$  is lying on  $p(M \rightarrow T)$ . Since both  $p(O \rightarrow N)$  and  $p(O \rightarrow \text{MATE}(W))$  are heads of  $p(O \rightarrow T)$ , it follows that  $p(O \rightarrow N)$  is a head of  $p(O \rightarrow \text{MATE}(W))$ . We show that in the beginning of the  $\ell(M)$ -th substage  $S \in T(W)$ . This relation implies that  $p(O \rightarrow \text{MATE}(W))$  is a head of  $p(O \rightarrow S)$  and thus  $p(O \rightarrow N)$  is a head of  $p(O \rightarrow S)$ .

If  $ST$  has failed the legality test before the  $\ell(M)$ -th substage then at that point both  $S$  and  $T$  were contained in the same set  $T(R')$  and thus, in the beginning of the  $\ell(M)$ -th substage  $S$  belongs to the same set as  $T$ , i.e.,  $S \in T(W)$ . If  $ST$  has not failed the legality test, then  $ST$  belongs to the set  $\{S'T' \mid S'T' \in \text{CHAIN}(T'), T' \in T(W)\}$ , and thus by Corollary 1.0  $ST$  can be

assigned as  $\text{LINK}(W)$ . Denote by  $d_1$  the number of free edges on the segment  $p(W \rightarrow T)$ , then the assignment  $\text{LINK}(W) = ST$  defines for  $W$  a level  $\ell_2$ , where  $\ell_2 = \ell(S) + 1 + d_1$ . However, since  $W$  is lying on  $p(M \rightarrow T)$ ,  $d_1 \leq \ell(T) - \ell(N)$  and thus  $\ell_2 \leq \ell_1 < \ell(M)$ . If the assignment  $\text{LINK}(W) = ST$  is legal, then by the Main Theorem of Chapter 1, in the beginning of the  $\ell(M)$ -th substage  $W$  must be already well-defined in contradiction to the choice of  $W$ . Thus, the assignment  $\text{LINK}(W) = ST$  is illegal, and therefore by Theorem 1.4,  $\text{SET}(W)$ .

Q.E.D.

#### Lemma 2.4

Let  $YZ$  be a matched edge such that  $d(YZ) < r$ . Let  $N = \text{BASE}(Y)$  and  $M = \text{MATE}(N)$ . Then  $\ell(Y) < \ell(M)$ .

Proof. Let  $Y$  be a vertex of least level for which the lemma does not hold, i.e.,  $\ell(Y) \geq \ell(M)$ . If  $\ell(Z) > \ell(Y)$ , then by Corollary 2.2,  $\ell(Z) \geq \ell(Y) \geq \ell(M) > \ell(N)$ , and thus  $d(MN) < d(YZ) < r$ , in contradiction to Lemma 2.1. Therefore  $\ell(Y) > \ell(Z)$ , i.e.,  $Y$  has a 2-link:  $\text{LINK}(Y) = TS$ .

By the definition  $\text{BASE}(Y) = \text{BASE}(T)$  and thus (Corollary 2.1),  $p(O \rightarrow N)$  is a head of  $p(O \rightarrow T)$ . If  $T = N$  then by Corollary 2.2  $\ell(T) < \ell(M)$ . If  $T \neq N$  then by Lemma 2.1,  $\ell(T) + \ell(\text{MATE}(T)) < r$  and thus, by our assumption on  $Y$ ,  $\ell(T) < \ell(M)$ . Therefore, in either case  $\ell(T) < \ell(M)$ .

By Lemma 2.2,  $d(ST) < r$ . Hence, Lemma 2.3 implies that  $p(O \rightarrow N)$ .



is a head of  $p(0 \rightarrow S)$ . However, since  $\text{LINK}(Y) = TS$ ,  $p(0 \rightarrow Z)$  too is a head of  $p(0 \rightarrow S)$ . Clearly,  $p(0 \rightarrow Z)$  cannot be a head of  $p(0 \rightarrow N)$  (for,  $p(0 \rightarrow N)$  is a head of  $p(0 \rightarrow Y)$ ) and thus  $p(0 \rightarrow N)$  is a head of  $p(0 \rightarrow Z)$ . Therefore,  $\ell(Z) \geq \ell(N)$ . Since we assume that  $\ell(Y) \geq \ell(M)$ , it follows that  $d(MN) \leq d(YZ) < r$ , in contradiction to Lemma 2.1. Hence  $\ell(Y) < \ell(M)$ .

Q.E.D.

Lemma 2.5

Let  $XY$  be an edge such that  $d(XY) < r$ . Then  $\text{BASE}(X) = \text{BASE}(Y)$ .

Proof. (a) If  $XY$  is a bridge: Let  $N_1 = \text{BASE}(X)$ ,  $M_1 = \text{MATE}(N_1)$ ,  $N_2 = \text{BASE}(Y)$ ,  $M_2 = \text{MATE}(N_2)$ , and  $Z = \text{MATE}(Y)$ . If  $d(YZ) \geq r$  then  $Y = N_2$ ,  $Z = M_2$  and there exists:  $\ell(X) + 1 = d(XY) - \ell(Y) < d(YZ) - \ell(Y) = \ell(Z)$ . If we assume that  $\ell(Z) \leq \ell(Y)$ , then, when the edge  $XY$  has been searched during substage  $\ell(X) + 1$  of the first stage, neither  $Y$  nor  $Z$  were well-defined, and thus, by the algorithm of the first stage (step 4) we would assign  $\text{LINK}(Z) \leftarrow X$ ,  $\ell(Z) \leftarrow \ell(X) + 1$  - in contradiction to the facts that  $YZ$  is a bridge and  $\ell(Z) > \ell(X) + 1$ . Therefore  $\ell(M_2) = \ell(Z) > \ell(Y)$ . If  $d(YZ) < r$ , then by Lemma 2.4  $\ell(M_2) > \ell(Y)$ . Thus, in either case,  $\ell(M_2) > \ell(Y)$ . Therefore, by Lemma 2.3  $p(0 \rightarrow N_2)$  is a head of  $p(0 \rightarrow X)$ . Since by Corollary 2.1  $p(0 \rightarrow N_1)$  is the longest head of  $p(0 \rightarrow X)$  such that  $d(N_1 M_1) \geq r$ , it follows that  $p(0 \rightarrow N_2)$  is a head of  $p(0 \rightarrow N_1)$ . However, by a similar argument  $p(0 \rightarrow N_1)$  is a head of  $p(0 \rightarrow N_2)$ . Hence  $N_1 = N_2$ .

Q.E.D.

(b) If  $XY$  is a matched edge: Without loss of generality assume that  $Y$  has a 2-link:  $LINK(Y)=ST$ . By the definition  $BASE(Y)=BASE(S)$ . By Eq. (1.3),  $d(ST)=\ell(S)+\ell(T)+1=\ell(X)+\ell(Y)=d(XY)<r$  and thus, by part (a) of this lemma,  $BASE(S)=BASE(T)$ . Therefore,  $BASE(Y)=BASE(T)$ . Since  $LINK(Y)=ST$ ,  $p(0 \rightarrow X)$  is a head of  $p(0 \rightarrow T)$ . On the other hand, since  $BASE(Y)=BASE(S)$ , it follows by Lemma 2.1 that for every matched edge  $UV$  on the segment  $p(Y \leftarrow T)$  there exists  $d(UV)<r$ . Therefore, by Lemma 2.1  $BASE(T)=BASE(X)$ , and thus  $BASE(Y)=BASE(X)$ .

Q.E.D.

(c) If  $XY$  is a free edge but not a bridge: Without loss of generality assume that  $LINK(MATE(Y))=X$ . Let  $Z=MATE(Y)$ , then  $\ell(Z)=\ell(X)+1$ . Therefore  $d(YZ)=\ell(Y)+\ell(Z)=\ell(X)+\ell(Y)+1=d(XY)<r$ , and thus by part (b) of this lemma,  $BASE(Y)=BASE(Z)$ . On the other hand, by the definition,  $BASE(Z)=BASE(X)$ . Therefore  $BASE(X)=BASE(Y)$ .

Q.E.D.

### Corollary 2.3

Let  $V$  be a vertex on  $p(BASE(Y) \rightarrow Y)$ . Then  $BASE(V)=BASE(Y)$ .

Proof. Let  $Z=MATE(Y)$ . The claim is trivial if  $d(YZ) \geq r$ . Assume that  $d(YZ) < r$  and let  $V$  be the vertex closest to  $Y$  on  $p(BASE(Y) \rightarrow Y)$  for which the lemma does not hold. Let  $W$  be the vertex which follows  $V$  on  $p(BASE(Y) \rightarrow Y)$ , then by our assumption  $BASE(W)=BASE(Y)$ . On the other hand, by Lemma 2.2,  $d(VW) < r$ , and thus by Lemma 2.5  $BASE(W)=BASE(V)$ . Therefore  $BASE(V)=BASE(Y)$ .

Q.E.D.

Theorem 2.1

Let  $p$  be a minimum legal augmenting path which is passing through a vertex  $V$ . Then  $p$  is also passing through  $\text{BASE}(V)$ , and for each vertex  $V'$  which is lying on  $p$  between  $V$  and  $\text{BASE}(V)$ , there exists:  $\text{BASE}(V') = \text{BASE}(V)$ .

Proof. Let  $p(A \rightarrow B)$  (where  $A$  is a 1-exposed vertex and  $B$  is a 2-exposed vertex) be a minimum legal augmenting path which is passing through  $V$ . Let  $N = \text{BASE}(V)$ . If  $N = V$  then the theorem is trivial. Therefore assume that  $N \neq V$ .

Let  $V_1 \xrightarrow{1} V_2$  denote the segment from  $V_1$  to  $V_2$  on the augmenting path  $p(A \rightarrow B)$ . Let  $Y$  be the first vertex on  $p(A \rightarrow B)$  such that for each vertex  $V'$  on  $Y \xrightarrow{1} V$  there exists:  $\text{BASE}(V') = N$ . We assume that  $Y \neq N$  [otherwise, throughout the proof of Theorem 2.1, one has to replace  $p(A \rightarrow B)$  by  $p(A \leftarrow B)$ . In this case  $Y$  is defined to be the first vertex on  $p(A \leftarrow B)$ , such that for each vertex  $V'$  on  $Y \xrightarrow{1} V$  there exists:  $\text{BASE}(V') = N$ . Clearly by this definition  $Y \neq N$ ]. Let  $Z = \text{MATE}(Y)$ . Since  $Y \neq N$ , by the definition  $d(YZ) < r$ , and thus, by Lemma 2.5,  $\text{BASE}(Z) = N$ . Let  $X$  be the vertex which precedes  $Y$  on  $p(A \rightarrow B)$ , then  $XY$  is a free edge [see Fig. 2.3], and since  $\text{BASE}(X) \neq N$ ,  $d(XY) \geq r$ . Let  $S$  be the first vertex on  $Y \xrightarrow{1} B$  such that  $\text{BASE}(S) \neq N$ , and let  $T$  be the vertex which precedes  $S$  on  $Y \xrightarrow{1} B$ . Thus,  $\text{BASE}(T) = N$ , and by Lemma 2.5  $d(ST) \geq r$ . Clearly  $V$  is lying on  $Y \xrightarrow{1} T$ . Also, for each vertex  $V'$  which is lying on  $Y \xrightarrow{1} T$  there exists:  $\text{BASE}(V') = N$ . We shall show that  $T = N$  and by this we shall prove the theorem.

Assume that  $T \neq N$  and let  $U = \text{MATE}(T)$ , then by the definition  $d(UT) < r$ , and by Lemma 2.5  $\text{BASE}(U) = N$ . Therefore  $U \neq S$ , and  $ST$  is a free edge [see Fig. 2.3].

Since the augmenting path  $p(A \rightarrow B)$  is legal and minimum, its length is  $r$ , and thus:  $|A \xrightarrow{1} X| + |X \xrightarrow{1} Y| + |Y \xrightarrow{1} B| = r$ . On the other hand,  $\ell(X) + \ell(Y) + 1 = d(XY) \geq r$ . Since, by the definition,  $\ell(X) \leq |A \xrightarrow{1} X|$ ,  $\ell(Y) \leq |Y \xrightarrow{1} B|$ , it follows that  $\ell(X) = |A \xrightarrow{1} X|$ ,  $\ell(Y) = |Y \xrightarrow{1} B|$ . By similar arguments,  $\ell(S) = |S \xrightarrow{1} B|$ ,  $\ell(T) = |A \xrightarrow{1} T|$ .

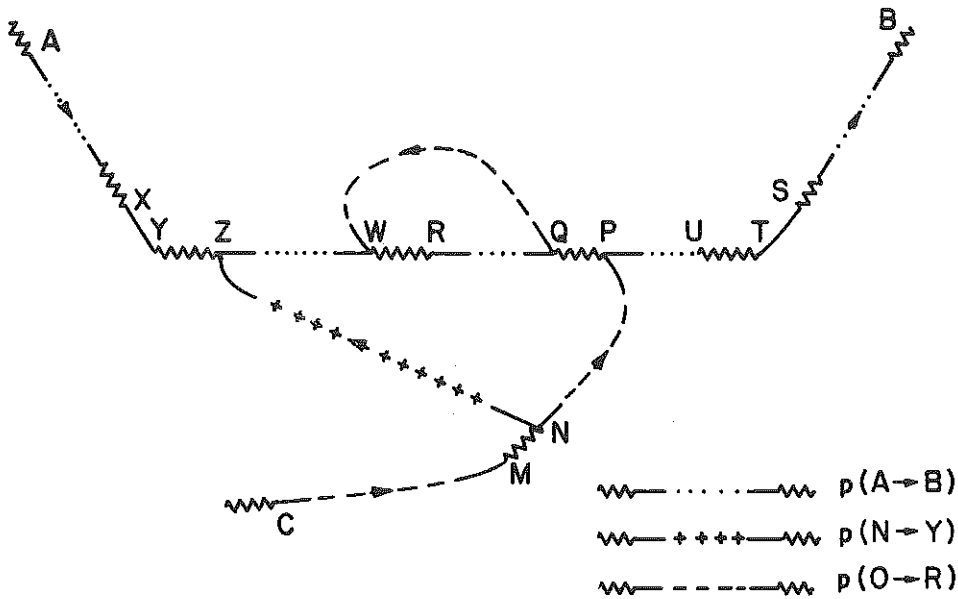


Fig. 2.3

Let  $W$  be the first vertex which is f-lying on  $T \xrightarrow{1} Y$  such that  $|B \xrightarrow{1} W| = \ell(W)$ . Let  $R = \text{MATE}(W)$ , and let  $d_1$  be the number of free edges on the segment  $T \xrightarrow{1} W$  (if  $W = U$  then  $d_1 = 0$ , else  $d_1 > 0$ ).

Then:  $\ell(W) = |B \xrightarrow{1} W| = |B \xrightarrow{1} S| + |S \xrightarrow{1} T| + |T \xrightarrow{1} W| = \ell(S) + 1 + d_1$ . On the other hand:  $\ell(W) + \ell(R) = d(WR) < r \leq d(ST) = \ell(S) + \ell(T) + 1$ . Therefore:  $\ell(R) + d_1 < \ell(T)$ . Clearly, this implies that  $R \neq T$  and  $U \neq W$ .

Let  $V_1 \xrightarrow{2} V_2$  denote the segment from  $V_1$  to  $V_2$  on the al-path  $p(O \rightarrow R)$ . Let  $C$  be the 1-exposed vertex from which  $p(O \rightarrow R)$  starts. If we assume that  $p(O \rightarrow R)$  and  $R \xrightarrow{1} T$  are disjoint (except for the vertex  $R$ ), then the alternating path  $C \xrightarrow{2} R \xrightarrow{1} T$  is a legal al-path of length  $\ell(R) + d_1$  (i.e., less than  $\ell(T)$ ) which is leading to  $T$  - in contradiction to the definition of  $\ell(T)$ . Thus  $p(O \rightarrow R)$  and  $R \xrightarrow{1} T$  are not disjoint. Let  $P$  be the last vertex which is f-lying on  $R \xrightarrow{1} T$  and is common to  $p(O \rightarrow R)$ , and let  $Q = \text{MATE}(P)$ . If we assume that  $Q$  is not f-lying on  $p(O \rightarrow R)$ , then  $P$  is f-lying on  $p(O \rightarrow R)$ , and  $|C \xrightarrow{2} P| < |C \xrightarrow{2} R|$ . Clearly  $|P \xrightarrow{1} T| < d_1$ . Thus, in this case,  $C \xrightarrow{2} P \xrightarrow{1} T$  is a legal alternating path of length less than  $\ell(T)$  which is leading to  $T$  - in contradiction to the definition of  $\ell(T)$ . Therefore,  $Q$  is f-lying on  $p(O \rightarrow R)$  and thus  $\ell(Q) < \ell(R)$ , [see Fig. 2.3].

Let  $K$  be the first vertex which is f-lying on  $T \xrightarrow{1} W$  such that  $\ell(K) < \ell(R)$  [such a vertex exists, e.g.  $Q$ ]. Let  $V_1 \xrightarrow{3} V_2$  denote the segment from  $V_1$  to  $V_2$  on the al-path  $p(O \rightarrow K)$ . Let  $D$  be the 1-exposed vertex from which  $p(O \rightarrow K)$  starts. By the choice of  $W$ ,  $\ell(K) < |B \xrightarrow{1} K|$ . If we assume that  $p(O \rightarrow K)$  and  $K \xrightarrow{1} W$  are disjoint (except for the vertex  $K$ ), then the alternating path  $D \xrightarrow{3} K \xrightarrow{1} W$  is a legal al-path which is leading to  $W$ , such that its length is:  $\ell(K) + |K \xrightarrow{1} W| < |B \xrightarrow{1} K| + |K \xrightarrow{1} W| = |B \xrightarrow{1} W| = \ell(W)$  - in contradi-

ction to the definition of  $\ell(W)$ . Thus  $p(O \rightarrow K)$  and  $K \xrightarrow{1} W$  are not disjoint.

Let  $L$  be the first vertex which is  $f$ -lying on  $p(O \rightarrow K)$  and is common to  $T \xrightarrow{1} W$  [see Fig. 2.4]. Thus,  $\ell(L) < \ell(K) < \ell(R)$ .

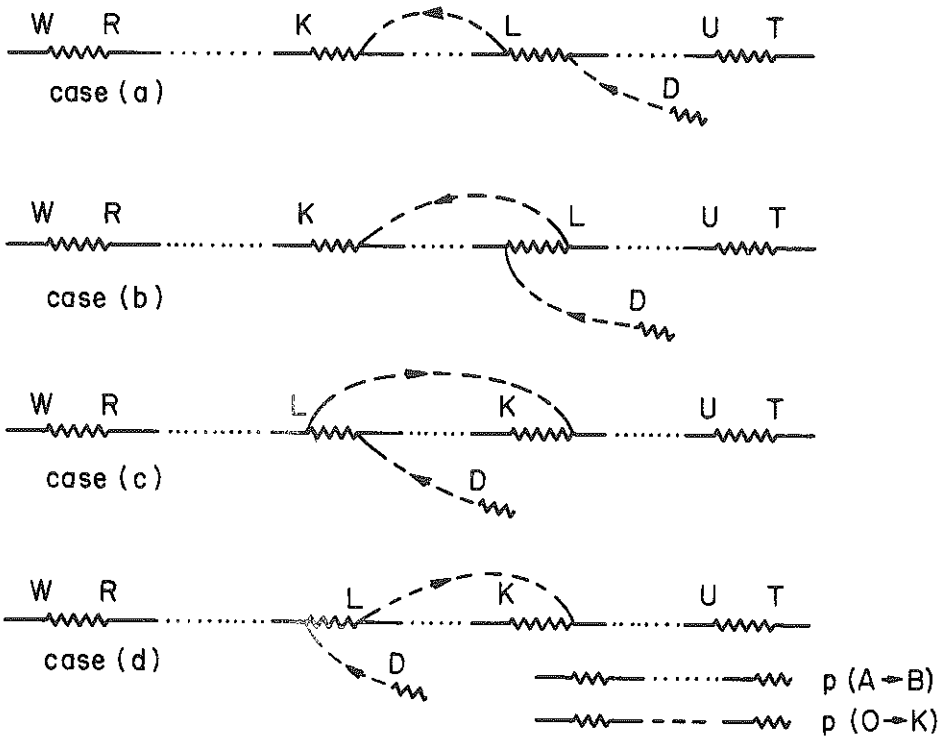


Fig. 2.4

Assume that  $L$  is  $f$ -lying on  $T \xrightarrow{1} W$ . Clearly  $L$  cannot be lying on  $T \xrightarrow{1} K$  [case (a) in Fig. 2.4], for in this case  $L$  contradicts the choice of  $K$ . Thus,  $L$  is  $f$ -lying on  $K \xrightarrow{1} W$  [case (c) in Fig. 2.4].

However, in this case the alternating path  $D \xrightarrow{4} L \xrightarrow{1} W$  is a legal  $a$ -path leading to  $W$ , such that its length is:  $\ell(L) + |L \xrightarrow{1} W| < \ell(K) + |K \xrightarrow{1} W| < |B \xrightarrow{1} K| + |K \xrightarrow{1} W| = |B \xrightarrow{1} W| = \ell(W)$  - in contradiction to the definition of  $\ell(W)$ . Therefore  $L$  is not  $f$ -lying on  $T \xrightarrow{1} W$  [cases (b) and (d) in Fig. 2.4]. However, in this case, the alternating path  $D \xrightarrow{4} L \xrightarrow{1} T$  is a legal  $a$ -path leading to  $T$  such that its length is:  $\ell(L) + |L \xrightarrow{1} T| < \ell(R) + |R \xrightarrow{1} T| = \ell(R) + d_1 < \ell(T)$  - in contradiction to the definition of  $\ell(T)$ .

Therefore, our assumption that  $T \neq N$  is false, i.e.,  $T = N$  and the theorem holds.

Q.E.D.

#### Corollary 2.4

Let  $Y$  be a vertex,  $N = \text{BASE}(Y)$  and  $M = \text{MATE}(N)$ . If  $d(MN) > r$  then no minimum legal augmenting path is passing through  $Y$ .

Proof. Since  $d(MN) > r$ , no minimum legal augmenting path is passing through  $N$ , and thus, by Theorem 2.1, no such path can pass through  $Y$ .

Q.E.D.

Hence, if  $Y$  is a vertex such that  $d(MN) > r$  (where  $N = \text{BASE}(Y)$ ,  $M = \text{MATE}(N)$ ), then  $Y$  is irrelevant for the present phase and therefore can be ignored.

### C. The Reduced Graph

Let  $V$  be the set of vertices of the original graph  $G$ , and let  $E$  be the set of its edges.

The reduced graph  $\tilde{G}$  is the graph whose set of vertices  $\tilde{V}$  and set of edges  $\tilde{E}$  are defined as follows:

$$\tilde{V} = \{N \mid N \in V, \text{BASE}(N)=N, d(N, \text{MATE}(N))=r\} \quad (2.3)$$

$$\tilde{E} = \{N_1 N_2 \mid N_1, N_2 \in \tilde{V}, N_1 \neq N_2, \text{ and there exists in } E \text{ at least one edge } XY \text{ such that } \text{BASE}(X)=N_1, \text{BASE}(Y)=N_2, d(XY)=r\}$$

[The reduced graph  $\tilde{G}$  can be derived from the original graph  $G$  by the following process:

(i) Remove from  $G$  all vertices which, by Corollary 2.4, are not relevant to the present phase, and remove all edges of  $G$  which are incident to these vertices. Remove all edges  $XY$  of  $G$  such that  $d(XY) > r$ .

(ii) Shrink all other vertices of  $G$  into their BASEs. Delete from the graph all the self-loops which are created by this process, and if parallel edges are created delete all of them but one.]

The following sequence of lemmas and theorems proves that the reduced graph  $\tilde{G}$  has the following property: Every maximal set of disjoint arbitrary legal augmenting paths in  $\tilde{G}$  correspond to a maximal set of disjoint minimum legal augmenting paths in  $G$ .



Lemma 2.6

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

Proof. (a) Let  $MN$  be a matched edge in  $\tilde{G}$ , i.e.,  $M=MATE(N)$ . Clearly  $M$  and  $N$  are also MATEs in  $G$ , and thus  $MN$  is a matched edge of  $G$ . Since  $M$  and  $N$  are vertices of  $\tilde{G}$ , by definition (2.3) of the reduced graph,  $d(MN)=r$ .

(b) Let  $YZ$  be a matched edge of  $G$  such that  $d(YZ)=r$ . Then, by the definition of  $BASE$ ,  $BASE(Y)=Y$  and  $BASE(Z)=Z$ . Thus, by definition (2.3), both  $Y$  and  $Z$  belong to  $\tilde{V}$  and the edge  $YZ$  belongs to  $\tilde{E}$ .

Q.E.D.

Corollary 2.5

The set of the 1-exposed [2-exposed] vertices in  $\tilde{G}$  is exactly the set of the 1-exposed [2-exposed] vertices of  $G$  which lie at the ends of minimum legal augmenting paths.

Proof. Let  $A_1$  be a 1-exposed vertex and let  $A_2$  be a 2-exposed vertex such that  $A_2=MATE(A_1)$ . Since  $\lambda(A_1)=0$ , then  $d(A_1 A_2)=r$  if and only if  $\lambda(A_2)=r$ , i.e., if and only if  $A_2$  is lying at the end of a minimum legal augmenting path  $p(B_1 \rightarrow A_2)$  in  $G$ . Clearly, in this case,  $A_1$  is lying at the end of the legal augmenting path  $p(A_1 \rightarrow B_1)$  (where  $B_1=MATE(B_2)$ ).

Q.E.D.

Definition. Let  $XY$  be an edge in  $G$  such that:  $N_1 = \text{BASE}(X)$ ,  $N_2 = \text{BASE}(Y)$ ,  $d(XY) = r$  and  $N_1 N_2$  is an edge of  $\tilde{G}$ . Then  $XY$  is called a source of  $N_1 N_2$ . [Notice that  $N_1 N_2$  may have some sources].

Lemma 2.7

Let  $N_1 N_2$  be an edge of  $\tilde{G}$  and let  $XY$  be its source in  $G$ .  $N_1 N_2$  is a free edge in  $\tilde{G}$  if and only if  $XY$  is a free edge in  $G$ .

Proof. By the definition of source,  $N_1 = \text{BASE}(X)$ , and thus by Corollary 2.1  $p(0 \rightarrow N_1)$  is a head of  $p(0 \rightarrow X)$ . Therefore  $\ell(N_1) \leq \ell(X)$ . Similarly  $\ell(N_2) \leq \ell(Y)$ . By the definition of source,  $d(XY) = r$ . If  $XY$  is a free edge in  $G$  then:  $\ell(N_1) + \ell(N_2) < \ell(X) + \ell(Y) + 1 = d(XY) = r$ , and thus, by definition (2.3),  $N_1 N_2$  cannot be matched in  $\tilde{G}$ . If  $XY$  is matched in  $G$ , then by Lemma 2.6,  $N_1 N_2$  is matched in  $\tilde{G}$ .

Q.E.D.

Lemma 2.8

Let  $N_1 N_2$  be a free edge in  $\tilde{G}$  and  $XY$  be its source. Then the segment  $p(N_1 \rightarrow X) \cdot p(N_2 \leftarrow Y)$  is an alternating path in  $G$  which contains  $\ell$  free edges and  $\ell - 1$  matched edges, where  $\ell = r - \ell(N_1 N_2)$ .

Proof. By Corollary 2.1  $p(0 \rightarrow N_1)$  is a head of  $p(0 \rightarrow X)$ . Therefore, the segment  $p(N_1 \rightarrow X)$  is an alternating path of length  $\ell(X) - \ell(N_1)$  whose first edge is free, while its last edge is matched. By similar considerations  $p(N_2 \leftarrow Y)$  is an alternating path of length  $\ell(Y) - \ell(N_2)$ , such that the edge which is incident to  $Y$  is matched

while the edge which is incident to  $N_2$  is free. The segment  $p(N_1 \rightarrow X) \cdot p(N_2 \leftarrow Y)$  consists of the two segments  $p(N_1 \rightarrow X)$  and  $p(N_2 \leftarrow Y)$  joined by the free edge  $XY$ . Therefore the segment  $p(N_1 \rightarrow X) \cdot p(N_2 \leftarrow Y)$  is an alternating path of length  $\ell$ , whose first and last edges are free, i.e., it contains  $\ell$  free edges and  $\ell-1$  matched edges. By the definition of source  $d(XY)=r$ . Therefore:

$$\begin{aligned} \ell &= (\ell(X) - \ell(N_1)) + (\ell(Y) - \ell(N_2)) + 1 = \ell(X) + \ell(Y) + 1 - \ell(N_1 N_2) \\ &= d(XY) - \ell(N_1 N_2) = r - \ell(N_1 N_2). \end{aligned}$$

Q.E.D.

Definition. Let  $\pi$  be an augmenting path in  $\tilde{G}$ , and let  $p(\pi)$  be the path in  $G$  which is received from  $\pi$  by replacing each free edge  $N_1 N_2$  on  $\pi$  by the segment  $p(N_1 \rightarrow X) \cdot p(N_2 \leftarrow Y)$  where  $XY$  is a source of  $N_1 N_2$ . Then the path  $p(\pi)$  is called a source of (the augmenting path)  $\pi$ . [Since each free edge  $N_1 N_2$  of  $\tilde{G}$  may have some sources in  $G$ , there may be some paths in  $G$  which are sources of the same augmenting path  $\pi$  in  $\tilde{G}$ ].

Lemma 2.9.

Let  $\pi(A \rightarrow B)$  be a legal augmenting path in  $\tilde{G}$  which leads from 1-exposed vertex  $A$  to the 2-exposed vertex  $B$ , and let  $p(\pi)$  be its source. Then  $p(\pi)$  is a legal augmenting path in  $G$ , which leads from  $A$  to  $B$ .

Proof. By Lemma 2.8, each segment  $p(N_1 \rightarrow X) \cdot p(N_2 \leftarrow Y)$  (where

$N_1 N_2$  is a free edge of  $\pi$ , and  $XY$  its source) is an alternating path in  $G$  whose first and last edges are free. Therefore by replacing each free edge  $N_1 N_2$  of  $\pi$  by a segment  $p(N_1 \rightarrow X) \cdot p(N_2 \leftarrow Y)$  we receive an alternating path which connects  $A$  and  $B$  - i.e., an augmenting path which leads from  $A$  to  $B$ . By Corollary 2.3, the BASE of all vertices on  $p(N_1 \rightarrow X)$  is the vertex  $N_1$ . Hence, every two segments  $p(N_1 \rightarrow X_i)$  and  $p(N_j \rightarrow X_j)$  on  $p(\pi)$  are disjoint. Since each of these segments is legal, it follows that  $p(\pi)$  is a legal augmenting path.

Q.E.D.

Lemma 2.10

Let  $N$  be a vertex which is  $f$ -lying on a legal augmenting path  $\pi(A \rightarrow B)$  in  $\tilde{G}$ . Then the length of the segment of  $p(\pi)$  from  $A$  to  $N$  is  $\ell(N)$ .

Proof. By the definition the lemma holds if  $N=A$ . Assume that  $N_1$  is the first vertex which is  $f$ -lying on  $\pi(A \rightarrow B)$ , for which the lemma does not hold. Let  $M_1 = \text{MATE}(N_1)$ , and let  $N_2$  be the vertex which precedes  $M_1$  on  $\pi(A \rightarrow B)$  (thus  $N_2$  is  $f$ -lying on  $\pi(A \rightarrow B)$ ). The segment of  $p(\pi)$  from  $A$  to  $N_1$  consists of the segment from  $A$  to  $N_2$  and the segment from  $N_2$  to  $N_1$ . By the choice of  $N_1$ , the length of the segment from  $A$  to  $N_2$  is  $\ell(N_2)$ . The length of the segment from  $N_2$  to  $N_1$  is the number of free edges on the segment  $p(M_1 \rightarrow X) \cdot p(N_2 \leftarrow Y)$  (where  $XY$  is a source of  $M_1 N_2$ ), and thus, by Lemma 2.8 it is equal to  $r - \ell(M_1 N_2)$ . Therefore, the length of the segment on  $p(\pi)$  from  $A$  to  $N_1$  is  $\ell(N_2) + r - \ell(M_1 N_2) = r - \ell(M_1)$ .

However, by Lemma 2.6,  $r - \ell(M_1) = \ell(N_1)$ . Thus, the lemma holds for  $N_1$  too.

Q.E.D.

### Corollary 2.6

Let  $\pi$  be a legal augmenting path in  $\tilde{G}$ . Then  $p(\pi)$  is a minimum legal augmenting path in  $G$ .

Proof. Let  $\pi(A \rightarrow B)$  be a legal augmenting path in  $\tilde{G}$ , which leads from the 1-exposed vertex  $A$  to the 2-exposed vertex  $B$ . By Lemma 2.9  $p(\pi)$  is a legal augmenting path in  $G$ . By Lemma 2.10 its length is  $\ell(B)$ , i.e., by Corollary 2.5, is equal to  $r$ . Therefore  $p(\pi)$  is minimum in  $G$ .

Q.E.D.

### Theorem 2.2

Let  $\{\pi_1, \dots, \pi_k\}$  be a set of disjoint legal augmenting paths in  $\tilde{G}$ . Then  $\{p(\pi_1), \dots, p(\pi_k)\}$  is a set of disjoint legal minimum augmenting paths in  $G$ .

Proof. Let  $V_i$  be a vertex on  $p(\pi_i)$  and  $V_j$  be a vertex on  $p(\pi_j)$ . By the construction of  $p(\pi_i)$   $V_i$  is lying on a segment  $p(N_i \rightarrow X_i)$  such that  $N_i = \text{BASE}(X_i)$  and  $N_i$  is lying on  $\pi_i$ . By Corollary 2.3,  $\text{BASE}(V_i) = N_i$ . By similar arguments,  $\text{BASE}(V_j) = N_j$  where  $N_j$  is lying on  $\pi_j$ . Since  $\pi_i$  and  $\pi_j$  are disjoint,  $N_i \neq N_j$  and thus  $V_i \neq V_j$ . Therefore  $p(\pi_i)$  and  $p(\pi_j)$  are disjoint.

Q.E.D.

Definition. Let  $p(A \rightarrow B)$  be a minimum legal augmenting path in  $G$ . By  $\pi_p$  we denote the path which is derived from  $p(A \rightarrow B)$  by the following process: Let  $N_1$  and  $N_2$  be two vertices on  $p(A \rightarrow B)$  such that  $N_1 = \text{BASE}(N_1)$ ,  $N_2 = \text{BASE}(N_2)$ . If there exists no other vertex  $N_3$  on  $p(A \rightarrow B)$  between  $N_1$  and  $N_2$  such that  $N_3 = \text{BASE}(N_3)$ , then replace the segment of  $p(A \rightarrow B)$  from  $N_1$  to  $N_2$ , by the edge  $N_1 N_2$ .

[The following theorem proves that  $\pi_p$  is a path of  $\tilde{G}$ . It also shows that  $\pi_p$  is in fact the path which is derived from  $p(A \rightarrow B)$  during the process of shrinking  $G$  into  $\tilde{G}$ , namely,  $\pi_p$  is derived from  $p(A \rightarrow B)$  by shrinking each vertex on  $p(A \rightarrow B)$  into its BASE ].

### Theorem 2.3

Let  $p(A \rightarrow B)$  be a minimum legal augmenting path in  $G$ . Then  $\pi_p$  is a legal augmenting path in  $\tilde{G}$  which leads from  $A$  to  $B$ .

Proof. Denote by  $V_1 \xrightarrow{i} V_2$  the segment from  $V_1$  to  $V_2$  on the path  $p(A \rightarrow B)$ . Let  $N_1$  be a vertex on  $p(A \rightarrow B)$  such that  $N_1 = \text{BASE}(N_1)$ , and  $M_1 = \text{MATE}(N_1)$ . By the definition of BASE,  $d(M_1 N_1) \geq r$  and thus  $\text{BASE}(M_1) = M_1$ . Therefore, if  $N_1$  is lying on  $\pi_p$  then  $\text{MATE}(N_1)$  is also lying on  $\pi_p$ , and the (matched) edge  $N_1 M_1$  belongs to  $\pi_p$ . In particular the 1-exposed vertex  $A$  and the 2-exposed vertex  $B$  belong to  $\pi_p$ .

Let  $N_1$  and  $N_2$  be two vertices on  $p(A \rightarrow B)$  such that  $N_1 = \text{BASE}(N_1)$ ,  $N_2 = \text{BASE}(N_2)$ , and there exists no other vertex  $N_3$  on

$N_1 \xrightarrow{1} N_2$  such that  $N_3 = \text{BASE}(N_3)$ . If  $N_1 \neq \text{MATE}(N_2)$  then the first edge of  $N_1 \xrightarrow{1} N_2$  is free (for otherwise,  $\text{MATE}(N_1)$  would contradict the assumption on  $N_1 \xrightarrow{1} N_2$ ). Similarly, the last edge of  $N_1 \xrightarrow{1} N_2$  is free. Therefore, by replacing the segment  $N_1 \xrightarrow{1} N_2$  of  $p(A \rightarrow B)$  by the (free) edge  $N_1 N_2$ , the resulting path is also alternating.

Thus,  $\pi_p$  is an alternating path which leads from  $A$  to  $B$ , i.e., it is an augmenting path. Since all the vertices of  $\pi_p$  are also vertices of  $p(A \rightarrow B)$  and their order on  $\pi_p$  is the same as on  $p(A \rightarrow B)$ , it follows that  $\pi_p$  is a legal augmenting path.

Let  $N_1 N_2$  be an edge of  $\pi_p$ . If  $N_1 N_2$  is a matched edge, then  $N_1 N_2$  is also an edge of  $p(A \rightarrow B)$  and thus (since  $p(A \rightarrow B)$  is a minimum legal augmenting path)  $d(N_1 N_2) \leq r$ . On the other hand, since  $N_1 = \text{BASE}(N_1)$ ,  $N_2 = \text{BASE}(N_2)$ , it follows by the definition of  $\text{BASE}$  that  $N_1 N_2$  is an edge of  $\mathcal{G}$ . If  $N_1 N_2$  is a free edge, then let  $X$  be the last vertex on  $N_1 \xrightarrow{1} N_2$  such that  $\text{BASE}(X) = N_1$  and let  $Y$  be the vertex which follows  $X$  on  $N_1 \xrightarrow{1} N_2$ . Let  $N_3 = \text{BASE}(Y)$ , then by Theorem 2.1  $N_3$  is lying on  $p(A \rightarrow B)$  and all vertices on  $N_3 \xrightarrow{1} Y$  have  $N_3$  as their  $\text{BASE}$ . Thus, if  $N_3 \neq N_2$  then  $N_3$  is lying on  $N_1 \xrightarrow{1} N_2$  and  $N_3 = \text{BASE}(N_3)$  - in contradiction to the assumption on  $N_1 \xrightarrow{1} N_2$ . Therefore  $N_3 = N_2$ , namely  $\text{BASE}(Y) = N_2$ . Since  $XY$  is lying on a minimum legal augmenting path,  $d(XY) \leq r$ . On the other hand, if  $d(XY) < r$  then by Lemma 2.5  $N_1 = N_2$ . Thus,  $d(XY) = r$ , and by the definition of  $\mathcal{G}$ ,  $N_1 N_2$  is a (free) edge of  $\mathcal{G}$ .

Thus,  $\pi_p$  is a legal augmenting path in  $\mathcal{G}$ .

Q.E.D.

Theorem 2.4

Let  $\{\pi_1, \pi_2, \dots, \pi_k\}$  be a maximal set of (arbitrary) disjoint legal augmenting paths in  $\tilde{G}$ . Then  $\{p(\pi_1), p(\pi_2), \dots, p(\pi_k)\}$  is a maximal set of disjoint minimum legal augmenting paths in  $G$ .

Proof. By Theorem 2.2  $\{p(\pi_1), p(\pi_2), \dots, p(\pi_k)\}$  is a set of disjoint legal minimum augmenting paths in  $G$ . Assume that it is not maximal, then there exists a minimum legal augmenting path  $p$  in  $G$  which is disjoint from each of the augmenting paths  $\{p(\pi_1), p(\pi_2), \dots, p(\pi_k)\}$ . However, by the definition, all the vertices of an augmenting path  $\pi_i$  in  $\tilde{G}$  are also vertices of its source  $p(\pi_i)$  in  $G$ . Also, all the vertices of the augmenting path  $\pi_p$  in  $\tilde{G}$  are also vertices of the augmenting path  $p$  in  $G$  from which  $\pi_p$  is derived. Therefore, the fact that  $p$  is disjoint from each of the augmenting paths  $\{p(\pi_1), p(\pi_2), \dots, p(\pi_k)\}$  in  $G$ , implies that  $\pi_p$  is disjoint to each of the augmenting paths  $\{\pi_1, \pi_2, \dots, \pi_k\}$  in  $\tilde{G}$  - in contradiction to the maximality of the set  $\{\pi_1, \pi_2, \dots, \pi_k\}$ . Therefore  $\{p(\pi_1), p(\pi_2), \dots, p(\pi_k)\}$  is a maximal set of disjoint minimum legal augmenting paths in  $G$ .

Q.E.D.

D. The Data Structures

Let  $n_r$  be the number of vertices of the reduced graph  $\tilde{G}$ , and let  $m_r$  be the number of its edges. During the performance of the second stage we use the following data structures:



BASE(Y) .  $BASE(Y)$  is the variable which is defined by definition (2.2) in Section B for each vertex  $Y$  of the original graph  $G$  .

VLIST .  $VLIST$  is a list which contains exactly all the  $n_r$  vertices of the reduced graph  $\tilde{G}$  . These vertices are arranged in non-decreasing order of their levels.

BSLIST(N) . For each vertex  $N$  of the reduced graph  $\tilde{G}$  ,  $BSLIST(N)$  is the list of all vertices  $Y$  of the original graph such that  $BASE(Y)=N$  . The list  $BSLIST(N)$  is defined only when  $N$  is found to be a vertex of the reduced graph.

ELIST(N) . For each vertex  $N$  of the reduced graph  $\tilde{G}$  ,  $ELIST(N)$  is the list of all vertices  $M$  such that  $NM$  is a free edge of the reduced graph. The list  $ELIST(N)$  is defined only when  $N$  is found to be a vertex of the reduced graph. The vertices of  $ELIST(N)$  are arranged in non-increasing order of their levels.

SOURCE( $N_1 N_2$ ) . For each free edge  $N_1 N_2$  of the reduced graph,  $SOURCE(N_1 N_2)$  is a free edge  $XY$  of the original graph  $G$  , such that  $XY$  is a source of  $N_1 N_2$  .

The second stage is performed in two parts:

In the first part, the vertices of the original graph are searched according to non-decreasing order of their levels. (In order to perform this search we use the  $LIST(j)$ 's which were computed during the first stage of the algorithm). For each vertex  $Y$  of the

original graph, we find  $\text{BASE}(Y)$  according to definition (2.2). Since when  $\text{BASE}(Y)$  is computed, all BASEs of the vertices whose levels are less than  $\ell(Y)$  are already known, this computation can be done in one step only. Let  $N=\text{BASE}(Y)$  and let  $M=\text{MATE}(N)$ . If  $d(MN)=r$ , then  $N$  is a vertex of the reduced graph. In this case, if  $Y=N$  then  $Y$  is attached at the end of  $\text{VLIST}$  and we define the lists:  $\text{BSLIST}(Y) \leftarrow \{Y\}$ ,  $\text{ELIST}(Y) \leftarrow \emptyset$ . If  $Y \neq N$ , then  $Y$  is inserted into  $\text{BSLIST}(N)$ . Thus, at the end of the first part of the second stage, all vertices of  $\tilde{G}$  are listed in  $\text{VLIST}$  in non-decreasing order of their levels, and for each vertex  $N$  of  $\tilde{G}$ ,  $\text{BSLIST}(N)$  contains exactly all vertices of  $G$  whose BASEs are  $N$ .

In the second part of the second stage, the edges of the original graph are searched in the following order: The vertices of the reduced graph are searched according to the list  $\text{VLIST}$  (i.e., according to the order of their levels). For each vertex  $N$  of the reduced graph the list  $\text{BSLIST}(N)$  is searched. Let  $Y$  be a vertex of  $\text{BSLIST}(N)$ ; then all free edges  $XY$  of the original graph are searched. Let  $M=\text{BASE}(X)$ . If by (2.3)  $MN$  is an edge of the reduced graph and  $XY$  is its source, then we check whether  $N$  is already contained in  $\text{ELIST}(M)$ , and if the answer is negative, we insert  $N$  as the first vertex of  $\text{ELIST}(M)$  and assign:  $\text{SOURCE}(MN) \leftarrow XY$  (notice that by the construction of  $\text{ELIST}(M)$ , in order to know whether  $N$  is already contained in  $\text{ELIST}(M)$ , it is sufficient to check whether  $N$  is the first vertex of  $\text{ELIST}(M)$ ). Thus, at the end of the second part of the second stage, all the free edges of  $\tilde{G}$  which are incident to  $M$  are listed in  $\text{ELIST}(M)$  in non-increasing order of their

levels, and for each free edge  $MN$  of  $\tilde{G}$ ,  $SOURCE(MN)$  is one of its sources in  $G$ .

### E. The Algorithm

0. [Initialization.] Assign:  $VLIST \leftarrow \emptyset$ ;  $j \leftarrow 0$ .

FIND VERTICES OF  $\tilde{G}$ :

1. [Search  $LIST(j)$ .] If all the vertices of  $LIST(j)$  have already been searched then go to 4. Else, let  $Y$  be a vertex in  $LIST(j)$  which has not been searched yet.
2. [Find  $BASE(Y)$ .] Let  $Z \leftarrow MATE(Y)$ . If  $\ell(Y) + \ell(Z) > r$  then assign:  $BASE(Y) \leftarrow Y$  and return to 1 [ $Y$  is not a vertex of the reduced graph]. If  $\ell(Y) + \ell(Z) = r$ , then assign:  $BASE(Y) \leftarrow Y$  and go to 3. [ $Y$  is a vertex of the reduced graph]. If  $\ell(Y) + \ell(Z) < r$  and  $Y$  has a 1-link, then assign:  $BASE(Y) \leftarrow BASE(LINK(Y))$ , insert  $Y$  into  $BSLIST(BASE(Y))$  and return to 1. If  $\ell(Y) + \ell(Z) < r$  and  $Y$  has a 2-link, then let  $S \leftarrow LINK(Y)$ , assign:  $BASE(Y) \leftarrow BASE(S)$ , insert  $Y$  into  $BSLIST(BASE(Y))$  and return to 1.
3. [ $Y$  is a vertex of the reduced graph.] Attach  $Y$  to the end of  $VLIST$ . Define:  $BSLIST(Y) \leftarrow \{Y\}$ ,  $ELIST(Y) \leftarrow \emptyset$ . Return to 1.
4. If  $j = r$  go to 5. [All vertices of  $\tilde{G}$  are listed in  $VLIST$  in non-decreasing order of their levels]. Else, assign:  $j \leftarrow j + 1$  and return to 1.

FIND EDGES OF  $\tilde{G}$  :

5. [Search the vertices of VLIST .] If all the vertices of VLIST have already been searched, then proceed to the third stage [for each vertex M of VLIST , all the edges MN of the reduced graph are listed in ELIST(M) in non-increasing order of their levels]. Else, let N be the first vertex of VLIST which has not been searched yet.
6. [Search the vertices of BSLIST(N) .] If all the vertices of BSLIST(N) have already been searched, then return to 5. Else let Y be a vertex of BSLIST(N) which has not been searched yet.
7. [Search the free edges XY in G .] If all the free edges of G which are incident to Y have already been searched, then return to 6. Else, let XY be a free edge of G which has not been searched yet.
8. If  $\lambda(X) + \lambda(Y) + 1 \neq r$  then return to 7 [XY is not a source of any free edge of  $\tilde{G}$ ]. Else let  $M \leftarrow \text{BASE}(X)$ . If  $\lambda(M) + \lambda(\text{MATE}(M)) > r$  [M does not belong to  $\tilde{G}$ ], then return to 7. Else, if  $M=N$  then return to 7 [XY is reduced to a self-loop in  $\tilde{G}$ ]. Else, if the first vertex of ELIST(M) is N, return to 7 [MN is already contained in ELIST(M)]. Else, insert N as the first vertex of ELIST(M), assign:  $\text{SOURCE}(MN) \leftarrow XY$  and return to 7.

F. The Complexity of the Second Stage

During the first part of the algorithm of the second stage (steps 1-4), we search the vertices of the original graph G. For

each vertex of  $G$ , a constant number of operations are performed. Thus, the complexity of the first part is  $O(n)$ .

During the second part of the algorithm (steps 5-8), we search the (free) edges of the original graph  $G$ . For every free edge of  $G$ , a constant number of operations are performed. Thus, the complexity of the second part is  $O(m)$ .

Therefore, the complexity of the second stage is  $O(m)$ .

### 3. THE THIRD STAGE

#### A. General Description

The goal of the third stage is to find a maximal set of disjoint legal augmenting paths in the reduced graph. In order to achieve this goal we perform a process which is similar to the process done during the first stage of the algorithm: The free edges of the reduced graph are searched, and by using these edges we try to build alternating paths which are leading to the vertices of the reduced graph. We also use the same technique which is used in the first stage in order to record these  $a_1$ -paths, namely the 1-link and 2-link. [In order to distinguish between the LINKS of the first stage and those of the third stage, we call the latter RLINKs. The  $a_1$ -path in the reduced graph which is leading to a vertex  $N$  is denoted by  $\tilde{p}(0 \rightarrow N)$  ].

However, while in the first stage we want to find minimum legal alternating paths, and for that purpose we search the free edges of the graph according to the BFS method, our goal during the third stage is to find disjoint legal augmenting paths in the graph. Apparently, the natural method by which the edges of the graph should be searched in order to build disjoint paths is the Depth First Search (DFS) method. However, by using this method, the problem of verifying the legality of the  $a_1$ -paths arises: The legality test of the first stage is based on the fact that if the assignment

LINK(R)=ST which defines for R a level  $j$  is illegal, then in the  $j$ -th substage both S and T belong to T(R) (Theorem 1.4). In other words, because of the BFS method which is used in the first stage, every odd loop in the graph is searched at the same time in both directions, and thus, the base of the loop is reached in the same substage from both directions of the loop - a property which makes possible the identification of an illegal path. Clearly, this is not the situation in the DFS method. Thus, the DFS method must be amended in order to enable both the construction of disjoint  $\alpha$ -paths and the verifications of their legality

Our solution to this problem is a composition of some version of the DFS method together with the method which is used by Gabow's algorithm [3]. The version of the DFS method (which is called the Highest Level First Search - HLFS - method) assures that the  $\alpha$ -paths which are found are disjoint. The method of Gabow assures that these  $\alpha$ -paths are legal.

#### B. The Parts of the Third Stage and the HLFS Method

The Label LIFE . In the third stage we attach to each vertex  $N$  of the reduced graph a label called LIFE(N) . In the beginning of the third stage, the LIFEs of all the vertices are equal to  $n+1$  . During the third stage the two vertices  $N$  and  $MATE(N)$  have always the same value of the label LIFE .

The algorithm of the third stage is performed in parts, where

in the  $i$ -th part we try to find a legal augmenting path which connects the  $i$ -th 1-exposed vertex to some 2-exposed vertex, such that this path is disjoint from the augmenting paths which may have been found in previous parts. This is done by starting from the  $i$ -th 1-exposed vertex and performing a search through free edges whose vertices have LIFE equal to  $n+1$ . Using the edges which are searched during the  $i$ -th part, we try to build alternating paths leading from the  $i$ -th 1-exposed vertex to vertices whose LIFE are  $n+1$ . If such a path which is leading to a vertex  $N$  is found, we assign:  $LIFE(N) \leftarrow i$ ,  $LIFE(MATE(N)) \leftarrow i$ . [The  $a_i$ -path itself is denoted by  $\tilde{p}(i \rightarrow N)$ ]. Thus, the meaning of  $LIFE(N) = i$  is that we have built an alternating path leading from the  $i$ -th 1-exposed vertex to either  $N$  or  $MATE(N)$ . We shall prove (Theorem 3.1) that there exists in the graph no legal augmenting path which passes through a vertex whose LIFE is  $i$  and is disjoint to all the augmenting paths which may have been found during the performance of the first  $i$  parts of the third stage. In other words, the vertices whose LIFE is  $i$ , may be ignored in latter parts. Therefore, if a vertex  $N$  is reached during the search of the  $i$ -th part, it does not participate in the searches which are performed in the following parts. It follows that once  $LIFE(N)$  is assigned a value other than  $n+1$ , it is not changed any more, and thus, the paths which are found during the performance of the  $i$ -th part of the third stage are disjoint from the paths which are found during the performance of the other parts.

[Notice that since in the third stage we are not looking for



minimum augmenting paths, no levels are defined for the vertices (although, we use the levels which have been defined during the first stage in order to perform the HLFS method). In fact, the role which the labels LIFE have in the third stage is very similar to the role which the levels have in the first stage; namely, LIFE(N) [ℓ(R)] identifies the part [substage] during which the vertex N [R] has been reached, and it is used to assure that the  $\alpha_1$ -paths which are found by the algorithm are really disjoint [minimum].]

The HLFS Method. The order by which the free edges are searched during the  $i$ -th part is determined by the following rule of the HLFS method: Let  $P$  be a vertex of highest level such that LIFE( $P$ )= $i$ , RLINK( $P$ ) $\neq 0$ , and not all the free edges which are incident to  $P$  have already been searched. Then, search a free edge  $PN$  which have not yet been searched.

[Notice that in order to find the vertex  $P$  we use the levels which were defined in the first stage. Since by Lemma 2.10 the level of a vertex  $P$  in the reduced graph is in fact the place of  $P$  on a minimum legal augmenting path in the original graph, we can think of the HLFS method as some kind of Depth First Search on minimum legal augmenting paths in the original graph].

Let  $PN$  be a free edge which is searched during the  $i$ -th part according to the HLFS method, and let  $M=MATE(N)$ . If LIFE( $N$ ) $<i$ , then nothing is done, and we proceed to perform the next search according to the HLFS method.

Otherwise, (if  $LIFE(N)$  is not less than  $i$ ), then one of the following three cases may happen:

(a)  $RLINK(N)=0$  and  $RLINK(M)=0$ . In this case both  $M$  and  $N$  has not been reached so far during the search of the third stage, and thus  $LIFE(M)=LIFE(N)=n+1$ . Therefore we assign  $LIFE(M) \leftarrow i$ ,  $LIFE(N) \leftarrow i$ . Similarly to the algorithm of the first stage, we also assign for  $M$  a 1-link which points to the vertex  $P$ :  $RLINK(M) \leftarrow P$ . This assignment defines the  $a$ -path  $\tilde{p}(i \rightarrow M)$  through the following concatenation:  $\tilde{p}(i \rightarrow M) = \tilde{p}(i \rightarrow P) \cdot N \cdot M$ . Like in the first stage, this  $a$ -path is legal, and thus no further tests are required.

(b)  $RLINK(N)=0$  but  $RLINK(M) \neq 0$ . In this case, we have already built an  $a$ -path  $\tilde{p}(i \rightarrow M)$  which is leading from the  $i$ -th 1-exposed vertex to  $M$ , and thus,  $LIFE(M)=LIFE(N)=i$ . Nothing is done in this case, and we proceed to perform the next search according to the HLFS method.

(c)  $RLINK(N) \neq 0$ . In this case, an  $a$ -path  $\tilde{p}(i \rightarrow N)$  which is leading to  $N$  has already been found in the graph. If  $PN$  is a bridge then the possibility of assigning a 2-link which points to this bridge has to be checked. However, since both  $\tilde{p}(i \rightarrow P)$  and  $\tilde{p}(i \rightarrow N)$  emerge from the same 1-exposed vertex, a 2-link which points to the bridge  $PN$  may define an illegal  $a$ -path, and thus a further test must be carried out. For this purpose we call the subroutine  $SRCHLOOP(P,N)$  which searches the  $a$ -paths  $\tilde{p}(i \rightarrow P)$  and  $\tilde{p}(i \rightarrow N)$  and finds all the vertices  $R$  which are lying on these paths such that  $RLINK(R)=0$ , and the assignment  $RLINK(R) \leftarrow NP$  (or  $RLINK(R) \leftarrow PN$ )

defines a legal al-path  $\tilde{p}(i \rightarrow R)$  leading to  $R$ . This subroutine (which is a slight improvement of Gabow's subroutine PAIR-LINK [3]) is described in Section C.

The  $i$ -th part of the third stage is terminated in one of the following two ways:

(a) When a legal augmenting path which is leading from the  $i$ -th 1-exposed vertex to some 2-exposed vertex is found in the graph. Since the LIFEs of all the vertices on this augmenting path are equal to  $i$ , this path is disjoint from all other augmenting paths which may have been found in previous parts.

(b) When there exists in the graph no free edge which has to be searched according to the HLFS method, i.e., when all the free edges which are incident to vertices whose LIFE is  $i$  have already been searched. Theorem 3.1 implies that in this case there exists no legal augmenting path which connects the  $i$ -th 1-exposed vertex to some 2-exposed vertex in the graph, such that this path is disjoint from all the augmenting paths which may have been found so far.

### C. The Subroutine SRCHLOOP

During the performance of the  $i$ -th part, the subroutine SRCHLOOP( $P, N$ ) is called each time when a bridge  $PN$  is searched and both  $P$  and  $N$  have RLINKs other than 0; namely, when both al-path  $\tilde{p}(i \rightarrow P)$  and  $\tilde{p}(i \rightarrow N)$  have already been built in the

graph. Since both  $a_1$ -paths emerge from the same 1-exposed vertex, any  $a_1$ -path which is defined by a 2-link to the bridge NP [or to the bridge PN] must be suspected of being illegal. The task of subroutine SRCHLOOP(P,N) is to find those vertices R which are lying on  $\tilde{p}(i \rightarrow P)$  [or on  $\tilde{p}(i \rightarrow N)$ ], such that  $RLINK(R)=0$  and the assignment  $RLINK(R) \leftarrow NP$  [or  $RLINK(R) \leftarrow PN$ ] defines a legal  $a_1$ -path  $\tilde{p}(i \rightarrow R)$  leading to R. If SRCHLOOP(P,N) finds such a vertex R, it also performs the assignment  $RLINK(R) \leftarrow NP$  [or  $RLINK(R) \leftarrow PN$ ].

[Notice that since SRCHLOOP(P,N) performs only legal assignments of the form  $RLINK(R) \leftarrow NP$  and  $RLINK(R) \leftarrow PN$ , there exist no potential RLINKs throughout the performance of the third stage, and thus all the RLINKs and the  $a_1$ -paths which are defined by them are well-defined].

The subroutine SRCHLOOP can be implemented by the subroutine PAIR-LINK which is described in Gabow's algorithm [3]. However, by using the properties of the HLFS method we can slightly improve and simplify the subroutine suggested by Gabow (though these modifications do not reduce the complexity of the subroutine).

Before we describe the subroutine SRCHLOOP, we prove some of its properties which are derived from the HLFS method.

### Lemma 3.1

Let  $N_1 M_2$  be a free edge in the reduced graph and  $N_2 = \text{MATE}(M_2)$ . Then  $\ell(N_1) < \ell(N_2)$ .

Proof. Let  $XY = \text{SOURCE}(N_1 M_2)$  (see the algorithm of the second stage). Since in the original graph  $p(O \rightarrow N_1)$  is a head of  $p(O \rightarrow X)$ ,  $\ell(N_1) \leq \ell(X)$ . Similarly  $\ell(M_2) \leq \ell(Y)$ . On the other hand, by Lemma 2.7,  $XY$  is a free edge of the original graph, and by the definition of source,  $d(XY) = r$ . Also, by Lemma 2.6,  $d(N_2 M_2) = r$ . Therefore:

$$\ell(N_1) + \ell(M_2) \leq \ell(X) + \ell(Y) < d(XY) = r = d(N_2 M_2) = \ell(N_2) + \ell(M_2).$$

Thus,  $\ell(N_1) < \ell(N_2)$ .

Q.E.D.

### Corollary 3.1

Let  $\pi$  be an alternating path in the reduced graph which is leading from  $M$  to  $P$  (i.e.,  $M$  is incident to a free edge of  $\pi$ , while  $P$  is incident to a matched edge of  $\pi$ ). Then  $\ell(M) < \ell(P)$ . [Notice that  $\pi$  is not necessarily a legal alternating path].

Definitions. Let  $P$  be a vertex of the reduced graph, such that  $\text{LIFE}(P) = i$ , or  $\text{LIFE}(P) = n+1$ . During the  $i$ -th part of the third stage, if  $\text{RLINK}(P) = 0$  then  $P$  is called passive; if  $\text{RLINK}(P) \neq 0$  and not all the free edges  $PN$  have already been searched then  $P$  is called active; if  $\text{RLINK}(P) \neq 0$ , and all the free edges  $PN$  have already been searched then  $P$  is called dead.

Lemma 3.2

If  $M$  is  $f$ -lying on  $\tilde{p}(i \rightarrow P)$  and  $P$  is not passive, then  $M$  is not passive. In particular, if  $\text{MATE}(M)$  is passive then  $\tilde{p}(i \rightarrow M)$  is a head of  $\tilde{p}(i \rightarrow P)$ .

Proof. Let  $S$  be the first vertex on  $\tilde{p}(M \rightarrow P)$  such that  $\tilde{p}(i \rightarrow S)$  is a head of  $\tilde{p}(i \rightarrow P)$  and  $S$  is not passive. If  $S=M$  then the lemma holds for  $M$ . Assume that  $S \neq M$ .  $S$  cannot have a 1-link (for, if  $\text{RLINK}(S)=R$ , then  $R$  contradicts the choice of  $S$ ). Thus, let  $RW=\text{RLINK}(S)$ . If  $M$  is lying on  $\tilde{p}(i \rightarrow R)$  then (by the definition of 2-link)  $R$  contradicts the choice of  $S$ . Thus,  $M$  is lying on  $\tilde{p}(W \rightarrow S)$ , and by the definition of 2-link, both  $M$  and  $\text{MATE}(M)$  are not passive. Therefore,  $M$  is not passive, and in particular, if  $\text{MATE}(M)$  is passive, then  $S=M$ , i.e.,  $\tilde{p}(i \rightarrow M)$  is a head of  $\tilde{p}(i \rightarrow P)$ .

Q.E.D.

Corollary 3.2

If  $\text{SRCHLOOP}(P,N)$  is called before  $\text{SRCHLOOP}(N,P)$ , then all the assignments of the forms  $\text{RLINK}(R) \leftarrow NP$  and  $\text{RLINK}(R) \leftarrow PN$  are performed by  $\text{SRCHLOOP}(P,N)$ .

Proof. When  $\text{SRCHLOOP}(P,N)$  is called, both  $P$  and  $N$  are not passive. Let  $M$  be a vertex which is  $f$ -lying on  $\tilde{p}(i \rightarrow P)$  [or on  $\tilde{p}(i \rightarrow N)$ ] and let  $R=\text{MATE}(M)$ . By Lemma 3.2  $M$  is not passive. Therefore, if  $R$  is passive, then  $M$  has a 1-link, and by Lemma 3.2  $\tilde{p}(i \rightarrow M)$  is a head of  $\tilde{p}(i \rightarrow P)$ . It follows by the definition of 2-link that the bridge  $NP$  [or the bridge  $PN$ ] can be assigned as  $\text{RLINK}(R)$ . Hence, by the definition of  $\text{SRCHLOOP}$ , if this assignment is legal, it is performed by  $\text{SRCHLOOP}(P,N)$ .

Q.E.D.

Lemma 3.3

If  $M$  is dead while  $MATE(M)$  is still passive, then  $MATE(M)$  will remain passive.

Proof. Let  $M$  be the first vertex which becomes dead, for which the lemma does not hold. Since  $M$  has a 1-link, our assumption implies that after  $M$  becomes dead,  $MATE(M)$  is given a 2-link to some bridge  $PQ$ . By the definition of a 2-link  $\tilde{p}(i \rightarrow M)$  is a head of  $\tilde{p}(i \rightarrow Q)$  and thus (by Corollary 3.1)  $\ell(M) < \ell(Q)$ .

Assume that when  $M$  becomes dead,  $Q$  is still passive. Let  $S$  be the first vertex on  $\tilde{p}(M \rightarrow Q)$  such that  $\tilde{p}(i \rightarrow S)$  is a head of  $\tilde{p}(i \rightarrow Q)$  and when  $M$  becomes dead  $S$  is still passive. If  $S$  has a 1-link then let  $R = RLINK(S)$ ,  $W = MATE(S)$ . Since  $\tilde{p}(i \rightarrow R)$  is a head of  $\tilde{p}(i \rightarrow Q)$ , then by the choice of  $S$ , when  $M$  becomes dead  $R$  is not passive. Also, by Corollary 3.1  $\ell(M) \leq \ell(R)$  (where the equation holds only if  $R = M$ ) and thus, by the HLFS method,  $R$  becomes dead before  $M$ . It follows that  $RW$  is searched before  $M$  becomes dead, and thus the assignment  $RLINK(S) \leftarrow R$  is performed before  $M$  becomes dead - in contradiction to the assumption on  $S$ .

If  $S$  has a 2-link, then let  $RW = RLINK(S)$ . Both  $\tilde{p}(i \rightarrow M)$  and  $\tilde{p}(i \rightarrow R)$  are heads of  $\tilde{p}(i \rightarrow S)$ . Since  $S$  is lying on  $\tilde{p}(M \rightarrow Q)$ , then by the definition of 2-link  $RLINK(S)$  must be (well-) defined before the assignment  $RLINK(MATE(M)) \leftarrow PQ$  is performed. Therefore, the assignment  $RLINK(S) \leftarrow RW$  is performed when  $MATE(M)$  is still passive and thus  $\tilde{p}(i \rightarrow R)$  cannot be a head

of  $\tilde{p}(i \rightarrow M)$ , but  $\tilde{p}(i \rightarrow M)$  is a head of  $\tilde{p}(i \rightarrow R)$ . By the choice of  $S$ , when  $M$  becomes dead  $R$  is not passive. Since by Corollary 3.1  $\lambda(R) \geq \lambda(M)$  (and the equation holds only if  $R=M$ ), then by the HLFS method  $RW$  is searched before  $M$  becomes dead. Assume that when  $RW$  is searched  $W$  is still passive: Since by Corollary 3.1  $\lambda(\text{MATE}(W)) > \lambda(R)$ , then by the HLFS method  $\text{MATE}(W)$  is not active when  $RW$  is searched. If when  $RW$  is searched  $\text{MATE}(W)$  is passive, then we would assign  $\text{RLINK}(\text{MATE}(W)) \leftarrow R$  and  $RW$  would not be a bridge. If when  $RW$  is searched,  $\text{MATE}(W)$  is dead, then by our choice of  $M$ ,  $W$  would remain passive - in contradiction to the fact that  $RW$  is a bridge and  $\text{RLINK}(S) = RW$ . Therefore, the assumption that  $W$  is passive when  $RW$  is searched is false. It follows that when  $RW$  is searched,  $\text{SRCHLOOP}(R, W)$  is called. Therefore, by Corollary 3.2, the assignment  $\text{RLINK}(S) \leftarrow RW$  is performed before  $M$  becomes dead - in contradiction to the assumption on  $S$ .

Hence, when  $M$  becomes dead  $Q$  is not passive. Since  $\lambda(Q) > \lambda(M)$ ,  $Q$  becomes dead before  $M$ , i.e., the edge  $QP$  is searched before  $M$  becomes dead. Assume that when  $QP$  is searched  $P$  is still passive. By Corollary 3.1,  $\lambda(\text{MATE}(P)) > \lambda(Q)$ , and thus by the HLFS method  $\text{MATE}(P)$  is not active when  $QP$  is searched. If  $\text{MATE}(P)$  is passive when  $QP$  is searched, then we would assign  $\text{RLINK}(\text{MATE}(P)) \leftarrow Q$  - in contradiction to the fact that  $QP$  is a bridge. If  $\text{MATE}(P)$  is dead when  $QP$  is searched, then by the choice of  $M$ ,  $Q$  would remain passive - in contradiction to the assumption that  $\text{RLINK}(\text{MATE}(M)) = PQ$ . Thus when  $QP$  is searched,  $P$  is not passive, and therefore  $\text{SRCHLOOP}(Q, P)$  is called. By Corollary



3.2 it follows that the assignment  $RLINK(M) \leftarrow PQ$  is performed before  $M$  becomes dead - in contradiction to the fact that when  $M$  becomes dead,  $MATE(M)$  is still passive.

Therefore, there exists no vertex  $M$  which contradicts the Lemma.

Q.E.D.

### Corollary 3.3

If the subroutine  $SRCHLOOP(P,N)$  performs an assignment of the form  $RLINK(R) \leftarrow PN$  or  $RLINK(R) \leftarrow NP$ , then  $\ell(P) \geq \ell(N)$ , and when  $SRCHLOOP(P,N)$  is called both  $P$  and  $N$  are active.

Proof. By the definition, when  $SRCHLOOP(P,N)$  is called  $P$  is active and  $N$  is not passive. Assume that  $N$  is dead, then the bridge  $NP$  has been searched before  $SRCHLOOP(P,N)$  is called. If, when  $NP$  is searched  $P$  is not passive, then  $SRCHLOOP(N,P)$  would be called, and by Corollary 3.2, it would perform the assignment  $RLINK(R) \leftarrow PN$  [or  $RLINK(R) \leftarrow NP$ ] - in contradiction to the fact that this assignment is performed by  $SRCHLOOP(P,N)$ . Thus, when  $NP$  is searched,  $P$  is still passive. By Corollary 3.1,  $\ell(MATE(P)) > \ell(N)$ , and thus by the HLFS method when  $NP$  is searched  $MATE(P)$  is not active. If, when  $NP$  is searched,  $MATE(P)$  is passive, we would assign  $RLINK(MATE(P)) \leftarrow N$  - in contradiction to the fact that  $NP$  is a bridge. If, when  $NP$  is searched,  $MATE(P)$  is dead, then by Lemma 3.3  $P$  would remain passive - in contradiction to the assumption that  $RLINK(R) = PN$  [or  $RLINK(R) = NP$ ].

Therefore, when  $SRCHLOOP(P,N)$  is called,  $N$  is not dead but is active. By the HLFS method  $\ell(P) \geq \ell(N)$ .

Q.E.D.

Lemma 3.4

Let  $M$  be a vertex which is f-lying on  $\tilde{p}(i \rightarrow P)$  where  $P$  is active. If  $MATE(M)$  is passive, then  $M$  is active.

Proof. By Lemma 3.2  $M$  cannot be passive. Assume that  $M$  is dead. By Corollary 3.1 and the HLFS method, when  $M$  becomes dead  $P$  is not yet active, i.e., it is passive. Let  $S$  be the first vertex on  $\tilde{p}(M \rightarrow P)$  such that  $\tilde{p}(i \rightarrow S)$  is a head of  $\tilde{p}(i \rightarrow P)$  and when  $M$  becomes dead  $S$  is still passive. If  $S$  has a 1-link, then let  $R = RLINK(S)$  and  $W = MATE(S)$ . By the choice of  $S$ , when  $M$  becomes dead  $R$  is not passive. By Corollary 3.1  $\ell(M) \leq \ell(R)$  (and the equation holds only if  $R = M$ ). Therefore, by the HLFS method,  $R$  becomes dead before  $M$ , and thus  $RW$  is searched and the assignment  $RLINK(S) \leftarrow R$  is performed before  $M$  becomes dead - in contradiction to the assumption on  $S$ . If  $S$  has a 2-link, then let  $RW = RLINK(S)$ . By Lemma 3.2, when  $P$  is active,  $S$  is already not passive. It follows that when the assignment  $RLINK(S) \leftarrow RW$  is performed,  $MATE(M)$  is still passive. Therefore by the definition of 2-link  $M$  is not on  $\tilde{p}(W \rightarrow S)$  but  $R$  is lying on  $\tilde{p}(M \rightarrow P)$ . Thus, by the choice of  $S$ , when  $M$  becomes dead  $R$  is not passive. Therefore, by Corollary 3.1 and the HLFS method  $RW$  is searched before  $M$  becomes dead. Assume that when  $RW$  is searched  $W$  is still passive. Since by

Lemma 3.1  $\ell(\text{MATE}(W)) > \ell(R)$ , when  $RW$  is searched  $\text{MATE}(W)$  is not active. If  $\text{MATE}(W)$  is passive, we would assign  $\text{RLINK}(\text{MATE}(W)) \leftarrow R$  - in contradiction to the fact that  $RW$  is a bridge. If, when  $RW$  is searched,  $\text{MATE}(W)$  is already dead, then by Lemma 3.3  $W$  would remain dead - in contradiction to the assumption that  $\text{RLINK}(S) = RW$ . Thus, when  $RW$  is searched,  $W$  is not passive. Therefore, when  $RW$  is searched,  $\text{SRCHLOOP}(R, W)$  is called, and by Corollary 3.2 the assignment  $\text{RLINK}(S) \leftarrow RW$  is performed - in contradiction to the assumption that when  $M$  becomes dead  $S$  is still passive.

Thus, when  $P$  is active,  $M$  must be active too.

Q.E.D.

### Lemma 3.5

Assume that the procedure of the third stage is not in  $\text{SRCHLOOP}$  and let  $M$  be an active vertex such that  $\text{MATE}(M)$  is still passive. If  $P$  is an active vertex such that  $\ell(P) \geq \ell(M)$ , then  $\tilde{p}(i \rightarrow M)$  is a head of  $p(i \rightarrow P)$ . (If  $\ell(P) = \ell(M)$  then  $P = M$ ).

Proof. Let  $P$  be the first vertex which becomes active and contradicts the lemma. We first show that when  $\text{RLINK}(P)$  is assigned,  $P$  satisfies the lemma: Assume that  $P$  has a 1-link  $R = \text{RLINK}(P)$  and let  $W = \text{MATE}(P)$ . When  $RW$  is searched, there exists in the graph no active vertex of level higher than  $\ell(R)$ . Therefore, by the choice of  $P$ , for every active vertex  $M$  whose  $\text{MATE}$  is passive,  $\tilde{p}(i \rightarrow M)$  is a head of  $\tilde{p}(i \rightarrow R)$  and thus a head of  $\tilde{p}(i \rightarrow P)$ .

Assume that  $P$  has a 2-link  $RW = \text{RLINK}(P)$ . If the assignment  $\text{RLINK}(P) \leftarrow RW$  is performed by  $\text{SRCHLOOP}(R, W)$ , then by the HLFS method, when this subroutine is called, there exists in the graph no active vertex of level higher than  $\ell(R)$ . Therefore, by the choice of  $P$ , for every active vertex  $M$  such that when  $\text{SRCHLOOP}(R, W)$  is called  $\text{MATE}(M)$  is passive,  $\tilde{p}(i \rightarrow M)$  is a head of  $\tilde{p}(i \rightarrow R)$  and thus a head of  $\tilde{p}(i \rightarrow P)$ .

If the assignment  $\text{RLINK}(P) \leftarrow RW$  is performed by  $\text{SRCHLOOP}(W, R)$ , then by Corollary 3.3 when this subroutine is called  $R$  is active. Thus, by the choice of  $P$ , for every active vertex  $M$  such that  $\ell(M) \leq \ell(R)$  and when  $\text{SRCHLOOP}(W, R)$  is called  $\text{MATE}(M)$  is passive,  $\tilde{p}(i \rightarrow M)$  is a head of  $\tilde{p}(i \rightarrow R)$  and thus a head of  $\tilde{p}(i \rightarrow P)$ . Consider now an active vertex  $M$  such that  $\ell(M) > \ell(R)$  and when  $\text{SRCHLOOP}(W, R)$  is called  $\text{MATE}(M)$  is passive. By the HLFS method  $\ell(M) \leq \ell(W)$ , and thus by the choice of  $P$ ,  $\tilde{p}(i \rightarrow M)$  is a head of  $\tilde{p}(i \rightarrow W)$ . Let  $S$  be the first vertex which is f-lying on  $\tilde{p}(i \rightarrow W)$  and is common to  $\tilde{p}(i \rightarrow R)$ . By Corollary 3.1  $\ell(S) \leq \ell(W) < \ell(\text{MATE}(R))$ . If  $S$  is not f-lying on  $\tilde{p}(i \rightarrow R)$  then by Corollary 3.1  $\ell(\text{MATE}(R)) \leq \ell(S)$  - in contradiction to the inequality  $\ell(S) < \ell(\text{MATE}(R))$ . Thus,  $S$  is f-lying on  $\tilde{p}(i \rightarrow R)$  and  $\ell(S) \leq \ell(R)$ . Since  $\tilde{p}(i \rightarrow M)$  is a head of  $\tilde{p}(i \rightarrow W)$  and  $\ell(M) > \ell(R)$ , it follows that  $M$  is f-lying on  $\tilde{p}(S \rightarrow W)$  and  $M \neq S$ . Therefore, the assignment  $\text{RLINK}(\text{MATE}(M)) \leftarrow RW$  is legal and it is performed by  $\text{SRCHLOOP}(W, R)$ . It follows that if  $M$  is an active vertex such that  $\ell(M) > \ell(R)$  and when  $\text{SRCHLOOP}(W, R)$  is called  $\text{MATE}(M)$  is passive, then when the performance of  $\text{SRCHLOOP}(W, R)$  is terminated,  $\text{MATE}(M)$  is no more passive.

Thus, when the activation of the subroutine SRCHLOOP which performs the assignment  $RLINK(P) \leftarrow RW$  is terminated,  $P$  satisfies the lemma for every active vertex  $M$  such that when SRCHLOOP was called  $MATE(M)$  was passive. Since no passive vertex  $M$  whose  $MATE$  is passive becomes active during the performance of SRCHLOOP, this implies that when the performance of SRCHLOOP is terminated  $P$  satisfies the lemma for every active vertex  $M$  whose  $MATE$  is passive.

Let  $M$  be a vertex whose  $MATE$  is passive, such that  $M$  becomes active while  $P$  is active. Since  $MATE(M)$  is passive,  $M$  has a 1-link  $RLINK(M)=Q$ . When the assignment  $RLINK(M) \leftarrow Q$  is performed, there exists in the graph no active vertex of level higher than  $\ell(Q)$ . In particular,  $\ell(P) \leq \ell(Q)$  and thus  $\ell(P) < \ell(M)$ . Therefore  $M$  is not relevant to the lemma. It follows that the lemma holds for  $P$ .

Q.E.D.

#### Corollary 3.4

(a)  $SRCHLOOP(P,N)$  does not perform any assignment of the form  $RLINK(R) \leftarrow PN$ .

(b)  $SRCHLOOP(P,N)$  performs an assignment of the form  $RLINK(R) \leftarrow NP$  only if  $\ell(P) \geq \ell(MATE(R)) > \ell(N)$ .

Proof. Assume that  $SRCHLOOP(P,N)$  performs an assignment of the form  $RLINK(R) \leftarrow NP$  or  $RLINK(R) \leftarrow PN$ . By the definition of 2-link

MATE(R) is f-lying on  $\tilde{p}(i \rightarrow P)$  or on  $\tilde{p}(i \rightarrow N)$  respectively. Since by Corollary 3.3 when SRCHLOOP(P,N) is called both P and N are active, it follows by Lemma 3.4 that MATE(R) is active too. Also by Corollary 3.3  $\ell(P) \geq \ell(N)$ . Therefore, if  $\ell(\text{MATE}(R)) \leq \ell(N)$ , then by Lemma 3.5,  $\tilde{p}(i \rightarrow \text{MATE}(R))$  is a head of both  $\tilde{p}(i \rightarrow P)$  and  $\tilde{p}(i \rightarrow N)$ , and thus the assignment RLINK(R)  $\leftarrow$  NP or RLINK(R)  $\leftarrow$  PN is illegal and is not performed by SRCHLOOP.

Hence, SRCHLOOP(P,N) performs assignments of the forms RLINK(R)  $\leftarrow$  NP and RLINK(R)  $\leftarrow$  PN only if  $\ell(\text{MATE}(R)) > \ell(N)$ .

(a) If RLINK(R) = PN, then MATE(R) is f-lying on  $\tilde{p}(i \rightarrow N)$ , and thus by Corollary 3.1  $\ell(N) \geq \ell(\text{MATE}(R))$ . Therefore SRCHLOOP(P,N) does not perform any assignment of the form RLINK(R)  $\leftarrow$  PN.

(b) If RLINK(R) = NP, then MATE(R) is f-lying on  $\tilde{p}(i \rightarrow P)$ , and thus by Corollary 3.1  $\ell(P) \geq \ell(\text{MATE}(R))$ . Therefore SRCHLOOP(P,N) performs an assignment of the form RLINK(R)  $\leftarrow$  NP only if  $\ell(P) \geq \ell(\text{MATE}(R)) > \ell(N)$ .

Q.E.D.

### Corollary 3.5

Assume that the subroutine SRCHLOOP(P,N) is called, where  $\ell(P) > \ell(N)$ . Let Q be the passive vertex closest to N on  $\tilde{p}(i \rightarrow N)$  and let M = MATE(Q) [see Fig. 3.1]. Then

(a)  $\tilde{p}(i \rightarrow M)$  is a common head of both  $\tilde{p}(i \rightarrow N)$  and  $\tilde{p}(i \rightarrow P)$ .

(b)  $SRCHLOOP(P,N)$  performs an assignment of the form  $RLINK(R) \leftarrow NP$  if and only if  $R$  is a passive vertex on  $\tilde{p}(M \rightarrow P)$ .

Proof. (a) When  $SRCHLOOP(P,N)$  is called  $P$  is active and  $N$  is not passive. We first show that the condition  $\ell(P) > \ell(N)$  implies that  $N$  is active too: Assume that  $N$  is dead. Therefore the bridge  $NP$  has been searched before  $SRCHLOOP(P,N)$  is called. By the HLFS method, when  $NP$  is searched,  $P$  is not yet active, and thus it is passive. Since by Lemma 3.1  $\ell(MATE(P)) > \ell(N)$ , when  $NP$  is searched  $MATE(P)$  is not active. If, when  $NP$  is searched,  $MATE(P)$  is passive, then we would assign  $RLINK(MATE(P)) \leftarrow N$  - in contradiction to the fact that  $NP$  is a bridge. If, when  $NP$  is searched,  $MATE(P)$  is dead, then by Lemma 3.3,  $P$  would remain passive - in contradiction to the fact that when  $SRCHLOOP(P,N)$  is called,  $P$  is active. Thus, when  $SRCHLOOP$  is called  $N$  is active, and by Lemma 3.4  $M$  is active too. On the other hand,  $M$  is  $f$ -lying on  $\tilde{p}(i \rightarrow N)$  and thus, by Corollary 3.1,  $\ell(M) \leq \ell(N) < \ell(P)$ . Therefore by Lemma 3.5  $\tilde{p}(i \rightarrow M)$  is a common head of  $\tilde{p}(i \rightarrow N)$  and of  $\tilde{p}(i \rightarrow P)$  (see Fig. 3.1).

(b) If  $SRCHLOOP(P,N)$  performs an assignment of the form  $RLINK(R) \leftarrow NP$ , then  $R$  is a passive vertex on  $\tilde{p}(i \rightarrow P)$ . Since  $\tilde{p}(i \rightarrow M)$  is a head of both  $\tilde{p}(i \rightarrow N)$  and  $\tilde{p}(i \rightarrow P)$ ,  $R$  cannot be lying on  $\tilde{p}(i \rightarrow M)$  (for, otherwise the assignment  $RLINK(R) \leftarrow NP$  is illegal). Thus  $R$  is lying on  $\tilde{p}(M \rightarrow P)$ .

Let  $R$  be the passive vertex closest to  $P$  on  $\tilde{p}(M \rightarrow P)$  (see Fig. 3.1). By Lemma 3.2,  $\tilde{p}(i \rightarrow MATE(R))$  is a head of  $\tilde{p}(i \rightarrow P)$  and

by Lemma 3.4  $MATE(R)$  is active and thus has a 1-link. Therefore, by the definition of 2-link,  $NP$  can be assigned as  $RLINK(R)$ . Let  $S$  be the first vertex which is  $f$ -lying on  $\tilde{p}(i \rightarrow P)$ , and is common to  $\tilde{p}(i \rightarrow N)$ . By the proof of Lemma 3.5  $S$  is  $f$ -lying on  $\tilde{p}(i \rightarrow N)$ , and thus  $\ell(S) \leq \ell(N)$ . If  $R$  is not lying on  $\tilde{p}(S \rightarrow P)$ , then by Corollary 3.1  $\ell(MATE(R)) \leq \ell(S) \leq \ell(N) < \ell(P)$ , and thus, by Lemma 3.5,  $\tilde{p}(i \rightarrow MATE(R))$  is a common head of  $\tilde{p}(i \rightarrow P)$  and  $\tilde{p}(i \rightarrow N)$ . However, this implies that  $\tilde{p}(i \rightarrow MATE(R))$  is a head of  $\tilde{p}(i \rightarrow M)$  - in contradiction to the fact that  $R$  is lying on  $\tilde{p}(M \rightarrow P)$ . Therefore,  $R$  is lying on  $\tilde{p}(S \rightarrow P)$  and the assignment  $RLINK(R) \leftarrow NP$  is legal and is performed by  $SRCHLOOP(P,N)$ .

Let  $R'$  be a vertex on  $\tilde{p}(M \rightarrow P)$ , such that after  $RLINK(R)$  is defined,  $R'$  is the passive vertex closest to  $P$ . Clearly, after  $RLINK(R)$  is defined,  $SRCHLOOP(P,N)$  performs the assignment

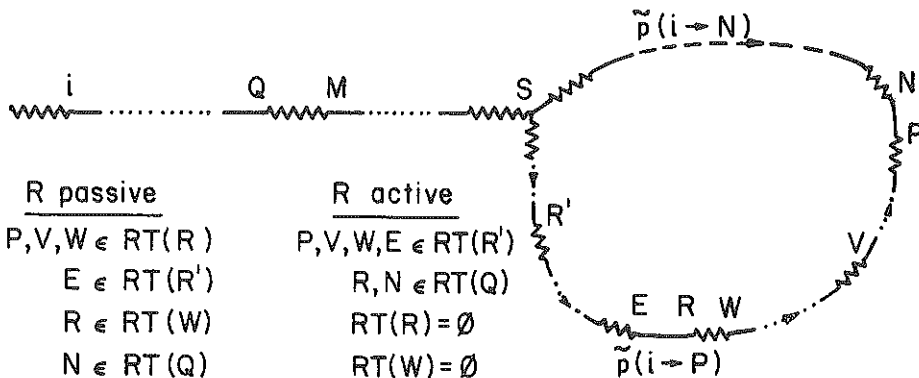


Fig. 3.1 - Subroutine SRCHLOOP(P,N)



$RLINK(R') \leftarrow NP$ . Thus, for each passive vertex  $R'$  on  $\tilde{p}(M \rightarrow P)$ , the assignment  $RLINK(R') \leftarrow NP$  is performed by  $SRCHLOOP(P, N)$ .

Q.E.D.

Corollaries 3.4 and 3.5 enable us to formulate the performance of subroutine  $SRCHLOOP(P, N)$  as follows:

- (a) If  $\ell(N) \geq \ell(P)$  then return [by Corollary 3.4];
- (b) Find the passive vertex  $Q$  which is the closest to  $N$  on  $\tilde{p}(i \rightarrow N)$ ;
- (c) Find the passive vertex  $R$  which is the closest to  $P$  on  $\tilde{p}(i \rightarrow P)$ . If  $R=Q$  then return [by Corollary 3.5];
- (d) Assign:  $RLINK(R) \leftarrow NP$  and return to (c) [by Corollary 3.5].

#### D. The Data Structures

The main procedure. In order to perform the search of the free edges as required by the HLFS method, we use the lists  $VLIST$  and  $ELIST(N)$  which were defined in the second stage of the algorithm [recall that  $VLIST$  is a list of the vertices of the reduced graph, where these vertices are arranged in nondecreasing order of their levels. For each vertex  $N$  of the reduced graph,  $ELIST(N)$  is the list of all vertices  $M$  such that  $NM$  is a free edge of the reduced graph].

In order to handle the bookkeeping which is required by the

HLFS method, we define a data structure ACTLIST which contains all the active vertices arranged according to their levels. The purpose of ACTLIST is to enable us to find an active vertex of highest level, as it is required by the rule of the HLFS method. There are some possible structures by which ACTLIST can be implemented. Since different structures determine different complexities for the third stage, we shall discuss this subject in Section G: "The Complexity of the Third Stage".

For the purpose of performing the third stage by parts, we use the label LIFE, as it is explained in Section B.

The subroutine SRCHLOOP. For each vertex N of the reduced graph, we define the variable TOP(N) as follows<sup>\*</sup>:

If N is passive then  $TOP(N)=MATE(N)$ ; else, TOP(N) is the passive vertex which is the closest to N on  $p(i \rightarrow N)$ .

Similarly to the first stage, we define for each vertex Q of the reduced graph a set RT(Q) as follows:

$$RT(Q) = \{N | TOP(N)=Q\} .$$

---

\* TOP(N) in our algorithm is in fact identical to TOP(LINK(N)) in Gabow's algorithm [3]. Like the variable TAIL in the first stage, TOP is not used by all the implementations of the third stage, and it is introduced here in order to enable a better understanding of the sets RT(Q).

The variables  $TOP$  and the sets  $RT$  are changed only when a passive vertex  $R$  becomes active, i.e., when  $RLINK(R)$  is defined. However, it is not difficult to see that an assignment of a 1-link does not change any set  $RT$ . On the other hand, an assignment of a 2-link,  $RLINK(R) \leftarrow NP$ , changes exactly the  $TOP$ s of all vertices  $V$  such that  $VERT(R)$ , and clearly it also changes  $TOP(R)$ .

Let  $V$  be a vertex which belongs to  $RT(R)$  before  $R$  becomes active and let  $W = MATE(R)$ . By Lemma 3.2  $\tilde{p}(i \rightarrow W)$  is a head of  $\tilde{p}(i \rightarrow V)$ . On the other hand,  $W$  has a 1-link  $E = RLINK(W)$ , and thus  $p(i \rightarrow V) = p(i \rightarrow E) \cdot p(R \rightarrow V)$  (see Fig. 3.1). It follows that after  $R$  becomes active, the new  $TOP(V)$  is the same as  $TOP(E)$ . Therefore, if  $E \in RT(R')$  [i.e., if  $TOP(E) = R'$ ], then after  $R$  becomes active  $VERT(R')$ . Namely, the assignment  $RLINK(R) \leftarrow NP$  implies the following change:  $RT(R') \leftarrow RT(R') \cup RT(R)$ ,  $RT(R) \leftarrow \emptyset$ .

Consider now  $TOP(R)$ : By the definition of  $TOP$ , before  $R$  becomes active,  $TOP(R) = W$ . On the other hand, by the definition of 2-link, after the assignment  $RLINK(R) \leftarrow NP$  is performed,  $\tilde{p}(i \rightarrow R) = \tilde{p}(i \rightarrow N) \cdot \tilde{p}(R \leftarrow P)$  and all the vertices on  $\tilde{p}(R \leftarrow P)$  are not passive. Thus, after  $R$  becomes active, the new  $TOP(R)$  is the same as  $TOP(N)$ . Therefore, if  $N \in RT(Q)$  [i.e., if  $TOP(N) = Q$ ], then the assignment  $RLINK(R) \leftarrow NP$  implies the following change:  $RT(Q) \leftarrow RT(Q) \cup RT(W)$ ,  $RT(W) \leftarrow \emptyset$ .

Using the definitions of the sets  $RT$ , we can now formulate the performance of the subroutine  $SRCHLOOP(P, N)$  as follows:

- If  $j=r$  then [M is a 2-exposed vertex and  $\tilde{p}(i \rightarrow M)$  is an augmenting path; the  $i$ -th part is successfully terminated] assign  $i \leftarrow i+1$  and return to PART. Else, assign  $P \leftarrow M$  and go to 3.
6. [N is not passive.] If  $\text{RLINK}(\text{MATE}(P))=N$  then [PN is not a bridge] return to 3. If  $\ell(P) \leq \ell(N)$  then return to 3 [although PN is a bridge,  $\text{SRCHLOOP}(P,N)$  performs nothing]. Else [ $\ell(P) > \ell(N)$ ] call  $\text{SRCHLOOP}(P,N)$ . Return to 3.
7. [Find an active vertex of highest level.] If  $\text{ACTLIST}=\emptyset$  then [there are no active vertices in the graph. The  $i$ -th part is terminated unsuccessfully without finding any legal augmenting path] assign  $i \leftarrow i+1$  and return to PART. Else find in ACTLIST a vertex P of maximal level\*. If  $\text{LIFE}(P) < i$  then [P has remained in ACTLIST from some previous part] delete\* P from ACTLIST and repeat 7. Else, [P is an active vertex of highest level in the graph], assign  $j \leftarrow \ell(P)$  and go to 3.

The Subroutine  $\text{SRCHLOOP}(P,N)$

1. FIND\* the vertex Q such that  $\text{NERT}(Q)$ .
- FIND\* the vertex R such that  $\text{PERT}(R)$
2. If  $R=Q$  then return.

---

\* The exact implementation of these instructions is discussed in Section G.

The variables  $TOP$  and the sets  $RT$  are changed only when a passive vertex  $R$  becomes active, i.e., when  $RLINK(R)$  is defined. However, it is not difficult to see that an assignment of a 1-link does not change any set  $RT$ . On the other hand, an assignment of a 2-link,  $RLINK(R) \leftarrow NP$ , changes exactly the  $TOP$ s of all vertices  $V$  such that  $VERT(R)$ , and clearly it also changes  $TOP(R)$ .

Let  $V$  be a vertex which belongs to  $RT(R)$  before  $R$  becomes active and let  $W = MATE(R)$ . By Lemma 3.2  $\tilde{p}(i \rightarrow W)$  is a head of  $\tilde{p}(i \rightarrow V)$ . On the other hand,  $W$  has a 1-link  $E = RLINK(W)$ , and thus  $p(i \rightarrow V) = p(i \rightarrow E) \cdot p(R \rightarrow V)$  (see Fig. 3.1). It follows that after  $R$  becomes active, the new  $TOP(V)$  is the same as  $TOP(E)$ . Therefore, if  $E \in RT(R')$  [i.e., if  $TOP(E) = R'$ ], then after  $R$  becomes active  $VERT(R')$ . Namely, the assignment  $RLINK(R) \leftarrow NP$  implies the following change:  $RT(R') \leftarrow RT(R') \cup RT(R)$ ,  $RT(R) \leftarrow \emptyset$ .

Consider now  $TOP(R)$ : By the definition of  $TOP$ , before  $R$  becomes active,  $TOP(R) = W$ . On the other hand, by the definition of 2-link, after the assignment  $RLINK(R) \leftarrow NP$  is performed,  $\tilde{p}(i \rightarrow R) = \tilde{p}(i \rightarrow N) \cdot \tilde{p}(R \rightarrow P)$  and all the vertices on  $\tilde{p}(R \rightarrow P)$  are not passive. Thus, after  $R$  becomes active, the new  $TOP(R)$  is the same as  $TOP(N)$ . Therefore, if  $N \in RT(Q)$  [i.e., if  $TOP(N) = Q$ ], then the assignment  $RLINK(R) \leftarrow NP$  implies the following change:  $RT(Q) \leftarrow RT(Q) \cup RT(W)$ ,  $RT(W) \leftarrow \emptyset$ .

Using the definitions of the sets  $RT$ , we can now formulate the performance of the subroutine  $SRCHLOOP(P, N)$  as follows:

- (a) If  $\ell(P) \leq \ell(N)$  then return;
- (b) FIND the vertex  $Q$  such that  $NERT(Q)$  [i.e., find  $TOP(N)$ ];
- (c) FIND the vertex  $R$  such that  $PERT(R)$  [i.e., find  $TOP(P)$ ];
- (d) If  $R=Q$  then return;
- (e) Assign:  $RLINK(R) \leftarrow NP$  ;
- (f) Let  $W \leftarrow MATE(R)$  . Perform the UNION :  $RT(Q) \leftarrow RT(Q) \cup RT(W)$  ;  
 $RT(W) \leftarrow \emptyset$  ;
- (g) Let  $E \leftarrow RLINK(W)$  . FIND the vertex  $R'$  such that  $EERT(R')$   
 [i.e., find  $TOP(E)$  ] . Perform the UNION :  
 $RT(R') \leftarrow RT(R') \cup RT(R)$  ;  $RT(R) \leftarrow \emptyset$  ;
- (h) Assign  $R \leftarrow R'$  and return to (d).

The description of  $SRCHLOOP(P,N)$  includes the instructions FIND and UNION. The exact performance of these instructions depends on the implementation of the sets  $RT$  . Since different implementations lead to different complexities of the third stage, we shall discuss this subject in Section G: "The Complexity of the Third Stage".

## E. The Algorithm

### The Main Procedure

#### 0. [Initialization.]

- (a) For each vertex  $N$  of the reduced graph, assign:  
 $LIFE(N) \leftarrow n+1$  ;  $RLINK(N) \leftarrow 0$  ;  $RT(N) \leftarrow \{MATE(N)\}$  .
- (b) Assign:  $ACTLIST \leftarrow \emptyset$  .
- (c)  $i \leftarrow 1$  .

## PART:

1. [Start the  $i$ -th part.] Assign:  $P \leftarrow \text{VLIST}(i)$  . If  $\ell(P) \neq 0$  then proceed to the fourth stage [all the 1-exposed vertices of the reduced graph have already been searched]. If  $\text{LIFE}(P) < n+1$  then assign  $i \leftarrow i+1$  and return to PART [we have already found an augmenting path which passes through the  $i$ -th 1-exposed vertex].
2. [Start the search] Assign:  $\text{RLINK}(P) \leftarrow \text{DUM}$ ,  $j \leftarrow 0$  [ $j$  is the highest level of an active vertex]. Insert\*  $P$  into  $\text{ACTLIST}$  and assign:  $\text{LIFE}(P) \leftarrow i$ ;  $\text{LIFE}(\text{MATE}(P)) \leftarrow i$  [ $P$  becomes active].
3. [Search from vertex  $P$  .] If all the edges of  $\text{ELIST}(P)$  have already been searched, then delete\*  $P$  from  $\text{ACTLIST}$  , and go to 7 [ $P$  becomes dead]. Else, let  $PN$  be an edge in  $\text{ELIST}(P)$  which has not been searched yet.
4. [Search the edge  $PN$  .] If  $\text{LIFE}(N) < i$  then return to 3 [ $N$  is not relevant to the  $i$ -th part]. If  $\text{RLINK}(N) \neq 0$  then go to 6. Else, let  $M \leftarrow \text{MATE}(N)$  .
5. [ $N$  is passive.] If  $\text{RLINK}(M) \neq 0$  then return to 3. Else, assign [a 1-link assignment]:  $\text{LIFE}(M) \leftarrow i$  ;  $\text{LIFE}(N) \leftarrow i$  ;  $\text{RLINK}(M) \leftarrow P$  . Insert\*  $M$  into  $\text{ACTLIST}$  and assign:  $j \leftarrow \ell(M)$  [ $M$  becomes an active vertex of highest level in the graph].

---

\* The exact implementation of these instructions is discussed in Section G.

- If  $j=r$  then  $[M$  is a 2-exposed vertex and  $\tilde{p}(i \rightarrow M)$  is an augmenting path; the  $i$ -th part is successfully terminated] assign  $i \leftarrow i+1$  and return to PART . Else, assign  $P \leftarrow M$  and go to 3.
6.  $[N$  is not passive.] If  $\text{RLINK}(\text{MATE}(P))=N$  then  $[PN$  is not a bridge] return to 3. If  $\ell(P) \leq \ell(N)$  then return to 3 [although  $PN$  is a bridge,  $\text{SRCHLOOP}(P,N)$  performs nothing]. Else  $[\ell(P) > \ell(N)]$  call  $\text{SRCHLOOP}(P,N)$  . Return to 3.
7.  $[\text{Find an active vertex of highest level.}]$  If  $\text{ACTLIST}=\emptyset$  then  $[\text{there are no active vertices in the graph. The } i\text{-th part is terminated unsuccessfully without finding any legal augmenting path}]$  assign  $i \leftarrow i+1$  and return to PART . Else find in ACTLIST a vertex  $P$  of maximal level<sup>\*</sup> . If  $\text{LIFE}(P) < i$  then  $[P$  has remained in ACTLIST from some previous part] delete<sup>\*</sup>  $P$  from ACTLIST and repeat 7. Else,  $[P$  is an active vertex of highest level in the graph], assign  $j \leftarrow \ell(P)$  and go to 3.

The Subroutine SRCHLOOP(P,N)

1. FIND<sup>\*</sup> the vertex  $Q$  such that  $\text{NERT}(Q)$  .  
FIND<sup>\*</sup> the vertex  $R$  such that  $\text{PERT}(R)$
2. If  $R=Q$  then return.

---

\* The exact implementation of these instructions is discussed in Section G.



3. Assign:  $RLINK(R) \leftarrow NP$  . Insert<sup>\*</sup>  $R$  into ACTLIST .
4. Let  $W \leftarrow MATE(R)$  . Perform the UNION<sup>\*</sup> :  $RT(Q) \leftarrow RT(Q) \cup RT(W)$  ;  
 $RT(W) \leftarrow \emptyset$  .
5. Let  $E \leftarrow RLINK(W)$  . FIND<sup>\*</sup> the vertex  $R'$  such that  $EERT(R')$  .  
Perform the UNION<sup>\*</sup> :  $RT(R') \leftarrow RT(R') \cup RT(R)$  ;  $RT(R) \leftarrow \emptyset$  .
6. If  $R' \neq Q$  then assign:  $R \leftarrow R'$  and return to 3.
7. [ $R$  is the vertex of highest level which has become active during the present activation of SRCHLOOP .] If  $\ell(R) \leq j$  then return. Else [ $R$  is an active vertex of highest level in the graph] assign:  $j \leftarrow \ell(R)$  ;  $P \leftarrow R$  and return.

#### F. The Correctness of the Third Stage

The goal of the third stage is to find a maximal set of disjoint legal augmenting paths in the reduced graph. Throughout the performance of the third stage we construct alternating paths leading to the vertices of the reduced graph. These paths are defined by 1-links and 2-links. Clearly, the  $a_1$ -paths which are defined by the 1-links are legal. The discussion of Section C assures that the  $a_1$ -paths which are defined by the 2-links are legal too. Therefore, the augmenting paths which we find in the graph are legal. The fact that these paths are disjoint is verified by using the label LIFE as it is explained in Section B.

---

\* The exact implementation of these instructions is discussed in Section G.

In this section we prove that the set of the (disjoint and legal) augmenting paths which we find during the performance of the third stage is maximal (and thus complete the proof of the correctness of the third stage). For this purpose, it is sufficient to show that if  $R$  is a vertex such that  $LIFE(R)=i$ , then there exists in the graph no legal augmenting path which passes through  $R$  and is disjoint from all the augmenting paths which may have been found during the first  $i$  parts (Theorem 3.1). This theorem is proved by the following sequence of lemmas:

Lemma 3.6

Let  $LIFE(M)=i$ . If at the end of the  $i$ -th part  $M$  is still active while  $MATE(M)$  is passive, then a legal augmenting path which passes through  $M$  has been found in the  $i$ -th part.

Proof. Since at the end of the  $i$ -th part  $M \in ACTLIST$ , the  $i$ -th part cannot be terminated in step 7 of the algorithm. It follows that the  $i$ -th part is terminated in step 5 of the algorithm, where a legal augmenting path leading to a 2-exposed vertex  $B$  is found in the graph. Since  $B$  is an active vertex and  $\ell(M) \leq r = \ell(B)$ , Lemma 3.5 implies that  $\tilde{p}(i \rightarrow M)$  is a head of  $\tilde{p}(i \rightarrow B)$ .

Q.E.D.

Lemma 3.7

Let  $M$  be a vertex such that  $LIFE(M)=i$ . If at the end of the  $i$ -th part  $M$  is dead while  $MATE(M)$  is passive, then there exists

in the graph no legal augmenting path which passes through  $M$  and is disjoint from all the vertices whose LIFEs are less than  $i$ .

Proof. Assume that the lemma does not hold: Let  $\pi$  be a legal augmenting path which does not pass through any vertex whose LIFE is less than  $i$ , and let  $M$  be the first vertex of  $\pi$  which becomes dead such that  $LIFE(M)=i$  and  $MATE(M)$  is passive. Let  $R$  be a vertex of highest level on  $\pi$  such that  $\tilde{p}(i \rightarrow M)$  is a head of  $\tilde{p}(i \rightarrow R)$ . By Lemma 3.2, when  $M$  becomes active  $R$  is still passive. Thus, by Lemma 3.3, when  $M$  becomes active,  $MATE(R)$  is not dead. Assume that when  $M$  becomes active,  $MATE(R)$  is active. Since  $M$  has a 1-link, it follows by the HLFS method that when  $M$  becomes active,  $\lambda(MATE(R)) < \lambda(M)$ , and thus by Lemma 3.5  $\tilde{p}(i \rightarrow MATE(R))$  is a head of  $\tilde{p}(i \rightarrow M)$  - in contradiction to the fact that  $\tilde{p}(i \rightarrow M)$  is a head of  $\tilde{p}(i \rightarrow R)$ . Therefore, when  $M$  becomes active  $MATE(R)$  is still passive.

By Lemma 3.4, at the end of the  $i$ -th part,  $R$  is not active, and therefore  $R$  cannot be a 2-exposed vertex (for, otherwise the  $i$ -th part would be terminated in step 5 of the algorithm, when  $R$  becomes active). Hence, there exists on  $\pi$  a free edge  $RW$ . If when  $M$  becomes active  $W$  is already dead, then the edge  $WR$  has been searched before  $M$  becomes active, i.e., when  $R$  and  $MATE(R)$  were still passive. However, in this case we would assign  $RLINK(MATE(R)) \leftarrow W$  - in contradiction to the fact that when  $M$  becomes active  $MATE(R)$  is still passive. Thus, when  $M$  becomes active,  $W$  is not dead. Assume that when  $M$  becomes active,  $W$  is active. Since

M has a 1-link, it follows by the HLFS method that  $\ell(W) < \ell(M)$  and thus by Lemma 3.5  $\tilde{p}(i \rightarrow W)$  is a head of  $\tilde{p}(i \rightarrow M)$ . Therefore,  $\tilde{p}(i \rightarrow W)$  is a head of  $\tilde{p}(i \rightarrow R)$  and  $\ell(W) < \ell(R)$ . Since at the end of the  $i$ -th part  $R$  is dead, it follows that the edge  $RW$  is searched during the  $i$ -th part and the subroutine  $SRCHLOOP(R,W)$  is called. However, Corollary 3.5 implies that in this case the assignment  $RLINK(MATE(M)) \leftarrow WR$  is performed - in contradiction to the fact that at the end of the  $i$ -th part  $MATE(M)$  is passive. Therefore, when  $M$  becomes active  $W$  is neither dead nor active, namely it is passive.

Let  $S = MATE(W)$ . By Lemma 3.1  $\ell(S) > \ell(R)$  and thus  $\ell(S) > \ell(M)$ . If while  $M$  is active,  $S$  is active too (or  $S$  becomes active), then by Lemma 3.5  $\tilde{p}(i \rightarrow M)$  is a head of  $\tilde{p}(i \rightarrow S)$  - in contradiction to the choice of  $R$ . On the other hand, the choice of  $M$  implies that when  $M$  becomes active  $S$  is not dead. Therefore, when  $M$  becomes active,  $S$  is passive, and it is still passive when  $M$  becomes dead. On the other hand, since  $RW$  is searched before  $M$  becomes dead, it follows that when  $M$  becomes dead  $W$  is not passive (for otherwise we would assign  $RLINK(S) \leftarrow R$ ). Therefore,  $W$  becomes active when  $M$  is active. However, since  $W$  has a 1-link, it follows by the HLFS method that  $\ell(W) > \ell(M)$  and thus  $W$  becomes dead before  $M$  - in contradiction to the choice of  $M$ .

Thus, the lemma is correct.

Q.E.D.

Lemma 3.8

Let  $R$  and  $M$  be two active vertices such that  $\ell(R) \geq \ell(M) \geq \lceil \frac{r}{2} \rceil$ . Then  $M$  is lying on  $\tilde{p}(i \rightarrow R)$ .

Proof. Let  $R$  be the first vertex which becomes active for which the lemma does not hold. First we show that when  $R$  becomes active, it is in accord with the lemma: If  $R$  has a 1-link,  $RLINK(R)=S$ , then by the HLFS method, when  $R$  becomes active there exists no active vertex of level higher than  $\ell(S)$ . Therefore, for each active vertex  $M$ ,  $\ell(M) \leq \ell(S)$ , and by the choice of  $R$ , if  $\ell(M) \geq \lceil \frac{r}{2} \rceil$  then  $M$  is lying on  $\tilde{p}(i \rightarrow S)$  and thus on  $\tilde{p}(i \rightarrow R)$ .

Assume that  $R$  has a 2-link,  $RLINK(R)=NP$ . By Corollary 3.5, the assignment  $RLINK(R) \leftarrow NP$  is performed by  $SRCHLOOP(P,N)$ , and by the HLFS method, when  $SRCHLOOP(P,N)$  is called there exists no active vertex of level higher than  $\ell(P)$ . Thus, for each active vertex  $M$ ,  $\ell(M) \leq \ell(P)$  and by the choice of  $R$ , if  $\ell(M) \geq \lceil \frac{r}{2} \rceil$  then  $M$  is lying on  $\tilde{p}(i \rightarrow P)$ . If  $M$  is a vertex which becomes active during the performance of  $SRCHLOOP(P,N)$ , then by Corollary 3.5  $M$  is lying on  $\tilde{p}(i \rightarrow P)$ . Therefore, when  $SRCHLOOP(P,N)$  is terminated, every active vertex  $M$  such that  $\ell(M) \geq \lceil \frac{r}{2} \rceil$  is lying on  $\tilde{p}(i \rightarrow P)$ . If  $M$  is lying on  $\tilde{p}(R \rightarrow P)$  then  $M$  is also lying on  $\tilde{p}(i \rightarrow R)$ . Assume that  $M$  is not lying on  $\tilde{p}(R \rightarrow P)$ : If  $M$  is f-lying on  $\tilde{p}(i \rightarrow P)$  (but is not lying on  $\tilde{p}(R \rightarrow P)$ ) then by Corollary 3.1  $\ell(M) > \ell(R)$  and thus  $M$  is not relevant to the lemma. If  $M$  is f-lying on  $\tilde{p}(i \rightarrow P)$  (but is not lying on  $\tilde{p}(R \rightarrow P)$ ) then by Corollary 3.1  $\ell(M) < \ell(MATE(R))$ . However, since  $\ell(R) + \ell(MATE(R)) = r$  while

$\ell(R) \geq \lceil \frac{r}{2} \rceil$ , it follows that  $\ell(\text{MATE}(R)) \leq \lceil \frac{r}{2} \rceil$ , and thus  $\ell(M) < \lceil \frac{r}{2} \rceil$  and again  $M$  is not relevant to the lemma.

Therefore, when  $R$  becomes active it is in accord with the lemma. Let  $M$  be a vertex which becomes active when  $R$  is active, such that  $\ell(R) \geq \ell(M) \geq \lceil \frac{r}{2} \rceil$ . By the HLFS method,  $M$  cannot have a 1-link and thus when  $M$  becomes active  $\text{MATE}(M)$  is not passive. By Lemma 3.3 it follows that when  $M$  becomes active  $\text{MATE}(M)$  is still active. Since  $\ell(M) + \ell(\text{MATE}(M)) = r$ , it follows that  $\ell(\text{MATE}(M)) \leq \lceil \frac{r}{2} \rceil \leq \ell(R)$  and thus by Lemma 3.5  $\text{MATE}(M)$  is lying on  $\tilde{p}(i \rightarrow R)$ . Therefore,  $M$  is lying on  $\tilde{p}(i \rightarrow R)$ .

Q.E.D.

### Corollary 3.6

Let  $M$  be a vertex such that  $\text{LIFE}(M) = i$  and  $\ell(M) \geq \lceil \frac{r}{2} \rceil$ . If at the end of the  $i$ -th part  $M$  is active, then a legal augmenting path which passes through  $M$  has been found in the  $i$ -th part.

Proof. Since at the end of the  $i$ -th part  $M \in \text{ACTLIST}$ , the  $i$ -th part cannot be terminated in step 7 of the algorithm. Therefore, the  $i$ -th part is terminated in step 5 of the algorithm, where a legal augmenting path leading to a 2-exposed vertex  $R$  is found. Since  $R$  is an active vertex and  $\ell(R) = r \geq \ell(M) \geq \lceil \frac{r}{2} \rceil$ , Lemma 3.8 implies that  $M$  is lying on this augmenting path.

Q.E.D.

Theorem 3.1

Let  $R$  be a vertex such that  $LIFE(R)=i$ . Then there exists in the graph no legal augmenting path which passes through  $R$  and is disjoint from all the augmenting paths which may have been found during the first  $i$  parts.

Proof. Let  $i$  be the first part for which the theorem does not hold, and let  $R$  be a vertex of highest level such that  $LIFE(R)=i$  and  $R$  does not satisfy the theorem; namely, there exists a legal augmenting path  $\pi$  which passes through  $R$  and is disjoint from all the augmenting paths which may have been found during the first  $i$  parts. If at the end of the  $i$ -th part  $R$  is passive, then by Lemma 3.7  $MATE(R)$  cannot be dead and by Lemma 3.6  $MATE(R)$  is not active. On the other hand, since  $LIFE(R)=i$ , both  $R$  and  $MATE(R)$  cannot be passive. Therefore, at the end of the  $i$ -th part  $R$  is not passive. Since both  $R$  and  $MATE(R)$  do not satisfy the theorem, then by the choice of  $R$ ,  $\ell(R) \geq \ell(MATE(R))$  and thus  $\ell(R) \geq \lceil \frac{r}{2} \rceil$ . Therefore, by Corollary 3.6, at the end of the  $i$ -th part  $R$  is not active. It follows that at the end of the  $i$ -th part  $R$  is dead.

Clearly  $R$  is not a 2-exposed vertex. Thus, there exists on  $\pi$  a free edge  $RW$ . Let  $S=MATE(W)$ , then by Lemma 3.1  $\ell(S) > \ell(R)$ . Since at the end of the  $i$ -th part  $R$  is dead, the edge  $RW$  has been searched during the  $i$ -th part. On the other hand, by the choice of  $i$ ,  $LIFE(S)$  is not less than  $i$ . Therefore, at the end of the

$i$ -th part  $LIFE(S)=i$  - in contradiction to the choice of  $R$ .

Hence, the theorem is correct.

Q.E.D.

### G. The Complexity of the Third Stage

Let  $n_r$  be the number of vertices of the reduced graph and let  $m_r$  be the number of its edges (clearly,  $n_r \leq n$ ,  $m_r \leq m$ ). Except for the operations on ACTLIST and the operations on the sets  $RT$ , a constant number of operations are performed by the algorithm of the third stage per each vertex and each free edge of the reduced graph. Therefore, except for the operations on ACTLIST and on the sets  $RT$ , the complexity of the third stage is  $O(m_r)$ .

The operations on ACTLIST. The following operations are performed on ACTLIST :

- (a) Each time a vertex becomes active, it is inserted into ACTLIST (steps 2 and 5 of the main procedure, and step 3 of subroutine SRCHLOOP );
- (b) Each time a vertex becomes dead, it is deleted from ACTLIST (steps 3 and 7 of the main procedure);
- (c) Each time a vertex is deleted from ACTLIST, we have to find in ACTLIST a vertex of maximal level.

Thus, throughout the whole performance of the third stage, each



of the instructions: "insert", "delete" and "find maximum" may be performed on ACTLIST at most  $n_r$  times.

There are some structures by which ACTLIST can be implemented:

(i) The simplest implementation is to represent ACTLIST as a simple list. In this case, each of the instructions "insert" and "delete" is performed in one step while "find maximum" is performed in (at most)  $n_r$  steps. This implementation determines a total complexity of  $O(n_r^2)$  for the operations on ACTLIST.

(ii) A more efficient implementation is to represent ACTLIST by some variation of a binary tree (e.g., by a 2-3 tree, see [6]). In this case, each of the instructions "insert", "delete" and "find maximum" is performed in  $O(\lg n_r)$  steps, and thus the total complexity of the operations on ACTLIST is  $O(n_r \lg n_r)^*$ .

(iii) By using the data structure which was suggested by Van Emde Boas [7], each of the operations "insert", "delete" and "find maximum" can be performed in  $O(\lg \lg n_r)$  steps, and thus the total complexity is  $O(n_r \lg \lg n_r)$ .

---

\* Since the levels of the vertices are not greater than  $r$ , by using a slightly more complicated data structure, we can lower the complexity of each operation to be  $O(\lg r)$ , and thus achieve a total complexity  $O(n_r \lg r)$ .

The operations on the sets  $RT$  . The operations on the sets  $RT$  are performed by the subroutine  $SRCHLOOP$  .

The following operations are required:

- (a) FIND the vertex  $R$  such that  $PERT(R)$  : Two such operations are required (in step 1) per each activation of  $SRCHLOOP$  , and one operation is required (in step 5) per each vertex of the reduced graph which is given a 2-link;
- (b) Perform the UNION of the sets  $RT(R)$  and  $RT(R')$  : Two such operations are performed (in steps 4 and 5) per each vertex of the reduced graph which is given a 2-link.

The subroutine  $SRCHLOOP$  may be activated at most once per each free edge of the reduced graph. Also, at most  $\lfloor \frac{n_r}{2} \rfloor$  vertices of the reduced graph can be given a 2-link. Therefore, the instruction  $FIND$  may be required at most  $m_r + \lfloor \frac{n_r}{2} \rfloor$  times throughout the whole performance of the third stage, while the instruction  $UNION$  may be required at most  $n_r$  times.

There are some possibilities to implement FIND and UNION :

- (i) The simplest implementation is to represent each set  $RT$  by a simple list, and to attach the variable  $TOP$  , which is defined in Section D, to each vertex of the reduced graph. In this implementation the instruction "FIND the vertex  $R$  such that  $PERT(R)$ " is replaced by the assignment  $R \leftarrow TOP(P)$  , which is performed in one step. The instruction "perform the UNION of the sets  $RT(R)$  and  $RT(R')$ " is

performed by concatenating the two lists which represent these sets and by updating the TOPs of the vertices which belong to these sets. Thus, the instruction UNION is at most performed in  $n_r$  steps. The total complexity of the operations on the sets  $RT$ , as determined by this implementation, is therefore  $O(n_r^2)$ .

(ii) Since the sets  $RT$  are disjoint, we can use the various disjoint-set union algorithms which are described in [6]. In particular, it can be proven that if we use the fast disjoint-set union algorithm, then the total complexity of the operations on the sets  $RT$  is  $O(m_r + n_r g(n_r))$ , where  $g(n)$  is the reciprocal Ackerman function, which can be considered as constant for all "practical" values of  $n$ .

The total complexity of the third stage. The above discussion implies that if we use the simplest implementations for the operations on ACTLIST and on the sets  $RT$ , then the total complexity of the third stage is  $O(n_r^2)$ . By using more sophisticated implementations we can achieve a complexity of  $O(m_r + n_r \lg \lg n_r)$ .

#### 4. THE FOURTH STAGE

##### A. General Description

In the fourth stage, the augmenting paths of the reduced graph which have been found during the third stage are retraced, and their sources in the original graph are restored (see the definitions of the second stage). The matching is improved by converting each free edge on these paths into a matched one and vice versa. The 2-exposed vertices at the ends of those augmenting paths, are removed from the graph.

The variable  $NEWMATE(R)$  . In order to record the restored augmenting paths and to perform the augmentation of the matching, we define for each vertex  $R$  of the original graph a variable  $NEWMATE(R)$  , as follows:

(i) In the beginning of the fourth stage, assign for each vertex  $R$  :  $NEWMATE(R) \leftarrow MATE(R)$  .

(ii) Let  $R$  be a vertex which is not 2-exposed, such that  $R$  is found to be lying on an augmenting path  $p$  which is restored during the fourth stage. Let  $RW$  be the free edge on  $p$  which is incident to  $R$  . Then assign:  $NEWMATE(R) \leftarrow W$  .

At the end of the fourth stage, for every vertex  $R$  (except for the 2-exposed vertices at the ends of the restored paths, which are removed from the graph)  $\text{NEWMATE}(R)$  is the new MATE of  $R$ . Therefore, for each vertex  $R$  of the graph we assign:  
 $\text{MATE}(R) \leftarrow \text{NEWMATE}(R)$ .

The procedure  $\text{PATH}(M,P)$ . In order to perform the assignments to the variables  $\text{NEWMATE}$  as required by rule (ii) of the definition, we define a (recursive) procedure  $\text{PATH}(M,P)$  as follows:

Let  $M$  and  $P$  be vertices of the original graph such that  $p(O \rightarrow M)$  is a head of  $p(O \rightarrow P)$ . For each free edge  $RW$  which is lying on  $p(M \rightarrow P)$ ,  $\text{PATH}(M,P)$  performs the assignments:  $\text{NEWMATE}(R) \leftarrow W$ ,  $\text{NEWMATE}(W) \leftarrow R$ .

[The formal definition of  $\text{PATH}(M,P)$  is described in Section B.]

The procedure  $\text{RPATH}(M,P)$ . Let  $M$  and  $P$  be vertices of the reduced graph such that  $\tilde{p}(i \rightarrow M)$  is a head of  $\tilde{p}(i \rightarrow P)$ . In order to get the source in the original graph of the segment  $\tilde{p}(M \rightarrow P)$ , one has to replace each free edge  $RW$  of  $\tilde{p}(M \rightarrow P)$  by the segment  $p(R \rightarrow X) \cdot p(W \leftarrow Y)$  where  $XY = \text{SOURCE}(RW)$  (see the algorithm of the second stage). Thus, in order to retrace the augmenting paths which have been found during the third stage, and to restore their sources in the original graph, we define a (recursive) procedure  $\text{RPATH}(M,P)$  as follows:

The vectors MATE , LEV , LINK . Three other vectors are required throughout the whole algorithm:  $MATE(-n:n)$  ,  $LEV(-n:n)$  and  $LINK(-n:n)$  :  $MATE(i)$  points to the vertex which is the mate of vertex  $i$  ,  $LEV(i)$  contains the level of vertex  $i$  and  $LINK(i)$  gives the link of vertex  $i$  as follows: If vertex  $i$  has a 2-link to edge number  $k$  , then  $LINK(i)=k$  . If vertex  $i$  has a 1-link to vertex  $i_1$  , then  $LINK(i)-3\cdot m$  is the number of the edge which goes from  $i_1$  to  $MATE(i)$  [thus,  $i$  has a 1-link if and only if  $LINK(i)>m$ ]. If vertex  $i$  has no link then  $LINK(i)=0$  .

The initialization of the algorithm. The algorithm is initialized by assigning for each  $i$  ,  $i=1\cdots n$  :  $MATE(i)=-i$  ,  $MATE(-i)=i$ . At this point we also build the list  $LIST(0)$  which contains all the vertices  $i$  , where  $i=1\cdots n$  .

The first stage. For the implementation of the first stage we need the following vectors (all these vectors can be released at the end of the first stage):

$FCHN(-n:n)$  ,  $LCHN(-n:n)$  ,  $NXCHN(-m:m)$  : These vectors describe the CHAINS :  $FCHN(i)$  is the number of the first edge (bridge) of  $CHAIN(i)$  ;  $LCHN(i)$  is the number of the last edge of  $CHAIN(i)$  ; (if  $CHAIN(i)$  is empty then  $FCHN(i)=0$  ,  $LCHN(i)=0$  ;)  $NXCHN(k)$  is the number of the edge which follows edge number  $k$  in  $CHAIN(E(-k))$  (if edge number  $k$  does not belong to  $CHAIN(E(-k))$  , or if it is the last edge in this CHAIN , then  $NXCHN(k)=0$  ) .

$TAIL(-n:n)$  ,  $NEXT(-n:n)$  : these vectors represent the set  $T(R)$

(see Implementation 1 in Section H of Chapter 1). If vertex number  $i$  belongs to the set  $T(i_1)$ , then  $TAIL(i)=i_1$  and  $NEXT(i)$  is the number of the vertex which follows  $i$  in  $T(i_1)$  (if  $i$  is the last vertex of  $T(i_1)$  then  $NEXT(i)=0$ ). The first vertex of  $T(i_1)$  is always  $MATE(i_1)$ , and thus it requires no special space.

For the implementation of subroutine MERGE we use Implementation 1 of Section H in Chapter 1.

The reduced graph. The reduced graph is represented by the vectors  $ER(-m:m)$ ,  $NXTER(-m:m)$  and  $ELIST(n)$  in the same manner the original graph is represented by the vectors  $E(-m:m)$ ,  $NXTE(-m:m)$  and  $V(n)$ . However, in order to preserve this representation, step 8 in the algorithm of the second stage must be slightly changed such that the insertion of an edge into the adjacency lists is performed as follows: Let  $i_1 i_2$  be an edge of the reduced graph such that  $i_1 < i_2$ : Both edges  $i_1 i_2$  and  $i_2 i_1$  are inserted into the adjacency lists of the reduced graph only when the edge  $i_1 i_2$  is searched (from  $i_1$  to  $i_2$ ). Obviously, this rule fulfils the requirement that when an edge  $YX$ , which is a source of the edge  $i_1 i_2$ , is searched in the original graph (step 8 in the algorithm of the second stage), then the edges  $i_1 i_2$  and  $i_2 i_1$  are already contained in the adjacency lists of the reduced graph if and only if the first vertex of the list  $ELIST(i_2)$  is  $i_1$  (see the discussion of Section D in Chapter 2).

Thus, in this implementation, the insertion of the  $k$ -th edge into the adjacency lists of the reduced graph (step 8 in the algorithm of the second stage) is performed as follows (we assume that an edge  $YX$

of the original graph is being searched, where  $i_1 = \text{BASE}(Y)$  ):

8. If  $\ell(X) + \ell(Y) + 1 \neq r$  then return to 7. Else, let  $i_2 \leftarrow \text{BASE}(X)$ .  
 If  $\ell(i_2) + (\text{MATE}(i_2)) > r$  return to 7. If  $i_1 \geq i_2$  then return to 7.  
 If  $\text{ELIST}(i_2) \neq 0$  and  $\text{ER}(-\text{ELIST}(i_2)) = i_1$  then return to 7.  
 Else, [record the edge  $i_1 i_2$  as the  $k$ -th edge of the reduced graph], assign:  $\text{ER}(k) \leftarrow i_1$ ,  $\text{ER}(-k) \leftarrow i_2$ ; [insert the edge  $i_2 i_1$  as the first edge of the adjacency list of vertex  $i_2$ ] assign:  $\text{NXTER}(-k) \leftarrow \text{ELIST}(i_2)$ ,  $\text{ELIST}(i_2) \leftarrow k$ ; [insert the edge  $i_1 i_2$  as the first edge of the adjacency list of vertex  $i_1$ ] assign:  $\text{NXTER}(k) \leftarrow \text{ELIST}(i_1)$ ,  $\text{ELIST}(i_1) \leftarrow k$ ; Assign:  $\text{SOURCE}(k) = YX$  and return to 7.

The vectors  $\text{ELIST}$  and  $\text{NXTER}$  are required for the performance of the second and the third stages of the algorithm. The vector  $\text{ER}$  is required also during the performance of the fourth stage.

The vector SOURCE. Since we apply the same representation to both the original graph and the reduced graph, it suffices to use a vector of  $m$  entries in order to implement the variables  $\text{SOURCE}$ . Thus, if  $\text{SOURCE}(k') = k$  (where  $k'$  and  $k$  are free edges of the reduced and the original graphs respectively), then  $\text{SOURCE}(-k') = -k$ . The vector  $\text{SOURCE}$  is required throughout the performance of the second, third and fourth stages of the algorithm.

The vectors VLIST, BASE, BSLIST. By Corollary 2.5, the 2-exposed vertices which belong to the reduced graph are exactly the mates of the 1-exposed vertices which belong to this graph. Since no free edges meet the 2-exposed vertices, it is convenient to save space



in the memory by ignoring the 2-exposed vertices throughout the performance of the second stage. Namely, the list `VLIST` of the vertices of the reduced graph does not contain the 2-exposed vertices, and thus it can be implemented by a vector `VLIST` of  $n$  entries (instead of a vector of  $2n$  entries): `VLIST(i)` contains the  $i$ -th vertex of the reduced graph (where the vertices of the reduced graph are listed in non-decreasing order of their levels). The vector `VLIST` is required throughout the second, third and fourth stages of the algorithm.

Since the 2-exposed vertices of the original graph are not searched in this implementation, no `BASEs` are defined for them. (Since no free edges meet these vertices, no problems arise during the performance of step 8 in the algorithm of the second stage). Thus, the vector `BASE` requires only  $n$  entries.

The list `BSLIST(Q)`, of all the vertices of the original graph whose `BASEs` are `Q`, is implemented by a vector `BSLIST(n)` as follows: For each vertex `Y` of the original graph, `BSLIST(Y)` points to the vertex which follows `Y` in the list `BSLIST(Q)`. (If `Y` is the last vertex of the list `BSLIST(Q)` then `BSLIST(Y)=0`). The first vertex of the list `BSLIST(Q)` is always the vertex `Q` itself, and thus it requires no special space. Since no `BASEs` are defined for the 2-exposed vertices, they require no entries of `BSLIST`, and thus `BSLIST` is implemented by a vector of length  $n$ .

Both vectors `BASE` and `BSLIST` can be released at the end of the second stage.

The third stage. Five new vectors are used during the third stage: LIFE(-n:n) , RLINK(-n:n) , TOP(-n:n) , RTNXT(-n:n) and ACTLIST(2n) .

LIFE(-n:n) and RLINK(-n:n) are the variables which are defined in Chapter 3. RLINK(i) represents the RLINK of vertex i in the same manner LINK(i) represents its LINK . Namely: if vertex i has a 2-link to the edge k then RLINK(i)=k ; if vertex i has a 1-link to the vertex  $i_1$  then RLINK(i)-3·m is the number of the edge which goes from  $i_1$  to MATE(i) . The vectors LIFE and RLINK are required also throughout the performance of the fourth stage.

TOP(-n:n) and RTNXT(-n:n) represent the sets RT in the same way as the vectors TAIL and NEXT represent the sets T(R) . Namely: if a vertex i belongs to the set  $RT(i_1)$  then  $TOP(i)=i_1$  and RTNXT(i) points to the vertex which follows i in  $RT(i_1)$  . The first vertex of  $RT(i_1)$  is always MATE( $i_1$ ) , and thus it requires no special space. The vectors TOP and RTNXT are released at the end of the third stage.

Since the vectors ELIST and NXTER are not required during the fourth stage, we use these vectors to handle the search of the edges throughout the third stage as follows: Assume that by the HLFS method we have to search an edge which emerges from the vertex P : We search the edge which ELIST(P) points to . However, we also assign a new value to ELIST(P) :  $ELIST(P) \leftarrow NXTER(k)$  , where k is the edge which is now being searched. This assignment assures that the next search from the vertex P would be along the edge NXTER(k) .

The list ACTLIST . The list ACTLIST is implemented by a vector ACTLIST(2n) , and by an index NACT which gives the number of the vertices which are contained in ACTLIST (see implementation (i) in Section G of Chapter 3). These vertices are stored in the locations  $i=1,2,\dots,NACT$  of the vector ACTLIST . When a new vertex is inserted into ACTLIST , we increase NACT by 1 ( $NACT \leftarrow NACT+1$ ) , and locate the new vertex in ACTLIST(NACT). In particular, when the  $i$ -th PART of the third stage is initialized, we insert the  $i$ -th vertex of VLIST into ACTLIST(1) and assign  $NACT+1$  . Thus, throughout the  $i$ -th PART of the third stage, ACTLIST contains only vertices whose LIFEs are  $i$  . In order to delete a vertex U from ACTLIST we decrease NACT by one ( $NACT \leftarrow NACT-1$ ) , and search ACTLIST : if the vertex U is found in the first NACT locations of ACTLIST , then it is replaced by the vertex which is located in ACTLIST(NACT+1) . During this search we also find a vertex of highest level in ACTLIST (as required by step 7 of the algorithm of the third stage). The vertex ACTLIST(2n) is released at the end of the third stage.

The subroutine SRCHLOOP . The subroutine SRCHLOOP is performed according to implementation (i) of Section G in Chapter 3: The instruction "FIND the vertex R such that  $P \in RT(R)$ " is simply performed by  $R \leftarrow TOP(P)$  . The instruction "UNION  $RT(Q) \leftarrow RT(Q) \cup RT(W)$ " is performed by a subroutine UNION(Q,W) , which updates the TOPs of all the vertices R in  $RT(W)$  and concatenates the list of  $RT(W)$  to the list of  $RT(Q)$  (by the assignment  $RTNXT(S) \leftarrow MATE(W)$  , where S is the last vertex of  $RT(Q)$  ).

The fourth stage. For the implementation of the fourth stage only one new vector has to be allocated:  $NEWMATE(n)$ . For vertex number  $i$  ( $i > 0$ ), the entry  $NEWMATE(i)$  gives the value of the variable  $NEWMATE(i)$  as defined in Section A of Chapter 4. Since this variable is useless for the 2-exposed vertices, no space is allocated for these vertices, and the length of the vector  $NEWMATE$  is therefore  $n$ .

Since in our implementation the 2-exposed vertices are not listed in  $VLIST$ , step 1 in the algorithm of the fourth stage is performed by searching the 1-exposed vertices which are contained in  $VLIST$  and referring to their mates.

Let  $P$  be a 2-exposed vertex. Since the matched edge  $(P, MATE(P))$  is not contained in the list  $E(-m:m)$  of the edges of the graph, there is no need to remove this edge from this list in step 5 of the algorithm of the fourth stage.

The subroutines  $RPATH$  and  $PATH$ . Each of the recursive subroutines  $RPATH$  and  $PATH$  may be activated recursively at most  $\lfloor \frac{n}{2} \rfloor$  times (no more than the number of free edges on one augmenting path). If in each call we store  $d$  variables, then the total space which is required for the implementation of each of these subroutines is  $(d+1) \cdot \lfloor \frac{n}{2} \rfloor$  words (one word is required per each call in order to store the return address). In a direct implementation of the algorithms of Chapter 4, 6 variables  $(A, P, R, W, X, Y)$  are stored per each call of subroutine  $RPATH(A, P)$ , and 4 variables  $(A, P, R, W)$  are stored per each call of subroutine  $PATH(A, P)$ . Thus,  $RPATH(A, P)$  requires  $3.5n$  words while  $PATH(A, P)$  needs only  $2.5n$  words.

The output. The output of the maximum matching which is achieved by the algorithm, is received at the end of the algorithm by printing the vector  $MATE(1:n)$  .

### B. Minimum-Space Implementation

In this section we outline an implementation of the algorithm which requires minimum space ( $4m+8n$  words, including  $2m+n$  words for storing the graph itself).

The representation of the graph. In this implementation, the graph is represented by two vectors,  $E(2m)$  and  $V(n)$  , which give the adjacency lists of the vertices as follows: The vertices which are adjacent to the vertex  $i$  are continuously listed in vertex  $E$  , where this list starts in entry  $V(i)$  of vector  $E$  (and ends in entry  $V(i+1)-1$  ).

The representation of the exposed vertices. In this implementation we do not apply the concept of 1-exposed and 2-exposed vertices. We use a vector  $MATE(n)$  in order to describe the mates of the vertices, where a vertex  $i$  of the graph is exposed if and only if  $MATE(i)=0$  . Thus, each time we reach a vertex  $i$  during the performance of the algorithm, we first have to check whether this vertex is exposed or not (namely, whether  $MATE(i)=0$  ), and to perform the appropriate operations in case the answer is positive. The levels, the links and all other variables are defined for the exposed vertices as if they are 2-exposed.

The levels and the links. The levels of the vertices are given by a vector  $LEV(n)$ .

The links are represented by two vectors  $IINK1(n)$  and  $LINK2(n)$  as follows: If a vertex  $i$  has no link, then  $LINK1(i)=0$ ,  $LINK2(i)=0$ . If a vertex  $i$  has a 1-link to the vertex  $i_1$ , then  $LINK1(i)=i_1$ ,  $LINK2(i)=0$ . If a vertex  $i$  has a 2-link to an edge  $i_1 i_2$ , then  $LINK1(i)=i_1$  while  $LINK2(i)$  gives the place of vertex  $i_2$  in the adjacency list of vertex  $i_1$  (namely:  $E(V(i_1)-1+LINK2(i))=i_2$ ).

The lists  $LIST(j)$ . In this implementation we do not use the lists  $LIST(j)$ . Each time we have to search through vertices of level  $j$  ( $j>0$ ) in the graph (e.g. steps 1, 9 and 10 in the algorithm of the first stage, and step 1 in the algorithm of the second stage), we simply search through all the vertices of the graph and look for vertices of level  $j$ . Each time we have to search through vertices of level 0, we search through all the vertices of the graph and look for vertices whose  $MATEs$  are 0.

Clearly all these searches can be performed in  $O(n^2)$  steps (per each phase of the algorithm).

The sets  $T(R)$  and  $RT(R)$ . These sets are represented only by the variables  $TAIL$  and  $TOP$  respectively (namely, during the first stage we allocate a vector  $TAIL(n)$  while during the third stage we allocate a vector  $TOP(n)$ ). When a search through the vertices of one of these sets is required (e.g. during the performance of subroutine  $MERGE$  in the first stage or during the performance of  $UNION$  in the

third stage) we simply search through all the vertices of the graph and check the values of their TAILS or TOPs . Since each of the subroutines MERGE and UNION is not called more than  $O(n)$  times throughout the performance of one phase, the complexity of these operations is  $O(n^2)$  (per one phase) .

The CHAINS . The CHAINS are implemented in the same way as in the Direct Implementation (see Section A of this chapter). Namely, three vectors are allocated: FCHN(n) , LCHN(n) and NXCHN(2m) .

The second and the third stages. The second and the third stages of the algorithm are performed as a one stage: First we perform the first part (FIND VERTICES) of the second stage: We do not refer in this implementation to the list VLIST but we define the variables BASE and construct the lists BSLIST . These lists are implemented by a vector BSLIST(n) in the same way as it is implemented in the Direct Implementation of Section A.

Then we start to perform the third stage, where the search of the edges (step 3 in the algorithm of the third stage) is done according to the second part (FIND EDGES) of the second stage. Namely: In order to search a free edge of the reduced graph which emerges from a vertex P of the reduced graph, we search a free edge which is incident to P in the original graph, and find (by step 8 in the algorithm of the second stage) whether this edge is a source of an edge in the reduced graph. If the answer is positive, we perform step 4 in the algorithm of the third stage. If all the free edges which are incident to P in

the original graph have already been searched, then we delete  $P$  from the list  $ACTLIST$  and we insert into this list the vertex  $Y$  which follows  $P$  in the list  $BSLIST(P)$ . The performance of the algorithm goes on as usual, but in the remainder of this stage we refer to  $Y$  as if its level is  $l(P)$  (the level of  $P$ ).

In this implementation, we do not define the variables  $SOURCE$ . The list  $ACTLIST$  and the variables  $LIFE$  are implemented in one vector,  $ACTLIFE(n)$ , as follows: The vertex  $i$  (of the original graph) is active if and only if  $ACTLIFE(i) > 0$ . The value of  $LIFE(i)$  is equal to the absolute value of  $ACTLIFE(i)$ .

The fourth stage. In the fourth stage we search through the exposed vertices of the graph. For each such a vertex  $P$  we check whether  $RLINK(P) = 0$ . If the answer is negative we call the subroutine  $RPATH(A, P)$ . Notice that in fact, the identity of vertex  $A$  is irrelevant to the performance of subroutine  $RPATH(A, P)$  (the check whether  $A = P$  can be replaced by another appropriate check), and thus the variables  $LIFE$  are not required during the fourth stage.

In order to find the source in the original graph of a free edge in the reduced graph we perform again the search through the free edges of the original graph as described in steps 6-8 of the algorithm of the second stage. For this purpose we need the vectors  $BASE$  and  $BSLIST$  during the fourth stage. Notice that after we find the source of an edge  $RW$  in the reduced graph, we do not need any more the levels of the vertices which are lying between  $R$  and  $N$  on the augmenting path of the original graph. Thus, we can use the space



of the vector LEV in order to store in it the variables NEWMATE .

It is not difficult to see that the subroutines RPATH(M,P) and PATH(M,P) can be implemented in such a way that only two words are required per each call of these subroutines: One word to store the vertex P and the other word to store the return address. Since each of these subroutines may be activated recursively at most  $\lfloor \frac{n}{2} \rfloor$  times, the space required for the implementation of each of these subroutines is n words of the memory.

The space requirements. The following vectors are required throughout the whole performance of the algorithm: E(2m) , V(n) , MATE(n) , LINK1(n) , LINK2(n) and LEV(n) (the latter is used during the fourth stage also for storing the variables NEWMATE ).

Apart from these vectors , the following vectors are required for the performance of the different stages:

For the first stage : TAIL(n) , FCHN(n) , LCHN(n) , NXCHN(2m) .

For the second and third stages : BASE(n) , BSLIST(n) , TOP(n) ,  
ACTLIFE(n) , RLINK1(n) , RLINK2(n) .

For the fourth stage : BASE(n) , BSLIST(n) , RLINK1(n) , RLINK2(n)  
and 2n words for the implementation of  
subroutines RPATH and PATH .

Thus, in this implementation the first stage of the algorithm requires  $4m+8n$  words of the memory, while the other stages requires  $2m+11n$  words.

APPENDIX A

In this appendix we prove part (b) of Theorem 1.4 in Chapter 1.

Theorem 1.4 (part (b))

Let  $R$  be a vertex such that at the end of the SEARCH part of the  $j$ -th substage  $R$  is not yet declared to be well-defined. Let  $ST$  be a bridge which can be assigned as  $LINK(R)$  such that this assignment defines for  $R$  a level  $\leq j$ . If  $LINK(R)=ST$  is illegal, then  $SET(R)$ .





Proof

Notations. Since  $ST$  can be assigned as  $LINK(R)$ ,  $ST$  must be a well-defined bridge. Denote by  $p_1$  the  $a_1$ -path  $p(O \rightarrow S)$  and let  $A$  be the 1-exposed vertex from which  $p_1$  starts. Denote by  $p_2$  the  $a_1$ -path  $p(O \rightarrow T)$  and let  $B$  be the 1-exposed vertex from which  $p_2$  starts.

Let  $P$  and  $Q$  be two vertices on  $p_1$ . By the notation  $P \xrightarrow{1} Q$  we mean: the segment of the  $a_1$ -path  $p_1$  from  $P$  to  $Q$ . Assume that  $Q$  is also lying on  $p_2$  and let  $R$  be another vertex on  $p_2$ . By the notation  $P \xrightarrow{1} Q \xrightarrow{2} R$  we mean: the  $a_1$ -path which is described by the concatenation of  $P \xrightarrow{1} Q$  and  $Q \xrightarrow{2} R$  (clearly vertex  $Q$  appears in  $P \xrightarrow{1} Q \xrightarrow{2} R$  only once). If  $P \xrightarrow{1} Q$  and  $Q \xrightarrow{2} R$  are disjoint (except for the vertex  $Q$ ) then  $P \xrightarrow{1} Q \xrightarrow{2} R$  is legal and we denote it by the notation:

$$\{\text{legal}\} = P \xrightarrow{1} Q \xrightarrow{2} R$$

The length of the alternating path  $P \xrightarrow{1} Q \xrightarrow{2} R$  is denoted by  $|P \xrightarrow{1} Q \xrightarrow{2} R|$ . Clearly:  $|P \xrightarrow{1} Q \xrightarrow{2} R| = |P \xrightarrow{1} Q| + |Q \xrightarrow{2} R|$ .

In the figures we denote a matched edge by  and a free edge by . The alternating path  $p_1$  is denoted by , while  $p_2$  is denoted by .

Introduction. Recall that by the inductive hypothesis and the algorithm, in the beginning of the  $j$ -th substage the vertices which are declared to be well-defined are exactly the vertices of levels less than  $j$ . Clearly no higher levels are defined to these vertices during the SEARCH part. On the other hand, by step 4 of the algorithm and by Theorem 1.1, the only vertices which are declared to be well-defined during the SEARCH part of the  $j$ -th substage are vertices of 1-link and level  $j$ . Therefore, if at the end of the SEARCH part of the  $j$ -th substage a vertex  $V$  is well-defined then either  $\ell(V) < j$  or  $V$  has a 1-link and  $\ell(V) = j$ .

Since at the end of the SEARCH part of the  $j$ -th substage  $ST$  can be assigned as  $\text{LINK}(R)$ , both  $S$  and  $T$  must be well-defined. Also, By Corollary 1.0 (in Section C of Chapter 1),  $T \in T(R)$ , and thus  $R$  is the last vertex on  $p(O \rightarrow T)$  which is not yet well-defined.

Since  $T$  is well-defined,  $\ell(T) \leq j$ . If  $\ell(T) = j$  then  $T$  must have a 1-link and thus  $\ell(\text{MATE}(T)) \geq \ell(T) = j$ , i.e.,  $\text{MATE}(T)$  is not yet well-defined. This implies that  $\text{MATE}(T) = R$ . In this case the al-path which is defined by  $\text{LINK}(R) = ST$  is  $p(O \rightarrow R) = p(O \rightarrow S) \cdot T \cdot R$ . This al-path is illegal if and only if the matched edge  $TR$  is lying on

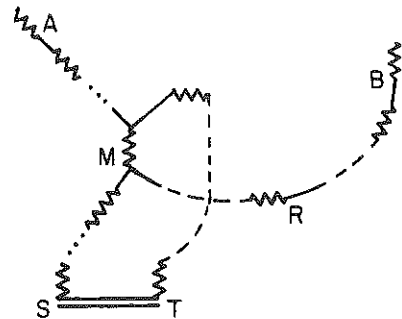
$p(0 \rightarrow S)$ . However, if  $T$  is  $f$ -lying on  $p(0 \rightarrow S)$ , then  $\ell(S) > \ell(T) = j$ , in contradiction to the fact that  $S$  is well-defined. If  $R$  is  $f$ -lying on  $p(0 \rightarrow S)$  then  $A \xrightarrow{1} R$  is a legal  $a_1$ -path of length less than  $j$ , and thus by the inductive hypothesis  $R$  must be well-defined. Thus,  $\ell(T) \neq j$  but  $\ell(T) < j$ . Since  $\text{LINK}(R) = ST$  defines for  $R$  a level not greater than  $j$ ,  $\ell(S) < j$ . Therefore  $\ell(S), \ell(T) < j$ .

Let  $\ell_1$  be the level which is defined for  $R$  by the assignment  $\text{LINK}(R) = ST$ . Since the  $a_1$ -path which is defined by  $\text{LINK}(R) = ST$  is illegal,  $p(0 \rightarrow S)$  and  $p(R \rightarrow T)$  [or in the present notation  $A \xrightarrow{1} S$  and  $R \xrightarrow{2} T$ ] must have a common vertex. Let  $M$  be the first vertex which is  $f$ -lying on  $p_1$  and is common to  $p(R \rightarrow T)$ .

(1) If  $M$  is not  $f$ -lying on  $p_2$  (Fig. A.1) then:

$$\{\text{legal}\} = |A \xrightarrow{1} M \xrightarrow{2} R| < |A \xrightarrow{1} M \xrightarrow{1} ST \xrightarrow{2} M \xrightarrow{2} R| = \ell_1 \leq j$$

$A \xrightarrow{1} M \xrightarrow{2} R$  is therefore a legal  $a_1$ -path of length less than  $j$  which is leading to  $R$ . Therefore, by the inductive hypothesis  $R$  must be well-defined - which is a contradiction. Thus,  $M$  is  $f$ -lying on  $p_2$ .



(2) Let  $M'$  be the first vertex which is  $f$ -lying on  $p(R \rightarrow T)$  and is common to  $p_1$ .

Fig. A.1

By a similar proof,  $M'$  is f-lying on  $p_1$ .

(3) Assume that  $M \neq M'$  (Fig. A.2).

If  $|A \xrightarrow{1} M| < |B \xrightarrow{2} M' \xrightarrow{2} M|$  then  $A \xrightarrow{1} M \xrightarrow{2} T$  is a legal alternating path which is leading to  $T$  and is shorter than  $\ell(T)$ . Since  $T$  is already well-defined this is a contradiction.

If  $|A \xrightarrow{1} M| \geq |B \xrightarrow{2} M' \xrightarrow{2} M|$ , then we have to consider two subcases:

(3.a) If  $B \xrightarrow{2} R$  and  $M' \xrightarrow{1} S$  are not disjoint, then let  $L$  be the first vertex which is f-lying on  $B \xrightarrow{2} R$  and is common to  $M' \xrightarrow{1} S$ .

If  $L$  is not f-lying on  $M' \xrightarrow{1} S$  then:

$$\{\text{legal}\} = |B \xrightarrow{2} L \xrightarrow{1} M' \xrightarrow{2} R| =$$

$$= |B \xrightarrow{2} L| + |L \xrightarrow{1} M'| + |M' \xrightarrow{2} R| <$$

$$< |A \xrightarrow{1} M| + |M' \xrightarrow{1} L| + |M' \xrightarrow{2} R| < |A \xrightarrow{1} M \xrightarrow{1} S T \xrightarrow{2} R| = \ell_1 \leq j$$

This inequality is in contradiction to the inductive hypothesis.

If  $L$  is f-lying on  $M' \xrightarrow{1} S$  then:

$$\{\text{legal}\} = |B \xrightarrow{2} L \xrightarrow{1} S| < |B \xrightarrow{2} L \xrightarrow{2} R \xrightarrow{2} M| + |M \xrightarrow{1} S| \leq |A \xrightarrow{1} M \xrightarrow{1} S| = \ell(S)$$

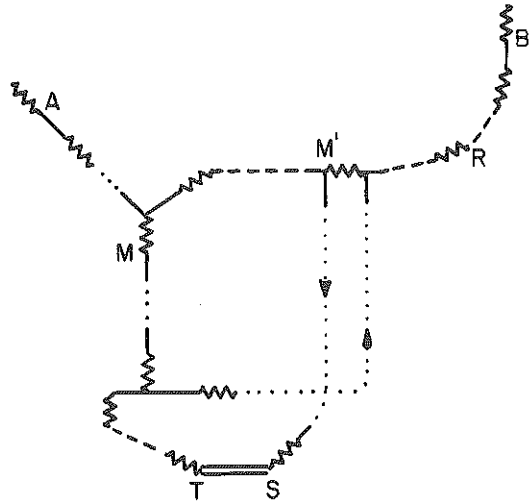


Fig. A.2

in contradiction to the fact that  $S$  is already well-defined.

(3.b) Thus,  $B \xrightarrow{2} R$  and  $M' \xrightarrow{1} S$  are disjoint. Therefore:

$$\{\text{legal}\} = |B \xrightarrow{2} M' \xrightarrow{1} S| < |B \xrightarrow{2} M' \xrightarrow{2} M \xrightarrow{1} S| \leq |A \xrightarrow{1} M \xrightarrow{1} S| = \ell(S)$$

in contradiction to the fact that  $S$  is already well-defined.

(4) Therefore  $M=M'$ .

The above proof also implies that:

$$(4.a) \quad |A \xrightarrow{2} M| = |B \xrightarrow{2} M|$$

(4.b)  $B \xrightarrow{2} R$  and  $M \xrightarrow{1} S$  are disjoint.

(5) Let  $N = \text{MATE}(M)$ . Since  $N$  is lying on  $R \xrightarrow{2} T$  then by the definition of  $\text{LINK}(R) = ST$  either  $N=R$  or  $N$  is well-defined. We consider separately the following two cases:

Case A.  $\ell(N) > \ell(S), \ell(T)$ . [ $\ell(N)$  is greater than both  $\ell(S)$  and  $\ell(T)$ ].

Case B. If  $\ell(N) > \ell(S)$  then  $\ell(N) \leq \ell(T)$  [i.e.,  $\ell(N)$  is not greater than both  $\ell(S)$  and  $\ell(T)$ ].

We shall prove that Case A implies that  $\text{SET}(R)$  while Case B leads to a contradiction.

Case A. Since  $\ell(S) < \ell(N)$  and  $\ell(S) < j$ , it follows by the inductive hypothesis that at the end of the  $\ell(S)$ -th substage  $N$  is not yet well-defined.

Consider the sequence of vertices on  $p_1$  which is defined as follows:

$$P_0 \leftarrow S ; \quad P_i = \begin{cases} P & \text{if LINK}(P_{i-1})=P \\ P & \text{if LINK}(P_{i-1})=PQ \end{cases}$$

Clearly, at the end of the  $\ell(S)$ -th substage all the vertices of this sequence are well-defined and for each  $i \geq 1$   $p(O \rightarrow P_i)$  is a head of  $p(O \rightarrow P_{i-1})$ . Thus, for each  $i$ ,  $p(O \rightarrow P_i)$  is a head of  $p(O \rightarrow S)$ .

Let  $k$  be the greatest index such that  $P_k$  is lying on  $M \xrightarrow{1} S$ . Assume that  $P_k \neq M$ . Clearly  $P_k$  cannot have a 1-link (otherwise  $\text{LINK}(P_k)$  would contradict the choice of  $P_k$ ). Thus, let  $\text{LINK}(P_k)=PQ$ . By the choice of  $P_k$ ,  $P$  is not lying on  $M \xrightarrow{1} S$ . It follows that  $M$  is lying on  $p(P_k \leftarrow Q)$  and therefore  $N$  is lying on  $p(P_k \leftarrow Q)$  too. But according to the 2-link definition, this implies that  $N$  is well-defined which is a contradiction.

Therefore  $P_k=M$  and thus  $p(O \rightarrow M)$  is a head of  $p(O \rightarrow S)$ .

By a similar proof,  $p(O \rightarrow M)$  is a head of  $p(O \rightarrow T)$ . On the other hand,  $ST$  can be assigned as  $\text{LINK}(R)$ , thus  $p(O \rightarrow \text{MATE}(R))$  must too be a head of  $p(O \rightarrow T)$ . Since (by its choice)  $M$  is lying on  $p(R \rightarrow T)$ , it follows that  $p(O \rightarrow \text{MATE}(R))$  is a head of  $p(O \rightarrow M)$ , and therefore is also a head of  $p(O \rightarrow S)$ .

Let  $W$  be the vertex such that in the beginning of the  $j$ -th substage  $\text{SET}(W)$ . Since during the SEARCH part of the algorithm the sets  $T(R')$  are not changed,  $S$  belongs to  $T(W)$  throughout the SEARCH part of the  $j$ -th substage. By its definition,  $W$  is the

vertex closest to  $S$  on  $p(0 \rightarrow S)$  such that  $W$  is not yet well-defined. Since  $p(0 \rightarrow \text{MATE}(R))$  is a head of  $p(0 \rightarrow S)$ ,  $R$  too is a vertex on  $p(0 \rightarrow S)$  which is not yet well-defined. Therefore, either  $W=R$ , or  $W$  is lying on  $p(\text{MATE}(R) \rightarrow S)$ .

Assume that  $W \neq R$ . Since  $ST$  can be assigned as  $\text{LINK}(R)$ , all vertices on  $p(\text{MATE}(R) \rightarrow T)$  are well-defined. In particular, all vertices on  $p(\text{MATE}(R) \rightarrow M)$  are well-defined, and thus  $W$  cannot be lying on this segment. Therefore,  $W$  is lying on  $p(M \rightarrow S)$ . Since  $\ell(T) < j$ , the bridge  $TS$  is searched and inserted into  $\text{CHAIN}(S)$  before the end of the  $\text{SEARCH}$  part of the  $j$ -th substage. If  $TS$  has failed the legality test, then at that point  $S$  and  $T$  were contained in the same set  $T(R')$ , and thus they must also belong to the same set at the end of the  $\text{SEARCH}$  part of the  $j$ -th substage, namely  $W=R$ . Since we assume that  $W \neq R$ , it follows that  $TS$  has not failed so far the legality test, and thus, at the end of the  $\text{SEARCH}$  part of the  $j$ -th substage  $TS$  belongs to the set

$\{T'S' \mid T'S' \in \text{CHAIN}(S'), S' \in T(W)\}$ . Therefore by Corollary 1.0,  $TS$  can be assigned as  $\text{LINK}(W)$ . This assignment defines for  $W$  a level

$\ell_2$  where:

$$\begin{aligned} \ell_2 &= |p(0 \rightarrow T) \cdot p(W \leftarrow S)| = |p(0 \rightarrow T)| + 1 + |p(W \leftarrow S)| = \\ &= |p(0 \rightarrow M)| + |p(M \rightarrow T)| + 1 + |p(W \leftarrow S)| < |p(0 \rightarrow M)| + |p(M \rightarrow T)| + |p(M \rightarrow S)| + 1 = \\ &= |p(0 \rightarrow S)| + |p(M \rightarrow T)| + 1 \leq |p(0 \rightarrow S)| + |p(R \rightarrow T)| + 1 = \ell_1 \leq j \end{aligned}$$

Thus  $\ell_2 < j$ , and therefore  $|p(0 \rightarrow T)| < j-1$ . This implies that the

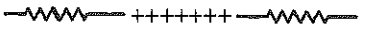
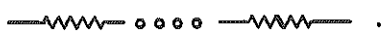


bridge  $TS$  has been searched and inserted into  $CHAIN(S)$  before the end of the  $SEARCH$  part of the  $j-1$  substage. Since  $S$  has been contained in  $T(W)$  before the end of the  $j-1$  substage, it follows that  $TS$  has belonged to the set  $\{T'S' \mid T'S' \in CHAIN(S'), S' \in T(W)\}$  already at the end of the  $j-1$  substage. However, this implies that at the end of the  $j-1$  substage  $W$  belongs to  $LIST(\ell_2)$ , and thus it must be well-defined - in contradiction to its choice.

Hence, the assumption that  $W \neq R$  is false, and therefore  $SET(R)$ .

Q.E.D.

Case B. In this case, if  $\ell(N) > \ell(S)$ , then  $\ell(N) \leq \ell(T)$ . Let  $U$  be the vertex which is defined as follows: If  $\ell(N) \leq \ell(S)$ , then  $U=S$ ; else  $U=T$  (i.e.,  $U$  is this vertex of  $S$  and  $T$  such that  $\ell(N) \leq \ell(U)$ ). Since  $\ell(S), \ell(T) < j$ , it follows that  $\ell(N) \leq \ell(U) < j$ . Thus, in the beginning of the  $j$ -th substage,  $N$  is already well-defined, and therefore  $N \neq R$ . Since  $M=MATE(N)$  is the first vertex which is  $f$ -lying on  $p(R \xrightarrow{2} T)$  and is common to  $p(A \xrightarrow{1} S)$ , it follows that  $R$  is not lying on  $p(A \xrightarrow{1} S)$ .

Denote by  $p_3$  the well-defined  $a_1$ -path  $p(O \rightarrow N)$ , and let  $C$  be the 1-exposed vertex from which  $p_3$  starts. In the figures this path is drawn as . Denote by  $p_4$  the segment  $p(M \rightarrow U)$ . This path is drawn as .

We show that the assumption of Case B implies the existence of an infinite sequence of vertices  $R_0, R_1, R_2, \dots$  such that  $R_0 = C$  and

and for  $i \geq 1$  the following conditions are satisfied:

- (i) All vertices  $R_i$  are  $f$ -lying on  $p_3$ , and in particular  $R_i$  is  $f$ -lying on  $R_{i-1} \xrightarrow{3} N$ ;
- (ii) All vertices  $R_{2i-1}$  are  $f$ -lying on  $p_2$  and in particular  $R_{2i+1}$  is  $f$ -lying on  $R_{2i-1} \xrightarrow{2} M$ ; All vertices  $R_{2i}$  are  $f$ -lying on  $p_1$  and in particular  $R_{2i+2}$  is  $f$ -lying on  $R_{2i} \xrightarrow{1} M$ ;
- (iii)  $|B \xrightarrow{2} R_{2i-1}| = |C \xrightarrow{3} R_{2i-1}|$ ;  $|A \xrightarrow{1} R_{2i}| = |C \xrightarrow{3} R_{2i}|$ ;
- (iv)  $B \xrightarrow{2} R_{2i-1}$  and  $R_{2i-1} \xrightarrow{3} N$  are disjoint;  $A \xrightarrow{2} R_{2i}$  and  $R_{2i} \xrightarrow{3} N$  are disjoint.

Clearly, the existence of such an infinite sequence leads to a contradiction which proves that Case B is impossible. This completes the proof of Theorem 1.4 (part (b)).

(6) If  $C \xrightarrow{3} N$  and  $R \xrightarrow{2} NM$  are disjoint, then:

$$\{\text{legal}\} = |C \xrightarrow{3} MN \xrightarrow{2} R| < |A \xrightarrow{1} M \xrightarrow{4} U \xrightarrow{2} MN \xrightarrow{2} R| = \ell_1 < j$$

$C \xrightarrow{3} MN \xrightarrow{2} R$  is therefore a legal  $al$ -path of length less than  $j$  which is leading to  $R$  - in contradiction to the inductive hypothesis. Therefore  $C \xrightarrow{3} N$  and  $R \xrightarrow{2} NM$  are not disjoint.

(7) Let  $R_1$  be the first vertex which is  $f$ -lying on  $R \xrightarrow{2} M$  and is common to  $C \xrightarrow{3} N$  (clearly the choice of  $R_1$  satisfies condition (ii)).

If  $R_1$  is not f-lying on  $C \xrightarrow{3} N$  (Fig. A.3) then:

$$\begin{aligned} \{lega\} &= |C \xrightarrow{3} R_1 \xrightarrow{2} R| < \\ &< |C \xrightarrow{3} MN \xrightarrow{2} R| < \\ &< |A \xrightarrow{1} M \xrightarrow{4} U \xrightarrow{2} MN \xrightarrow{2} R| = \\ &= \ell_1 \leq j \end{aligned}$$

in contradiction to the inductive hypothesis.

Fig. A.3

Therefore  $R_1$  is f-lying on  $C \xrightarrow{3} N$  and thus condition (i) is also satisfied by  $R_1$  (see Fig. A.4).

(8) Assume that  $|B \xrightarrow{2} R_1| < |C \xrightarrow{3} R_1|$  (see Fig. A.4):

(8.a) If  $B \xrightarrow{2} R$  and  $R_1 \xrightarrow{3} N$  are not disjoint, then let  $L$  be the first vertex which is f-lying on  $B \xrightarrow{2} R$  and is common to  $R_1 \xrightarrow{3} N$ .

If  $L$  is f-lying on  $R_1 \xrightarrow{3} N$  then:

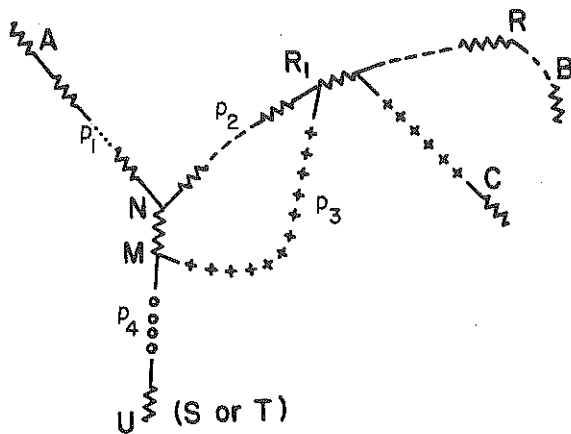
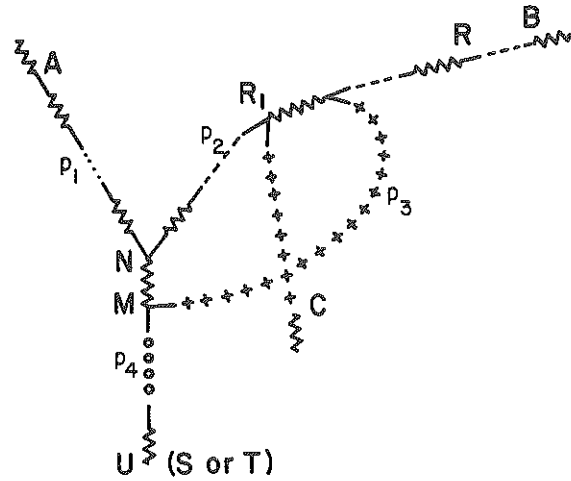


Fig. A.4

$$\begin{aligned} \{\text{legal}\} &= |B \xrightarrow{2} L \xrightarrow{3} N| < |B \xrightarrow{2} L \xrightarrow{2} R \xrightarrow{2} R_1 \xrightarrow{3} L \xrightarrow{3} N| \\ &\leq |C \xrightarrow{3} R_1 \xrightarrow{3} N| = \ell(N) \end{aligned}$$

which is a contradiction to the fact that  $N$  is already well-defined.

If  $L$  is not  $f$ -lying on  $R_1 \xrightarrow{3} N$  then:

$$\begin{aligned} \{\text{legal}\} &= |B \xrightarrow{2} L \xrightarrow{3} R_1 \xrightarrow{2} R| < |B \xrightarrow{2} L \xrightarrow{2} R \xrightarrow{2} R_1 \xrightarrow{3} L| \\ &\leq |C \xrightarrow{3} R_1 \xrightarrow{3} L| \leq |C \xrightarrow{3} N| = \ell(N) < j \end{aligned}$$

in contradiction to the inductive hypothesis.

(8.b) Therefore, if  $|B \xrightarrow{2} R_1| \leq |C \xrightarrow{3} R_1|$ , then  $B \xrightarrow{2} R$  and  $R_1 \xrightarrow{3} N$  are disjoint. By the definition of  $R_1$  it follows that in this case  $B \xrightarrow{2} R_1$  and  $R_1 \xrightarrow{3} N$  are disjoint, i.e., condition (iv) is satisfied by  $R_1$ .

(9) If  $|C \xrightarrow{3} R_1| > |B \xrightarrow{2} R_1|$  then:

$$\{\text{legal by (8.b)}\} = |B \xrightarrow{2} R_1 \xrightarrow{2} N| < |C \xrightarrow{3} R_1 \xrightarrow{3} N| = \ell(N)$$

which is a contradiction to the fact that  $N$  is already well-defined.

(10) Thus:  $|C \xrightarrow{3} R_1| \leq |B \xrightarrow{2} R_1|$ .

(10.a) If  $C \xrightarrow{3} R_1$  and  $R_1 \xrightarrow{2} M \xrightarrow{4} U$  are not disjoint, then let  $L$  be the first vertex which is  $f$ -lying on  $C \xrightarrow{3} R_1$  and is common to  $R_1 \xrightarrow{2} M \xrightarrow{4} U$ .

If  $L$  is  $f$ -lying on  $R_1 \xrightarrow{2} M \xrightarrow{4} U$  then:

$$\begin{aligned} \{\text{legal}\} &= |C \xrightarrow{3} L \xrightarrow{2} U| < |C \xrightarrow{3} L \xrightarrow{3} R_1 \xrightarrow{2} L \xrightarrow{2} U| < \\ &\leq |B \xrightarrow{2} R_1 \xrightarrow{2} M \xrightarrow{4} U| = \ell(U) \end{aligned}$$

which is a contradiction to the fact that  $U$  is already well-defined.

If  $L$  is not  $f$ -lying on  $R_1 \xrightarrow{2} M \xrightarrow{4} U$  then:

$$\begin{aligned} \{\text{legal}\} &= |C \xrightarrow{3} L \xrightarrow{4} R_1 \xrightarrow{2} R| = |C \xrightarrow{3} L| + |L \xrightarrow{4} R| < |C \xrightarrow{3} R_1| + |L \xrightarrow{4} R| < \\ &\leq |B \xrightarrow{2} R_1| + |L \xrightarrow{4} R| < |B \xrightarrow{2} M| + |L \xrightarrow{4} R| < \ell_1 \leq j \end{aligned}$$

which is a contradiction to the inductive hypothesis.

(10.b) Therefore  $C \xrightarrow{3} R_1$  and  $R_1 \xrightarrow{2} M \xrightarrow{4} U$  are disjoint.

(11) If  $|C \xrightarrow{3} R_1| < |B \xrightarrow{2} R_1|$  then:

$$\{\text{legal by (10.b)}\} = |C \xrightarrow{3} R_1 \xrightarrow{2} M \xrightarrow{4} U| < |B \xrightarrow{2} R_1 \xrightarrow{2} M \xrightarrow{4} U| = \ell(U)$$

which is a contradiction to the fact that  $U$  is already well-defined.

Therefore (by (10)),  $|C \xrightarrow{3} R_1| = |B \xrightarrow{2} R_1|$ , i.e., condition (iii) is satisfied by  $R_1$ . Also, by (8.b) it follows that condition (iv) is satisfied too.

(12) If  $A \xrightarrow{1} M$  and  $R_1 \xrightarrow{3} N$  are disjoint, then:

$$\begin{aligned} \{\text{legal}\} &= |A \xrightarrow{1} M \xrightarrow{3} R_1 \xrightarrow{2} R| = |A \xrightarrow{1} M| + |N \xrightarrow{3} R_1| + |R_1 \xrightarrow{2} R| = \\ &= |B \xrightarrow{2} M| + |N \xrightarrow{3} R_1| + |B \xrightarrow{2} R_1| - |B \xrightarrow{2} R| = \\ &= (|B \xrightarrow{2} M| - |B \xrightarrow{2} R|) + |N \xrightarrow{3} R_1| + |C \xrightarrow{3} R_1| = |M \xrightarrow{2} R| + |C \xrightarrow{3} N| = \\ &= \ell(N) + |M \xrightarrow{2} R| \leq \ell(U) + |M \xrightarrow{2} R| < \ell_1 \leq j \end{aligned}$$

which is a contradiction to the inductive hypothesis.

Therefore  $A \xrightarrow{1} M$  and  $R_1 \xrightarrow{3} N$  are not disjoint.

(13) Let  $R_2$  be the first vertex which is  $f$ -lying on  $R_1 \xrightarrow{3} N$  and is common to  $A \xrightarrow{1} M$  (thus, condition (i) is satisfied by  $R_2$ ).

If  $R_2$  is not  $f$ -lying on  $A \xrightarrow{1} M$  then:

$$\begin{aligned} \{\text{legal}\} &= |A \xrightarrow{1} R_2 \xrightarrow{3} R_1 \xrightarrow{2} R| < |A \xrightarrow{1} NM \xrightarrow{3} R_1 \xrightarrow{2} R| = \\ &= |A \xrightarrow{1} M| + |N \xrightarrow{3} R_1| + |R_1 \xrightarrow{2} R| = |B \xrightarrow{2} M| + |N \xrightarrow{3} R_1| + |B \xrightarrow{2} R_1| - |B \xrightarrow{2} R| \\ &= (|B \xrightarrow{2} M| - |B \xrightarrow{2} R|) + |N \xrightarrow{3} R_1| + |C \xrightarrow{3} R_1| = \\ &= |M \xrightarrow{2} R| + |C \xrightarrow{3} N| = \ell(N) + |M \xrightarrow{2} R| \leq \ell(U) + |M \xrightarrow{2} R| < \ell_1 \leq j \end{aligned}$$

which is a contradiction to the inductive hypothesis.

Therefore,  $R_2$  is  $f$ -lying on  $A \xrightarrow{1} M$  and condition (ii) is satisfied by  $R_2$  (see Fig. A.5).

(14) Assume that  $|A \xrightarrow{1} R_2| \leq |C \xrightarrow{3} R_2|$  (see Fig. A.5).

(14.a) If  $A \xrightarrow{1} R_2$  and  $R_2 \xrightarrow{3} N$  are not disjoint, then let  $L$  be the first vertex which is  $f$ -lying on  $A \xrightarrow{1} R_2$  and is common to  $R_2 \xrightarrow{3} N$ .

If  $L$  is not  $f$ -lying on  $R_2 \xrightarrow{3} N$  then:

$$\begin{aligned} \{\text{legal}\} &= |A \xrightarrow{1} L \xrightarrow{3} R_2 \xrightarrow{3} R_1 \xrightarrow{2} R| = |A \xrightarrow{1} L| + |L \xrightarrow{3} R_1| + |R_1 \xrightarrow{2} R| < \\ &< |A \xrightarrow{1} M| + |N \xrightarrow{3} R_1| + |R_1 \xrightarrow{2} R| = |B \xrightarrow{2} M| + |N \xrightarrow{3} R_1| + |R_1 \xrightarrow{2} R| = \end{aligned}$$

$$\begin{aligned}
 &= |B \xrightarrow{2} R_1| + |R_1 \xrightarrow{2} M| + |N \xrightarrow{3} R_1| + |R_1 \xrightarrow{2} R| = \\
 &= |C \xrightarrow{3} R_1| + |M \xrightarrow{2} R_1| + |N \xrightarrow{3} R_1| + |R_1 \xrightarrow{2} R| = |C \xrightarrow{3} N| + |M \xrightarrow{2} R| = \\
 &= \ell(N) + |M \xrightarrow{2} R| \leq \ell(U) + |M \xrightarrow{2} R| < \ell_1 \leq j
 \end{aligned}$$

which is a contradiction to the inductive hypothesis.

If  $L$  is f-lying on  $R_2 \xrightarrow{3} N$  then:

$$\begin{aligned}
 \{\text{legal}\} &= |A \xrightarrow{1} L \xrightarrow{3} N| < \\
 &< |A \xrightarrow{1} R_2 \xrightarrow{3} N| \leq \\
 &\leq |C \xrightarrow{3} R_2 \xrightarrow{3} N| = \ell(N)
 \end{aligned}$$

which is a contradiction to the fact that  $N$  is already well-defined.

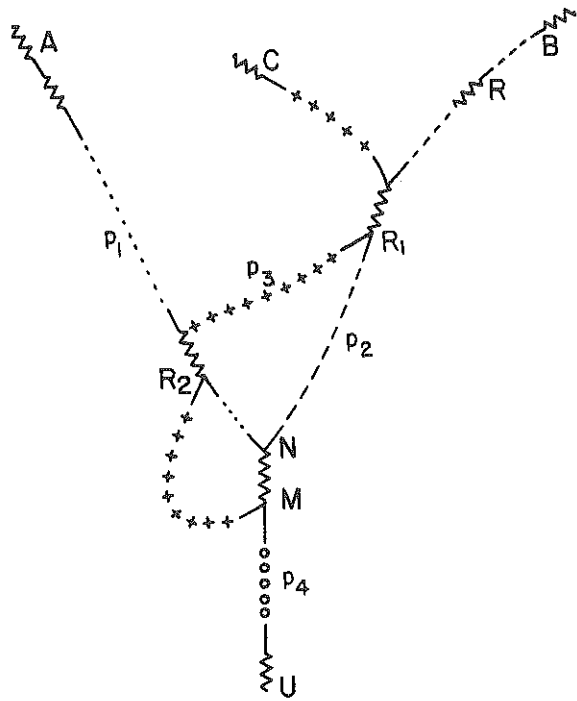


Fig. A.5

(14.b) Therefore,

if  $|A \xrightarrow{1} R_2| \leq |C \xrightarrow{3} R_2|$  then  $A \xrightarrow{1} R_2$  and  $R_2 \xrightarrow{3} N$  are disjoint, i.e., condition (iv) is satisfied by  $R_2$ .

(15) If  $|A \xrightarrow{1} R_2| < |C \xrightarrow{3} R_2|$  then:

$$\{\text{legal by (14.b)}\} = |A \xrightarrow{1} R_2 \xrightarrow{3} N| < |C \xrightarrow{3} R_2 \xrightarrow{3} N| = \ell(N)$$

in contradiction to the fact that  $N$  is already well-defined.

(16) Thus:  $|C \xrightarrow{3} R_2| \leq |A \xrightarrow{1} R_2|$ .

(16.a) If  $R_1 \xrightarrow{3} R_2$  and  $M \xrightarrow{4} U$  are not disjoint, then let  $L$  be the first vertex which is f-lying on  $R_1 \xrightarrow{3} R_2$  and is common to  $M \xrightarrow{4} U$ .

If  $L$  is not f-lying on  $M \xrightarrow{4} U$  then:

$$\begin{aligned}
 \{\text{legal}\} &= |A \xrightarrow{1} M \xrightarrow{4} L \xrightarrow{3} R_1 \xrightarrow{2} R| = |A \xrightarrow{1} M| + |M \xrightarrow{4} L| + |L \xrightarrow{3} R_1| + |R_1 \xrightarrow{2} R| \leq \\
 &\leq |B \xrightarrow{2} M| + |M \xrightarrow{4} U| + |L \xrightarrow{3} R_1| + |R_1 \xrightarrow{2} R| < \\
 &< |B \xrightarrow{2} R_1| + |R_1 \xrightarrow{2} M| + |M \xrightarrow{4} U| + |R_2 \xrightarrow{3} R_1| + |R_1 \xrightarrow{2} R| = \\
 &= |C \xrightarrow{3} R_1| + |M \xrightarrow{2} R| + |M \xrightarrow{4} U| + |R_2 \xrightarrow{3} R_1| = \\
 &= |C \xrightarrow{3} R_2| + |M \xrightarrow{4} U| + |M \xrightarrow{2} R| \leq |A \xrightarrow{1} R_2| + |M \xrightarrow{4} U| + |M \xrightarrow{2} R| < \\
 &< |A \xrightarrow{1} M \xrightarrow{4} U| + |M \xrightarrow{2} R| = \lambda(U) + |M \xrightarrow{2} R| < \lambda_1 \leq j
 \end{aligned}$$

in contradiction to the inductive hypothesis.

If  $L$  is f-lying on  $M \xrightarrow{4} U$  then:

$$\begin{aligned}
 \{\text{legal by (4.b)}\} &= |B \xrightarrow{2} R_1 \xrightarrow{3} L \xrightarrow{4} U| = |C \xrightarrow{3} R_1 \xrightarrow{3} L \xrightarrow{4} U| < \\
 &< |C \xrightarrow{3} L \xrightarrow{3} R_2 \xrightarrow{1} M \xrightarrow{4} L \xrightarrow{4} U| \leq |A \xrightarrow{1} R_2 \xrightarrow{1} M \xrightarrow{4} U| = \\
 &= \lambda(U)
 \end{aligned}$$

in contradiction to the fact that  $U$  is already well-defined.

(16.b) Therefore  $R_1 \xrightarrow{3} R_2$  and  $M \xrightarrow{4} U$  are disjoint. By (10.b) it follows that  $C \xrightarrow{3} R_2$  and  $M \xrightarrow{4} U$  are disjoint.



(17) If  $B \xrightarrow{2} R$  and  $R_2 \xrightarrow{1} M$  are not disjoint, then let  $L$  be the first vertex which is  $f$ -lying on  $B \xrightarrow{2} R$  and is common to  $R_2 \xrightarrow{1} M$ .

If  $L$  is not  $f$ -lying on  $R_2 \xrightarrow{1} M$  then:

$$\begin{aligned} \{\text{legal}\} &= |B \xrightarrow{2} L \xrightarrow{1} R_2 \xrightarrow{3} R_1 \xrightarrow{2} R| = |B \xrightarrow{2} L| + |L \xrightarrow{1} R_2| + |R_2 \xrightarrow{3} R_1| + |R_1 \xrightarrow{2} R| < \\ &< |B \xrightarrow{2} R| + |M \xrightarrow{1} R_2| + |R_1 \xrightarrow{3} R_2| + |R_1 \xrightarrow{2} R| = \\ &= |B \xrightarrow{2} R_1| + |R_1 \xrightarrow{3} R_2| + |M \xrightarrow{1} R_2| = |C \xrightarrow{3} R_1| + |R_1 \xrightarrow{3} R_2| + |M \xrightarrow{1} R_2| = \\ &= |C \xrightarrow{3} R_2| + |M \xrightarrow{1} R_2| \leq |A \xrightarrow{1} R_2| + |R_2 \xrightarrow{1} M| = \ell(M) < \ell_1 \leq j \end{aligned}$$

in contradiction to the inductive hypothesis.

If  $L$  is  $f$ -lying on  $R_2 \xrightarrow{1} M$  then:

$$\begin{aligned} \{\text{legal by (4.b)}\} &= |B \xrightarrow{2} L \xrightarrow{1} M \xrightarrow{4} U| = |B \xrightarrow{2} L| + |L \xrightarrow{1} M| + |M \xrightarrow{4} U| < \\ &< |B \xrightarrow{2} R_1| + |R_2 \xrightarrow{1} M| + |M \xrightarrow{4} U| = \\ &= |C \xrightarrow{3} R_1| + |R_2 \xrightarrow{1} M| + |M \xrightarrow{4} U| < |C \xrightarrow{3} R_2| + |R_2 \xrightarrow{1} M| + |M \xrightarrow{4} U| \\ &\leq |A \xrightarrow{1} R_2| + |R_2 \xrightarrow{1} M| + |M \xrightarrow{4} U| = \ell(U) \end{aligned}$$

in contradiction to the fact that  $U$  is already well-defined.

Therefore:  $B \xrightarrow{2} R$  and  $R_2 \xrightarrow{1} M$  are disjoint.

(18) If  $|C \xrightarrow{3} R_2| < |A \xrightarrow{1} R_2|$  then:

$$\begin{aligned} \{\text{legal by (4.b), (16.b) and (17)}\} &= |B \xrightarrow{2} R_1 \xrightarrow{3} R_2 \xrightarrow{1} M \xrightarrow{4} U| = \\ &= |C \xrightarrow{3} R_1 \xrightarrow{3} R_2 \xrightarrow{1} M \xrightarrow{4} U| < \end{aligned}$$

$$\langle |A \xrightarrow{1} R_2 \xrightarrow{1} M \xrightarrow{4} U| = \ell(U)$$

in contradiction to the fact that  $U$  is already well-defined.

Therefore, by (16)  $|C \xrightarrow{3} R_2| = |A \xrightarrow{1} R_2|$ , i.e., condition (iii) is satisfied by  $R_2$ . Also, by (14.b)  $A \xrightarrow{1} R_2$  and  $R_2 \xrightarrow{3} N$  are disjoint and therefore condition (iv) is satisfied too.

Thus, conditions (i)-(iv) hold for  $R_1$  and  $R_2$ . Hence, there exists in the graph a sequence  $R_0, R_1, R_2, \dots$  such that conditions (i)-(iv) are satisfied.

Assume now that the sequence is finite.

(19) Assume that the last vertex of the sequence  $R_0, R_1, R_2, \dots$  is  $R_{2k}$ .

If  $R_{2k-1} \xrightarrow{2} M$  and  $R_{2k} \xrightarrow{3} N$  are disjoint, then:

$$\begin{aligned} \{\text{legal}\} &= |A \xrightarrow{1} R_{2k} \xrightarrow{3} MN \xrightarrow{2} R| = |C \xrightarrow{3} R_{2k} \xrightarrow{3} MN \xrightarrow{2} R| = \ell(N) + |M \xrightarrow{2} R| \leq \\ &\leq \ell(U) + |M \xrightarrow{2} R| < \ell_1 \leq j \end{aligned}$$

in contradiction to the inductive hypothesis.

Thus,  $R_{2k-1} \xrightarrow{2} M$  and  $R_{2k} \xrightarrow{3} N$  are not disjoint.

(20) Let  $R_{2k+1}$  be the first vertex which is f-lying on  $R_{2k} \xrightarrow{3} N$  and is common to  $R_{2k-1} \xrightarrow{2} M$  (thus, condition (i) is satisfied by  $R_{2k+1}$ ).

If  $R_{2k+1}$  is not f-lying on  $R_{2k-1} \xrightarrow{2} M$  then:

$$\{\text{legal}\} = |A \xrightarrow{1} R_{2k} \xrightarrow{3} R_{2k+1} \xrightarrow{2} R_{2k-1} \xrightarrow{2} R| = |C \xrightarrow{3} R_{2k} \xrightarrow{3} R_{2k+1} \xrightarrow{2} R_{2k-1} \xrightarrow{2} R|$$

$$< |C \xrightarrow{3} N| + |M \xrightarrow{2} R| = \ell(N) + |M \xrightarrow{2} R| \leq \ell(U) + |M \xrightarrow{2} R| < \ell_1 \leq j$$

in contradiction to the inductive hypothesis.

Thus,  $R_{2k+1}$  is f-lying on  $R_{2k-1} \xrightarrow{2} M$  and thus condition (ii) is satisfied by  $R_{2k+1}$  (see Fig. A.6).

(21) If

$$|B \xrightarrow{2} R_{2k+1}| \leq |C \xrightarrow{3} R_{2k+1}|$$

then (see Fig. A.6):

(21.a) If

$$R_{2k-1} \xrightarrow{2} R_{2k+1} \text{ and}$$

$$R_{2k+1} \xrightarrow{3} N \text{ are not dis-}$$

joint, then let  $L$  be

the first vertex which is

f-lying on  $R_{2k-1} \xrightarrow{2} R_{2k+1}$

and is common to

$$R_{2k+1} \xrightarrow{3} N .$$

If  $L$  is not f-

lying on  $R_{2k+1} \xrightarrow{3} N$  then:

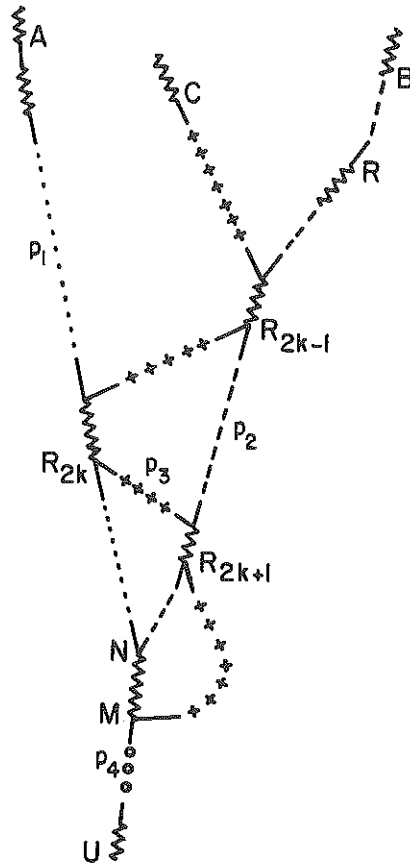


Fig. A.6

$$\begin{aligned}
 \{\text{legal}\} &= |A \xrightarrow{1} R_{2k} \xrightarrow{3} R_{2k+1} \xrightarrow{3} L \xrightarrow{2} R_{2k-1} \xrightarrow{2} R| = \\
 &= |A \xrightarrow{1} R_{2k}| + |R_{2k} \xrightarrow{3} L| + |L \xrightarrow{2} R| < |C \xrightarrow{3} R_{2k}| + |R_{2k} \xrightarrow{3} L| + |M \xrightarrow{2} R| < \\
 &< \ell(N) + |M \xrightarrow{2} R| \leq \ell(U) + |M \xrightarrow{2} R| < \ell_1 \leq j
 \end{aligned}$$

in contradiction to the inductive hypothesis.

If  $L$  is  $f$ -lying on  $R_{2k+1} \xrightarrow{3} N$  then:

$$\begin{aligned}
 \{\text{legal}\} &= |B \xrightarrow{2} R_{2k-1} \xrightarrow{2} L \xrightarrow{3} N| < |B \xrightarrow{2} R_{2k-1} \xrightarrow{2} L \xrightarrow{2} R_{2k+1} \xrightarrow{3} L \xrightarrow{3} N| = \\
 &= |B \xrightarrow{2} R_{2k+1}| + |R_{2k+1} \xrightarrow{3} N| \leq |C \xrightarrow{3} R_{2k+1}| + |R_{2k+1} \xrightarrow{3} N| = \ell(N)
 \end{aligned}$$

in contradiction to the fact that  $N$  is already well-defined.

(21.b) Therefore, if  $|B \xrightarrow{2} R_{2k+1}| \leq |C \xrightarrow{3} R_{2k+1}|$  then  $R_{2k-1} \xrightarrow{2} R_{2k+1}$  and  $R_{2k+1} \xrightarrow{3} N$  are disjoint. Since by condition (iv)  $B \xrightarrow{2} R_{2k-1}$  and  $R_{2k+1} \xrightarrow{3} N$  are disjoint, it follows that in this case  $B \xrightarrow{2} R_{2k+1}$  and  $R_{2k+1} \xrightarrow{3} N$  are disjoint, i.e., condition (iv) is satisfied by  $R_{2k+1}$ .

(22) If  $|B \xrightarrow{2} R_{2k+1}| < |C \xrightarrow{3} R_{2k+1}|$  then:

$$\{\text{legal by (21.b)}\} = |B \xrightarrow{2} R_{2k+1} \xrightarrow{3} N| < |C \xrightarrow{3} R_{2k+1} \xrightarrow{3} N| = \ell(N)$$

in contradiction to the fact that  $N$  is already well-defined.

(23) Thus:  $|C \xrightarrow{3} R_{2k+1}| \leq |B \xrightarrow{2} R_{2k+1}|$ .

(23.a) If  $R_{2k} \xrightarrow{3} R_{2k+1}$  and  $M \xrightarrow{4} U$  are not disjoint, then let  $L$  be the first vertex which is  $f$ -lying on  $R_{2k} \xrightarrow{3} R_{2k+1}$  and is

common to  $M \xrightarrow{4} U$ .

If  $L$  is not  $f$ -lying on  $M \xrightarrow{4} U$  then:

$$\begin{aligned} \{\text{legal}\} &= |A \xrightarrow{1} R_{2k} \xrightarrow{3} L \xrightarrow{4} M \xrightarrow{2} R| = |C \xrightarrow{3} R_{2k} \xrightarrow{3} L \xrightarrow{4} M \xrightarrow{2} R| < \\ &< |C \xrightarrow{3} R_{2k+1}| + |U \xrightarrow{4} M| + |M \xrightarrow{2} R| \leq |B \xrightarrow{2} R_{2k+1}| + |U \xrightarrow{4} M| + |M \xrightarrow{2} R| < \\ &< |B \xrightarrow{2} M| + |M \xrightarrow{4} U| + |M \xrightarrow{2} R| = \ell(U) + |M \xrightarrow{2} R| < \ell_1 \leq j \end{aligned}$$

in contradiction to the inductive hypothesis.

If  $L$  is  $f$ -lying of  $M \xrightarrow{4} U$  then:

$$\begin{aligned} \{\text{legal}\} &= |A \xrightarrow{1} R_{2k} \xrightarrow{3} L \xrightarrow{4} U| < |A \xrightarrow{1} R_{2k} \xrightarrow{3} L \xrightarrow{3} R_{2k+1} \xrightarrow{2} M \xrightarrow{4} L \xrightarrow{4} U| = \\ &= |C \xrightarrow{3} R_{2k} \xrightarrow{3} R_{2k+1} \xrightarrow{2} M \xrightarrow{4} U| \leq |B \xrightarrow{2} R_{2k+1} \xrightarrow{2} M \xrightarrow{4} U| = \ell(U) \end{aligned}$$

in contradiction to the fact that  $U$  is already well-defined.

(23.b) Therefore:  $R_{2k} \xrightarrow{3} R_{2k+1}$  and  $M \xrightarrow{4} U$  are disjoint.

(24) If  $|C \xrightarrow{3} R_{2k+1}| < |B \xrightarrow{2} R_{2k+1}|$  then:

$$\begin{aligned} \{\text{legal by (23.b)}\} &= |A \xrightarrow{1} R_{2k} \xrightarrow{3} R_{2k+1} \xrightarrow{2} M \xrightarrow{4} U| = \\ &= |C \xrightarrow{3} R_{2k} \xrightarrow{3} R_{2k+1} \xrightarrow{2} M \xrightarrow{4} U| < \\ &< |B \xrightarrow{2} R_{2k+1} \xrightarrow{2} M \xrightarrow{4} U| = \ell(U) \end{aligned}$$

in contradiction to the fact that  $U$  is already well-defined.

Therefore (by (23)),  $|C \xrightarrow{3} R_{2k+1}| = |B \xrightarrow{2} R_{2k+1}|$  and thus condition (iii) is satisfied by  $R_{2k+1}$ . Also, by (21.b), condition

(iv) is satisfied too.

Hence, conditions (i)-(iv) are satisfied by  $R_{2k+1}$ , and therefore  $R_{2k}$  cannot be the last vertex of the sequence  $R_0, R_1, R_2, \dots$ .

(25) Assume that the last vertex of the sequence  $R_0, R_1, R_2, \dots$  is  $R_{2k+1}$ .

If  $R_{2k} \xrightarrow{1} M$  and  $R_{2k+1} \xrightarrow{3} N$  are disjoint then:

$$\begin{aligned} \{legal\} &= |A \xrightarrow{1} R_{2k} \xrightarrow{1} NM \xrightarrow{3} R_{2k+1} \xrightarrow{2} R| = |A \xrightarrow{1} M| + |R_{2k+1} \xrightarrow{3} N| + |R \xrightarrow{2} R_{2k+1}| \\ &= |B \xrightarrow{2} M| + |R_{2k+1} \xrightarrow{3} N| + |R \xrightarrow{2} R_{2k+1}| = \\ &= |B \xrightarrow{2} R_{2k+1}| + |R_{2k+1} \xrightarrow{2} M| + |R_{2k+1} \xrightarrow{3} N| + |R \xrightarrow{2} R_{2k+1}| = \\ &= |C \xrightarrow{3} R_{2k+1}| + |R_{2k+1} \xrightarrow{3} N| + |R \xrightarrow{2} R_{2k+1}| + |R_{2k+1} \xrightarrow{2} M| = \\ &= \varrho(N) + |M \xrightarrow{2} R| < \varrho_1 \leq j \end{aligned}$$

in contradiction to the inductive hypothesis.

Thus,  $R_{2k} \xrightarrow{1} M$  and  $R_{2k+1} \xrightarrow{3} N$  are not disjoint.

(26) Let  $R_{2k+2}$  be the first vertex which is f-lying on  $R_{2k+1} \xrightarrow{3} N$  and is common to  $R_{2k} \xrightarrow{1} M$  (thus, condition (i) is satisfied by  $R_{2k+2}$ ).

If  $R_{2k+2}$  is not f-lying on  $R_{2k} \xrightarrow{1} M$  then:

$$\begin{aligned}
\{legal\} &= |A \xrightarrow{1} R_{2k} \xrightarrow{1} R_{2k+2} \xrightarrow{3} R_{2k+1} \xrightarrow{2} R| = \\
&= |A \xrightarrow{1} R_{2k+2}| + |R_{2k+1} \xrightarrow{3} R_{2k+2}| + |R_{2k+1} \xrightarrow{2} R| < \\
&< |A \xrightarrow{1} M| + |R_{2k+1} \xrightarrow{3} N| + |R_{2k+1} \xrightarrow{2} R| = \\
&= |B \xrightarrow{2} M| + |R_{2k+1} \xrightarrow{3} N| + |R_{2k+1} \xrightarrow{2} R| = \\
&= |B \xrightarrow{2} R| + |R \xrightarrow{2} M| + |R_{2k+1} \xrightarrow{3} N| + |R_{2k+1} \xrightarrow{2} R| = \\
&= |B \xrightarrow{2} R_{2k+1}| + |R_{2k+1} \xrightarrow{3} N| + |R \xrightarrow{2} M| = \\
&= |C \xrightarrow{3} R_{2k+1}| + |R_{2k+1} \xrightarrow{3} N| + |R \xrightarrow{2} M| = \ell(N) + |R \xrightarrow{2} M| \leq \\
&\leq \ell(U) + |R \xrightarrow{2} M| < \ell_1 \leq j
\end{aligned}$$

in contradiction to the inductive hypothesis.

Thus  $R_{2k+2}$  is f-lying on  $R_{2k} \xrightarrow{1} M$  and thus condition (ii) is satisfied by  $R_{2k+2}$  (see Fig. A.7).

(27) If  $|A \xrightarrow{1} R_{2k+2}| \leq |C \xrightarrow{3} R_{2k+2}|$  then (see Fig. A.7):

(27.a) If  $R_{2k} \xrightarrow{1} R_{2k+2}$  and  $R_{2k+2} \xrightarrow{3} N$  are not disjoint, then let  $L$  be the first vertex which is f-lying on  $R_{2k} \xrightarrow{1} R_{2k+2}$  and is common to  $R_{2k+2} \xrightarrow{3} N$ .

If  $L$  is not f-lying on  $R_{2k+2} \xrightarrow{3} N$  then:

$$\begin{aligned}
\{legal\} &= |A \xrightarrow{1} R_{2k} \xrightarrow{1} L \xrightarrow{3} R_{2k+2} \xrightarrow{3} R_{2k+1} \xrightarrow{2} R| = \\
&= |A \xrightarrow{1} L| + |R_{2k+1} \xrightarrow{3} L| + |R_{2k+1} \xrightarrow{2} R| < \\
&< |A \xrightarrow{1} M| + |R_{2k+1} \xrightarrow{3} N| + |R_{2k+1} \xrightarrow{2} R| =
\end{aligned}$$

$$\begin{aligned}
 &= |B \xrightarrow{2} M| + |R_{2k+1} \xrightarrow{3} N| + |R_{2k+1} \xrightarrow{2} R| = \\
 &= |B \xrightarrow{2} R| + |R \xrightarrow{2} M| + |R_{2k+1} \xrightarrow{3} N| + |R_{2k+1} \xrightarrow{2} R| = \\
 &= |B \xrightarrow{2} R_{2k+1}| + |R_{2k+1} \xrightarrow{3} N| + |R \xrightarrow{2} M| = \\
 &= |C \xrightarrow{3} R_{2k+1}| + |R_{2k+1} \xrightarrow{3} N| + |M \xrightarrow{2} R| = \ell(N) + |M \xrightarrow{2} R| \leq \\
 &\leq \ell(U) + |M \xrightarrow{2} R| < \ell_1 \leq j
 \end{aligned}$$

in contradiction to the inductive hypothesis.

If  $L$  is  $f$ -lying on

$R_{2k+2} \xrightarrow{3} N$  then:

$$\begin{aligned}
 \{\text{legal}\} &= |A \xrightarrow{1} R_{2k} \xrightarrow{1} L \xrightarrow{3} N| < \\
 &< |A \xrightarrow{1} R_{2k} \xrightarrow{1} L \xrightarrow{1} R_{2k+2} \xrightarrow{3} L \xrightarrow{3} N| \\
 &\leq |C \xrightarrow{3} R_{2k+2} \xrightarrow{3} L \xrightarrow{3} N| = \\
 &= \ell(N)
 \end{aligned}$$

in contradiction to the fact that  $N$  is already well-defined.

(27.b) Therefore, if

$|A \xrightarrow{1} R_{2k+2}| \leq |C \xrightarrow{3} R_{2k+2}|$   
 then  $R_{2k} \xrightarrow{1} R_{2k+2}$  and  $R_{2k+2} \xrightarrow{3} N$  are disjoint.  
 Since by condition (iv)

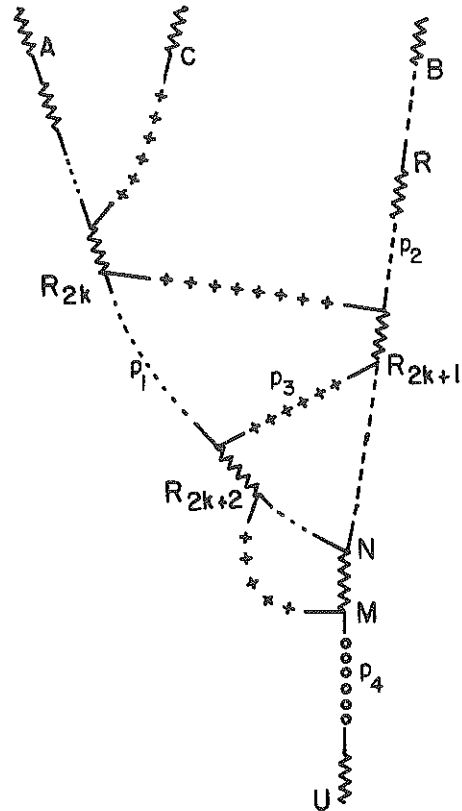


Fig. A.7



$A \xrightarrow{1} R_{2k}$  and  $R_{2k+2} \xrightarrow{3} N$  are disjoint, it follows that in this case  $A \xrightarrow{1} R_{2k+2}$  and  $R_{2k+2} \xrightarrow{3} N$  are disjoint, i.e., condition (iv) is satisfied by  $R_{2k+2}$ .

(28) If  $|A \xrightarrow{1} R_{2k+2}| < |C \xrightarrow{3} R_{2k+2}|$  then:

$$\{\text{legal by (27.b)}\} = |A \xrightarrow{1} R_{2k+2} \xrightarrow{3} N| < |C \xrightarrow{3} R_{2k+2} \xrightarrow{3} N| = \ell(N)$$

in contradiction to the fact that  $N$  is already well-defined.

(29) Thus:  $|C \xrightarrow{3} R_{2k+2}| \leq |A \xrightarrow{1} R_{2k+2}|$ .

(29.a) If  $R_{2k+1} \xrightarrow{3} R_{2k+2}$  and  $M \xrightarrow{4} U$  are not disjoint, then let  $L$  be the first vertex which is  $f$ -lying on  $R_{2k+1} \xrightarrow{3} R_{2k+2}$  and is common to  $M \xrightarrow{4} U$ .

If  $L$  is not  $f$ -lying on  $M \xrightarrow{4} U$  then:

$$\begin{aligned} \{\text{legal}\} &= |A \xrightarrow{1} R_{2k} \xrightarrow{1} R_{2k+2} \xrightarrow{1} M \xrightarrow{4} L \xrightarrow{3} R_{2k+1} \xrightarrow{2} R| < \\ &< |A \xrightarrow{1} M| + |M \xrightarrow{4} U| + |R_{2k+2} \xrightarrow{3} R_{2k+1}| + |R_{2k+1} \xrightarrow{2} R| = \\ &= |B \xrightarrow{2} M| + |M \xrightarrow{4} U| + |R_{2k+2} \xrightarrow{3} R_{2k+1}| + |R_{2k+1} \xrightarrow{2} R| = \\ &= |B \xrightarrow{2} R_{2k+1}| + |R_{2k+1} \xrightarrow{2} M| + |M \xrightarrow{4} U| + |R_{2k+2} \xrightarrow{3} R_{2k+1}| + |R_{2k+1} \xrightarrow{2} R| \\ &= |C \xrightarrow{3} R_{2k+1}| + |R_{2k+1} \xrightarrow{3} R_{2k+2}| + |M \xrightarrow{4} U| + |R \xrightarrow{2} M| \leq \\ &\leq |A \xrightarrow{1} R_{2k+2}| + |M \xrightarrow{4} U| + |R \xrightarrow{2} M| < |A \xrightarrow{1} M| + |M \xrightarrow{4} U| + |M \xrightarrow{2} R| = \\ &= \ell(U) + |M \xrightarrow{2} R| < \ell_1 \leq j \end{aligned}$$

in contradiction to the inductive hypothesis.

If  $L$  is  $f$ -lying on  $M \xrightarrow{4} U$  then:

$$\begin{aligned}
 \{\text{legal by (4.b)}\} &= |B \xrightarrow{2} R_{2k+1} \xrightarrow{3} L \xrightarrow{4} U| < \\
 &< |B \xrightarrow{2} R_{2k+1} \xrightarrow{3} L \xrightarrow{3} R_{2k+2} \xrightarrow{1} M \xrightarrow{4} L \xrightarrow{4} U| = \\
 &= |C \xrightarrow{3} R_{2k+1} \xrightarrow{3} R_{2k+2} \xrightarrow{1} M \xrightarrow{4} U| \leq \\
 &\leq |A \xrightarrow{1} R_{2k+2} \xrightarrow{1} M \xrightarrow{4} U| = \ell(U)
 \end{aligned}$$

in contradiction to the fact that  $U$  is already well-defined.

(29.b) Therefore:  $R_{2k+1} \xrightarrow{3} R_{2k+2}$  and  $M \xrightarrow{4} U$  are disjoint.

(30) If  $|C \xrightarrow{3} R_{2k+2}| < |A \xrightarrow{1} R_{2k+2}|$  then:

$$\begin{aligned}
 \{\text{legal by (29.b), (17), (4.b)}\} &= |B \xrightarrow{2} R_{2k+1} \xrightarrow{3} R_{2k+2} \xrightarrow{1} M \xrightarrow{4} U| = \\
 &= |C \xrightarrow{3} R_{2k+1} \xrightarrow{3} R_{2k+2} \xrightarrow{1} M \xrightarrow{4} U| < \\
 &< |A \xrightarrow{1} R_{2k+2} \xrightarrow{1} M \xrightarrow{4} U| = \ell(U)
 \end{aligned}$$

in contradiction to the fact that  $U$  is already well-defined.

Therefore (by (29))  $|C \xrightarrow{3} R_{2k+2}| = |A \xrightarrow{1} R_{2k+2}|$ , and thus condition (iii) is satisfied by  $R_{2k+2}$ . Also, by (27.b), condition (iv) is satisfied too.

Hence, conditions (i)-(iv) are satisfied by  $R_{2k+2}$ , and therefore  $R_{2k+1}$  cannot be the last vertex of the sequence  $R_0, R_1, R_2, \dots$ .

It follows by (24) and (30) that the sequence  $R_0, R_1, R_2, \dots$

is infinite. Clearly, this leads to a contradiction. Therefore Case B is impossible. Since Case A implies that  $\text{SET}(R)$ , the theorem is proved.

Q.E.D.

APPENDIX B

In this appendix we present an implementation of the subroutine MERGE which is based on a special structure of the sets  $T(R)$ . The algorithm of this implementation (Implementation 4), is similar to the algorithm of Implementation 2 in Section H of Chapter 1. However, the special structure of the sets  $T(R)$  enables the performance of subroutine MERGE throughout the whole first stage in no more than  $O(m + n^{1+\epsilon})$  steps, where  $\epsilon$  is an arbitrary positive value. Thus, by using Implementation 4 we determine a complexity of  $O(m\sqrt{n} + n^{1.5+\epsilon})$  for the whole algorithm.

The structure of the sets  $T(R)$ . Each set  $T(R)$  is represented by the following rooted tree: Let  $K$  be a constant integer number (whose value will be discussed later). The depth of the tree which represents  $T(R)$  is  $K+1$  (where the depth of the root is defined to be 0 and the depth of the leaves is  $K+1$ ). The root has  $n^{1-K\epsilon}$  sons, while each of the other (inner) vertices of the tree has  $n^\epsilon$  sons. Thus, the number of vertices whose depth is  $i$  is  $n^{1-(K+1-i)\epsilon}$ . In particular there exist  $n$  leaves of the tree. These leaves are numbered from 0 (the most "left" leaf) to  $n-1$  (the most "right" leaf).

All the leaves  $i$ , where  $(j-1) \cdot n^\epsilon \leq i < j \cdot n^\epsilon$  for some  $j$ , have a common father in the tree (the depth of this father is  $K$ ). We denote this set of leaves by  $FAMILY(R, j)$ .

For each vertex  $T$  of the graph we define the variable  $VALUE(T)$  in the same way as it is defined for Implementation 2 in Section H of Chapter 1. Namely:

If  $T$  is not well-defined yet, or if  $CHAIN(T)$  is empty then  $VALUE(T)=n$ . Else,  $VALUE(T)=\ell(ST)$  where  $ST$  is the first bridge of  $CHAIN(T)$ .

All the vertices of the graph which belong to  $T(R)$  and whose  $VALUE$  is  $j$  are contained and listed in the following chain: The first vertex of this chain is stored in the  $j$ -th leaf of the tree. To each vertex  $T$  of the chain we attach two variables:  $PRE(T)$ , which points to the vertex which precedes  $T$  in the chain, and  $NEXT(T)$  which points to the vertex which follows  $T$  in the chain (if  $T$  is the first vertex in the chain then  $PRE(T)=0$ ; if it is the last vertex in the chain then  $NEXT(T)=0$ ).

To each vertex  $D$  of the tree which represents the set  $T(R)$ , we attach a variable  $NUM(D)$  as follows:

(i) If  $D$  is a leaf of the tree, then  $NUM(D)=1$  if  $D$  contains a vertex of the graph and  $NUM(D)=0$  if it does not (namely: if  $D$  is the  $j$ -th leaf of the tree, then  $NUM(D)=0$  if and only if the set  $T(R)$  does not contain any vertex of the graph whose  $VALUE$  is  $j$ ).

(ii) If  $D$  is an inner vertex of the tree then  $NUM(D)$  is the number of the leaves which are descendants of  $D$  and their  $NUMs$  are not 0.

(iii) If  $D$  is the root of the tree then  $NUM(D)$  is the number of vertices of the graph which are contained in the set  $T(R)$  which is represented by the tree.

In addition to these variables we also attach to each leaf of the tree a pointer  $ROOT$  which points to the root of the tree. To the root itself we attach a variable  $NAME$ , which contains the name  $R$  of the set  $T(R)$  which is represented by the tree.

The algorithm of Implementation 4. According to the definition of subroutine  $MERGE(R_2, R)$ , the following operation has to be performed:  $T(R_2) \leftarrow T(R_2) \cup T(R)$ . In this implementation we perform this union in a way which assures that the following condition is satisfied:

Let  $\ell_1$  be the smallest index such that after the performance of  $T(R_2) \leftarrow T(R_2) \cup T(R)$ ,  $NUM(\ell_1) \neq 0$ . Then there exists in the set  $T(R_2)$  a vertex  $T$  such that  $CHAIN(T)$  contains a bridge of level  $\ell_1$  which presently passes the legality test.

[We shall describe later the details of the operation "perform the union  $T(R_2) \leftarrow T(R_2) \cup T(R)$ ".]

This property of the instruction UNION is utilized in the following algorithm of subroutine  $MERGE$  (we assume that  $MERGE$  is activated during the  $UPDATE$  part of the  $j_0$ -th substage):

1. Perform the UNION:  $T(R_2) \leftarrow T(R_2) \cup T(R)$ .
2. Delete  $R_2$  from  $LIST(\ell(R_2))$ .
3. Find the MINIMUM index  $j$  such that  $FAMILY(R_2, j) \neq \emptyset$ : If

- $(j-1) \cdot n^E \leq j_0$  then go to 4. Else, assign:  $\ell(R_2) \leftarrow \lfloor (j-1) \cdot n^E \rfloor - \frac{1}{2}$ ,  $\text{LINK}(R_2) = -1$  and go to 8.
4. Search the set  $\text{FAMILY}(R_2, j)$  and find the first leaf (leaf of smallest index)  $\ell_1$  such that  $\text{NUM}(\ell_1) = 1$ . Let  $T$  be the vertex of the graph which is contained in this leaf.
  5. If  $\text{CHAIN}(T) = \emptyset$  then go to 6. Else let  $ST$  be the first bridge of  $\text{CHAIN}(T)$ . FIND the vertex  $R_1$  of the graph such that  $\text{SET}(R_1)$ . If  $R_1 \neq R_2$  [ $ST$  passes the legality test] then go to 6. Else, delete  $ST$  from  $\text{CHAIN}(T)$ , and repeat 5.
  6. Update  $\text{VALUE}(T)$ . Let  $\ell_2 \leftarrow \text{VALUE}(T)$ . If  $\ell_1 = \ell_2$  then go to 7. Else, DELETE\* vertex  $T$  from the chain of the  $\ell_1$ -th leaf and INSERT\* it into the chain of the  $\ell_2$ -th leaf. Let  $T$  be the (new) vertex of the graph which is contained in the  $\ell_1$ -th leaf\*\*. Go to 5.
  7. Assign:  $\ell(R_2) \leftarrow \ell_1$ ,  $\text{LINK}(R_2) \leftarrow$  (if  $\text{CHAIN}(T) = \emptyset$  then 0; else  $ST$ ).
  8. Insert  $R_2$  into  $\text{LIST}(\ell(R_2))$  and return.

According to this algorithm there are two ways through which the level and the LINK of  $R_2$  can be assigned: If  $\ell(R_2)$  and  $\text{LINK}(R_2)$  are assigned in step 7 of the algorithm then obviously  $\ell_1 = \ell(R_2)$  is the minimum level which can presently be defined for  $R_2$  such that  $\text{LINK}(R_2) = ST$  passes the legality test. However, if  $\ell(R_2)$  and  $\text{LINK}(R_2)$

---

\* Clearly these operations require the updating of the appropriate variables NUM, PRE and NEXT.

\*\* By the property of the instruction UNION, such a vertex must exist.

are assigned in step 3, then both the level and the LINK of  $R_2$  are dummy. Clearly, at present there exists no legal LINK which can define for  $R_2$  a legal level less than  $\lceil (j-1) \cdot n^\epsilon \rceil^{-1/2}$ . On the other hand, the presently smallest legal level which can be defined for  $R_2$  is greater than  $\lceil (j-1) \cdot n^\epsilon \rceil^{-1/2}$ . If we shall find for  $R_2$  a legal level smaller than  $\lceil (j-1) \cdot n^\epsilon \rceil^{-1/2}$  before the beginning of the  $\lceil (j-1) \cdot n^\epsilon \rceil$ -th substage, then we would reassign the LINK of  $R_2$  and would amend the level of  $R_2$ —as it is done in the usual algorithm. If we reach the  $\lceil (j-1) \cdot n^\epsilon \rceil$ -th substage and the level of  $R_2$  is still  $\lceil (j-1) \cdot n^\epsilon \rceil^{-1/2}$ , then we interrupt the performance of the main algorithm of the first stage, and perform steps 2-8 of this algorithm. Thus, each time when we proceed from the  $j$ -th substage to the  $(j+1)$ -th substage, we first have to check whether there exists a dummy LIST of vertices whose levels are  $j+1/2$  [clearly these checks do not increase the order of the main algorithm].

The complexity of the operations on the tree. Apart from the operations which are performed on the tree of the set  $T(R_2)$ , all the other operations which are performed by this algorithm require either a constant number of steps per each activation of the subroutine MERGE (steps 2, 3, 7 and 8), or a constant number of steps per each bridge of the graph which fails the legality test (step 5, 6). Since subroutine MERGE may be activated at most  $n$  times during the first stage, the total complexity of these operations is  $O(m+n)$ .

Step 4 of the algorithm ("search FAMILY( $R_2, j$ )") is performed in  $n^\epsilon$  steps per each activation of MERGE, and thus its total complexity throughout the first stage is  $O(n^{1+\epsilon})$ .



The instructions "DELETE vertex  $T$  from the chain of the  $\ell_1$ -th leaf" and "INSERT vertex  $T$  into the chain of the  $\ell_2$ -th leaf" require  $K$  steps in order to search the tree from the  $\ell_1$ -th [ $\ell_2$ -th] leaf to the root and to update the NUMs of the  $K$  ancestors of the  $\ell_1$ -th [ $\ell_2$ -th] leaf. Since  $K$  is a constant integer, each of these instructions requires a constant number of steps. Each of the instructions DELETE and INSERT may be performed at most once per each bridge of the graph which fails the legality test. Thus, the total complexity of those instructions throughout the first stage is  $O(m)$ .

The instruction "Find the MINIMUM index  $j$  such that  $\text{FAMILY}(R_2, j) \neq \emptyset$ " is performed as follows: We search the  $n^{1-K\epsilon}$  vertices of depth 1 in the tree and find the first (the smallest, the most left) vertex whose NUM is not 0. Then we search the  $n^\epsilon$  sons of that vertex and find the first one whose NUM is not 0. Again, we search the  $n^\epsilon$  sons of this vertex to find its smallest son whose NUM is not 0, and so on. The number of steps we perform during this search on the vertices of the tree is therefore  $n^{1-K\epsilon} + K \cdot n^\epsilon$ . Since during the first stage MERGE is activated  $n$  times, the total complexity of the instruction "find MINIMUM" throughout the first stage is therefore  $O(n^{2-K\epsilon} + n^{1+\epsilon})$ .

The instruction  $\text{UNION } T(R_2) \leftarrow T(R_2) \cup T(R)$ . The instruction  $\text{UNION } T(R_2) \leftarrow T(R_2) \cup T(R)$  is performed as follows: We denote by  $T(R_g)$  this set among  $T(R_2)$  and  $T(R)$  which contains more vertices of the graph, and by  $T(R_s)$  the other set (thus, if  $\text{NUM}(\text{root}(R_2)) \geq \text{NUM}(\text{root}(R))$  then  $T(R_g) = T(R_2)$  and  $T(R_s) = T(R)$ ; else  $T(R_g) = T(R)$  and  $T(R_s) = T(R_2)$ ). In this implementation we transfer the vertices which are contained in

the tree of  $T(R_S)$  into the tree which represents  $T(R_G)$ . Thus, throughout the whole performance of the first stage each vertex of the graph is transferred at most  $\lg n$  times.

The transferring of each vertex is performed as follows: With the help of the instruction "Find MINIMUM" we find the minimum index  $j$  such that the set  $FAMILY(R_S, j)$  is not empty [this is done in  $O(n^{1-K\epsilon} + K \cdot n^\epsilon)$  steps]; Then, we search  $FAMILY(R_S, j)$  and find a leaf  $\ell_1$  whose NUM is not 0 [this search costs  $O(n^\epsilon)$  steps]; At last we perform the instructions "DELETE the vertices of the graph from the  $\ell_1$ -th leaf of  $T(R_S)$ " and "INSERT these vertices into the  $\ell_2$ -th leaf of  $T(R_G)$ " [these instructions are performed in a constant number of steps]. Thus, the transferring of each vertex costs  $O(n^{1-K\epsilon} + n^\epsilon)$  steps, and the total complexity of these operations throughout the first stage is therefore  $O((n^{2-K\epsilon} + n^{1+\epsilon}) \cdot \lg n)$ .

In order to satisfy the condition on the instruction UNION we perform the following operations on each vertex  $T$  which is transferred from  $T(R_S)$  into  $T(R_G)$ : We perform the legality test on the first bridge  $ST$  of  $CHAIN(T)$ . If  $ST$  fails this test, then both  $S$  and  $T$  belong to the same set  $T(R_2)$ . We delete the bridge  $TS$  from  $CHAIN(S)$ , and if  $TS$  is the first bridge of  $CHAIN(S)$  then we insert  $S$  into a stack. We repeat this process on  $T$  until its  $CHAIN$  becomes empty, or until its first bridge passes the legality test. Then the variable  $VALUE(T)$  is updated, and if necessary  $T$  is transferred into another leaf of  $T(R_2)$ . After those operations on  $T$  are terminated we repeat the process for each vertex  $S$  in the stack. Clearly, if at

the end of all those operations  $\ell_1$  is the smallest index such that  $\text{NUM}(\ell_1) \neq 0$ , then there exists in the chain of vertices of the  $\ell_1$ -th leaf a vertex  $T$  whose CHAIN contains a bridge of level  $\ell_1$  which presently passes the legality test.

Since a constant number of operations are performed per each bridge which fails the legality test, the total complexity of those operations is  $O(m)$ .

The complexity of the implementation and the value of  $K$ . By the above discussion, the complexity of Implementation 4 is  $O(m + (n^{2-K\epsilon} + n^{1+\epsilon}) \cdot \lg n)$ . If we choose  $K$  to be  $\frac{1}{\epsilon}$  then we get complexity  $O(m + n^{1+\epsilon} \lg n)$ . Since for any positive value  $\epsilon$  there exists  $O(\lg n) < O(n^\epsilon)$ , it follows that the complexity of this implementation is  $O(m + n^{1+2\cdot\epsilon})$  or  $O(m + n^{1+\epsilon'})$  where  $\epsilon'$  is an arbitrary positive value.

APPENDIX C

This appendix contains a listing of a PL/1 program which is based on Implementation 1 of Chapter 5. This program was run on an IBM-370/165 computer, and the cardinalities of the matchings which were achieved have been verified by comparing them to the cardinalities of the matchings which were received by a PL/1 program based on Gabow's algorithm [3].

The input. The input to the program is done by using punched cards: The first card is in the form:

$$N = n \quad , \quad M = m ;$$

[where  $n$  and  $m$  are the numbers of the vertices and the edges of the graph respectively].

The next cards contain a sequence of  $2m$  numbers which are separated by blanks or comma. The  $(2j-1)$ -th and the  $2j$ -th numbers in this sequence represent the  $j$ -th edge of the graph in the following way: Let  $n_{2j-1}$  be the  $(2j-1)$ -th number in the sequence, and let  $n_{2j}$  be the  $2j$ -th number. Then the  $j$ -th edge of the graph goes from vertex  $n_{2j-1}$  to vertex  $n_{2j}$  (clearly,  $1 \leq n_{2j-1}, n_{2j} \leq n$ ).

After receiving the input, the program prints this input in the following format: The first two rows are:

```
THE NUMBER OF VERTICES IS      n
THE NUMBER OF EDGES.  IS      m
```

Then, under the heading THE LIST OF EDGES , a list of the edges is printed, where the  $j$ -th edge which goes from vertex  $n_{2j-1}$  to vertex  $n_j$  appears in the following format,

$$\begin{array}{c} \underline{j} \\ n_{2j-1} \\ | \\ n_{2j} \end{array}$$

The output. The output is printed under the heading RESULTS .  
The first two rows of the output are:

NUMBER OF MATCHED EDGES:  $n_r$   
NUMBER OF EXPOSED VERTICES:  $n_n$

[where  $n_r$  and  $n_n$  are respectively the numbers of matched edges and exposed vertices according to the maximum matching].

Then, under the heading MATCHING , a list of the vertices and their mates is printed, where the  $i$ -th vertex and its mate appear in the following format:

$$\begin{array}{c} i \\ | \\ n_i \end{array}$$

[ $n_i$  is the number of the vertex which is the mate of the  $i$ -th vertex].

If the  $i$ -th vertex is exposed, then instead the number  $n_i$  , there appears the word NONE .

[Notice that if  $i$  is not exposed, then the matched edge  $i-n_i$

appears twice in the output: Once as  $\begin{matrix} i \\ | \\ n_i \end{matrix}$  and the other time as  $\begin{matrix} n_i \\ | \\ i \end{matrix}$ .  
 The output is ended by a list of the matched edges which are printed under the heading LIST OF MATCHED EDGES.

### The listing

The numbers of the labels are in accordance with the numbers of the steps in the algorithms (e.g., label B4 corresponds to step no. 4 in the second stage, and label SL5 corresponds to step no. 5 in the algorithm of SRCHLOOP ).

```

MATCH: PROC OPTIONS(MAIN);

/* DECLARE THE VARIABLES */
DEFAULT RANGE(A:Z) BIN FIXED;
DCL ((E,NXTE)(-M:M),V(N),
      (PRE,POST)(-N:N),MATE(-N:N),
      (LEV,LINK)(-N:N),LIST(0:N1),
      (TAIL,NEXT)(-N:N),(FCHN,LCHN)(-N:N),NXCHN(-M:M),
      (VLIST,ELIST,BASE,BSLIST)(N),(ER,NXTER)(-M:M),SOURCE(M),
      (LIFE,RLINK)(-N:N),(TOP,RTNXT)(-N:N),ACTLIST(2*N),
      NEWMATE(N) )
BIN FIXED CTL;
/* IN THE PROGRAM: THE INDEX I STANDS FOR VERTICES,
                   THE INDEX K STANDS FOR EDGES
                   THE INDICES J,L STAND FOR LEVELS. */

/* INPUT THE GRAPH */
GET DATA(N,M);
PUT PAGE EDIT('THE NUMBER OF VERTICES IS',N,'THE NUMBER OF',
              ' EDGES IS',M)(X(10),A,F(10),SKIP,X(10),A,A,F(10));
ALLOCATE E,NXTE,V;
NXTE=0; V=0;
DO K=1 TO M; GET LIST(E(K),E(-K));
              NXTE(K)=V(E(K)); NXTE(-K)=V(E(-K));
              V(E(K))=K; V(E(-K))=-K;
END;
PUT SKIP(3) EDIT('THE LIST OF EDGES',(25)'-')(X(30),A,SKIP,
                                                X(25),A);

K1=1;
IN1: K2=K1+14; If K2>M THEN K2=M; K3=K2+1-K1;
PUT SKIP(4) EDIT((K DO K=K1 TO K2),('---' DO K=K1 TO K2),(E(K)
DO K=K1 TO K2),('|' DO K=K1 TO K2),(E(-K) DO
K=K1 TO K2))((K3)F(8),SKIP,(K3)(X(5),A),SKIP,
(K3)F(8),SKIP,(K3)(X(7),A),SKIP,(K3)F(8));

```

```

IF K2=M THEN DO; K1=K2+1; GO TO IN1; END;

/* INITIALIZATION */
ALLOCATE PRE, POST, MATE, LEV, LINK, LIST(0:0);
N1=N/2+1; MM=3*M; NN=N;
/* NN IS THE NUMBER OF EXPOSED VERTICES */
DO I=1 TO N; MATE(I)=-I; MATE(-I)=I;
PRE(I)=I-1; POST(I)=I+1;
END; POST(N)=0; LIST(0)=1;

/* MAIN PROCEDURE */
M1: CALL STAGE1;
CALL STAGE2;
CALL STAGE3;
CALL STAGE4;
GO TO M1;

/* FIRST STAGE */
STAGE1: PROC;
A0: EXP=LIST(0); FREE LIST;
ALLOCATE TAIL, NEXT, FCHN, LCHN, NXCHN, LIST;
DO I=1 TO N1; LIST(I)=0; END; LIST(0)=EXP;
FCHN, LCHN=0; NXCHN=0; NEXT=0;
DO I=1 TO N; TAIL(I)=MATE(I); TAIL(-I)=MATE(-I);
IF MATE(I)>0 THEN DO; I1=LIST(N1);
POST(I)=I1; LIST(N1)=I;
IF I1=0 THEN PRE(I1)=I;
LINK(I)=0; LEV(I)=N1;
END;
ELSE DO; I1=LIST(N1); POST(-I)=I1; LIST(N1)=-I;
IF I1=0 THEN PRE(I1)=-I; LINK(-I)=0;
LEV(-I)=N1; LINK(I)=0; LEV(I)=0;
END;
END; PRE(LIST(N1))=0;
J=1;

SUBSTAGE: X=LIST(J-1);
SEARCH:
A1: IF X=0 THEN GO TO UPDATE; K=V(X); GO TO A2A;
A2: K=NXTE(K);
A2A: IF K=0 THEN DO; X=POST(X); GO TO SEARCH; END;
Y=E(-K); Z=MATE(Y); IF Z=X THEN GO TO A2;
A3: IF LEV(Y)<J THEN GO TO A6; IF LINK(Y)>M THEN GO TO A6;
A4: IF LINK(Z)=0 THEN GO TO A5;
LINK(Z)=MM+K; LEV(Z)=J; I1=PRE(Z); I2=POST(Z);
IF I1=0 THEN POST(I1)=I2; ELSE LIST(N1)=I2;
IF I2=0 THEN PRE(I2)=I1; I1=LIST(J); PRE(Z)=0; POST(Z)=I1;
LIST(J)=Z; IF I1=0 THEN PRE(I1)=Z; GO TO A2;
A5: IF FCHN(Y)=0 THEN DO; NXCHN(LCHN(Y))=K; LCHN(Y)=K; END;
ELSE DO; FCHN(Y)=K; LCHN(Y)=K; END;
GO TO A2;

```

```

A6: IF LINK(MATE(X))=MM-K THEN GO TO A2;
    R=TAIL(Y); IF R=TAIL(X) THEN GO TO A2;
A7: IF FCHN(Y)≠0 THEN DO; NXCHN(LCHN(Y))=K; LCHN(Y)=K; END;
    ELSE DO; FCHN(Y)=K; LCHN(Y)=K; END;
    L1=LEV(X)+LEV(Y)+1-LEV(MATE(R)); IF L1>=LEV(R) THEN GO TO A2;
A8: I1=PRE(R); I2=POST(R); IF I2≠0 THEN PRE(I2)=I1;
    IF I1≠0 THEN POST(I1)=I2; ELSE LIST(LEV(R))=I2;
    I1=LIST(L1); PRE(R)=0; POST(R)=I1; LIST(L1)=R;
    IF I1≠0 THEN PRE(I1)=R; LEV(R)=L1; LINK(R)=K; GO TO A2;

```

UPDATE:

```

A9: I1=0; R=LIST(J);
A9A: IF R=0 THEN GO TO A9B;
    IF R<0 THEN DO; RR=J; FREE TAIL,NEXT,FCHN,LCHN,NXCHN;
        RETURN; END;
    IF LINK(R)<=M THEN I1=1; R=POST(R); GO TO A9A;
A9B: IF I1=0 THEN GO TO ENDSUB; R=LIST(J);
A10: IF R=0 THEN GO TO ENDSUB;
    IF LINK(R)>M THEN DO; R=POST(R); GO TO A10; END;
    K=LINK(R); S=E(K);
A11: R2=TAIL(E(LINK(MATE(R))-MM)); R3=TAIL(S);
A12: CALL MERGE(R2,R); CALL MERGE(R3,MATE(R));
    R=POST(R); GO TO A10;

```

ENDSUB:

```

A13: IF J<N1-1 THEN DO; J=J+1; GO TO SUBSTAGE; END;
A14: GO TO RESULTS; /* MAXIMUM MATCHING */

```

MERGE: PROC(R2,R);

```

DCL R2,R;
    U=MATE(R);
MR1: TAIL(U)=R2; U=NEXT(U); IF U≠0 THEN GO TO MR1;
    U=MATE(R2);
MR2: IF NEXT(U)≠0 THEN DO; U=NEXT(U); GO TO MR2; END;
MR3: NEXT(U)=MATE(R);
MR4: U1=PRE(R2); U2=POST(R2); IF U2≠0 THEN PRE(U2)=U1;
    IF U1≠0 THEN POST(U1)=U2; ELSE LIST(LEV(R2))=U2;
    LEV(R2)=N1; LINK(R2)=0; T=MATE(R2);
MR5: IF FCHN(T)=0 THEN GO TO MR8;
    K=FCHN(T); S=E(K);
MR6: IF TAIL(S)=R2 THEN DO; FCHN(T)=NXCHN(K); GO TO MR5; END;
MR7: L2=LEV(S)+LEV(T)+1-LEV(MATE(R2));
    IF L2<LEV(R2) THEN DO; LEV(R2)=L2; LINK(R2)=K; END;
MR8: T=NEXT(T); IF T≠0 THEN GO TO MR5;
MR9: U1=LIST(LEV(R2)); IF U1≠0 THEN PRE(U1)=R2;
    PRE(R2)=0; POST(R2)=U1; LIST(LEV(R2))=R2;
RETURN;
END MERGE;
END STAGE1;

```

```

/* SECOND STAGE */
STAGE2: PROC;

```



```

B0: ALLOCATE VLIST,BASE,BSLIST,ELIST,ER,NXTER,SOURCE;
    VLIST=0; NR=0; J=0;
    /* NR IS THE NUMBER OF VERTICES IN THE REDUCED GRAPH */
VERTICES: Y=LIST(J); GO TO B1A;
B1: Y=POST(Y);
B1A: IF Y=0 THEN GO TO B4;
B2: Z=MATE(Y); L1=LEV(Y)+LEV(Z);
    IF L1>RR THEN DO; BASE(Y)=Y; GO TO B1; END;
    IF L1=RR THEN DO; BASE(Y)=Y; GO TO B3; END;
    IF LINK(Y)>M THEN S=E(LINK(Y)-MM); ELSE S=E(LINK(Y));
    BASE(Y)=BASE(S);
    BSLIST(Y)=BSLIST(S); BSLIST(S)=Y; GO TO B1;
B3: NR=NR+1; VLIST(NR)=Y; BSLIST(Y)=0; ELIST(Y)=0; GO TO B1;
B4: J=J+1; IF J=RR THEN GO TO EDGES; GO TO VERTICES;

EDGES: I=0; NXTER=0; MR=0;
    /* MR IS THE NUMBER OF EDGES IN THE REDUCED GRAPH */
B5: IF I=NR THEN DO; EXP=LIST(0);
    FREE BASE,BSLIST,LIST; ALLOCATE LIST(0:0); LIST(0)=EXP;
    RETURN; END;
    I=I+1; P=VLIST(I); Y=P; GO TO B7A;
B6: Y=BSLIST(Y); IF Y=0 THEN GO TO B5;
B7A: K=V(Y); GO TO B7B;
B7: K=NXTE(K);
B7B: IF K=0 THEN GO TO B6; X=E(-K); IF X=MATE(Y) THEN GO TO B7;
B8: IF LEV(X)+LEV(Y)+1=RR THEN GO TO B7;
    Q=BASE(X); IF LEV(Q)+LEV(MATE(Q))>RR THEN GO TO B7;
    IF P>=Q THEN GO TO B7;
    K1=ELIST(Q); IF K1=0 THEN GO TO B8A;
    IF ER(-K1)=P THEN GO TO B7;
B8A: MR=MR+1; ER(MR)=P; ER(-MR)=Q; SOURCE(MR)=K;
    NXTER(-MR)=K1; ELIST(Q)=-MR;
    NXTER(MR)=ELIST(P); ELIST(P)=MR;
    GO TO B7;
END STAGE2;

```

/\* THIRD STAGE \*/

```

STAGE3: PROC;
C0: ALLOCATE LIFE,RLINK,TOP,RTNXT,ACTLIST;
    LIFE=N+1; RLINK=0; RTNXT=0; NACT=0;
    DO I=1 TO N; TOP(I)=MATE(I); TOP(-I)=MATE(-I); END;
    I=0;
PART: I=I+1; IF I<=NR THEN GO TO C1;
C1A: FREE ELIST,NXTER,TOP,RTNXT,ACTLIST; RETURN;
C1: P=VLIST(I); IF LEV(P)=0 THEN GO TO C1A;
    IF LIFE(P)<N+1 THEN GO TO PART; RLINK(P)=-MM;
C2: J=0; NACT=1; ACTLIST(1)=P; LIFE(P)=I; LIFE(MATE(P))=I;
C3: K=ELIST(P); IF K=0 THEN GO TO C7;
    ELIST(P)=NXTER(K); Q=ER(-K);
C4: IF LIFE(Q)<I THEN GO TO C3;

```

```

IF RLINK(Q)≠0 THEN GO TO C6; U=MATE(Q);
C5: IF RLINK(U)≠0 THEN GO TO C3; RLINK(U)=MM+K;
LIFE(Q)=I; LIFE(U)=I; NACT=NACT+1; ACLIST(NACT)=U; J=LEV(U);
IF J=RR THEN GO TO PART; P=U; GO TO C3;
C6: IF RLINK(MATE(P))=MM-K THEN GO TO C3;
IF LEV(P)≤LEV(Q) THEN GO TO C3;
CALL SRCHLOOP(K); GO TO C3;
C7: IF NACT=1 THEN GO TO PART; L1=0; Q=ACTLIST(1); NACT=NACT-1;
DO I1=2 TO NACT; U=ACTLIST(I1);
IF U=P THEN DO; U=ACTLIST(NACT+1);
ACTLIST(I1)=U; END;
IF LEV(U)>L1 THEN DO; Q=U; L1=LEV(U); END;
END;
P=Q; J=L1; GO TO C3;

```

```

SRCHLOOP: PROC(K);
SL1: P=ER(K); Q=ER(-K); U=TOP(Q); R=TOP(P);
SL2: IF U=R THEN RETURN;
SL3: RLINK(R)=-K; NACT=NACT+1; ACTLIST(NACT)=R;
SL4: W=MATE(R); CALL UNION(U,W);
SL5: S=TOP(ER(RLINK(W)-MM)); CALL UNION(S,R);
SL6: IF S=U THEN DO; R=S; GO TO SL3; END;
SL7: IF LEV(R)≤J THEN RETURN; J=LEV(R); P=R; RETURN;

```

```

UNION: PROC(R1,R2);
DCL R1,R2;
S1=MATE(R2);
U1: TOP(S1)=R1;
S1=RTNXT(S1); IF S1=0 THEN GO TO U1;
S1=MATE(R1);
U2: IF RTNXT(S1)≠0 THEN DO; S1=RTNXT(S1); GO TO U2; END;
RTNXT(S1)=MATE(R2);
RETURN;
END UNION;
END SRCHLOOP;
END STAGE3;

```

/\* FOURTH STAGE \*/

```

STAGE4: PROC;
DO: ALLOCATE NEWMATE;
DO I=1 TO N; NEWMATE(I)=MATE(I); END; I=0;
D1: IF I=NR THEN GO TO D5; I=I+1; P=VLIST(I);
IF LEV(P)>0 THEN GO TO D5; P=MATE(P);
D2: IF RLINK(P)=0 THEN GO TO D1; A=VLIST(LIFE(P));
D3: CALL RPATH(A,P);
D4: I1=PRE(A); I2=POST(A); IF I2=0 THEN PRE(I2)=I1;
IF I1=0 THEN POST(I1)=I2; ELSE LIST(0)=I2;
IF A=MATE(P) THEN DO; A=MATE(P); GO TO D4; END;
NN=NN-2; GO TO D1;

```

```

D5: DO I=1 TO N; MATE(I)=NEWMATE(I); END;
    FREE VLIST,ER,SOURCE,LIFE,RLINK,NEWMATE; RETURN;

RPATH: PROC(A,P) RECURSIVE;
    DCL A,P,R,W,Y,X;
    RP1: IF A=P THEN RETURN;
        K=RLINK(P); IF K>M THEN K=K-MM; R=ER(K); W=ER(-K);
    RP2: IF K>0 THEN K1=SOURCE(K); ELSE K1=-SOURCE(-K);
        X=E(K1); Y=E(-K1); NEWMATE(X)=Y; NEWMATE(Y)=X;
    RP3: CALL PATH(R,X); CALL PATH(W,Y);
    RP4: CALL RPATH(A,R);
    RP5: IF W=MATE(R) THEN CALL RPATH(MATE(P),W);
    RP6: RETURN;

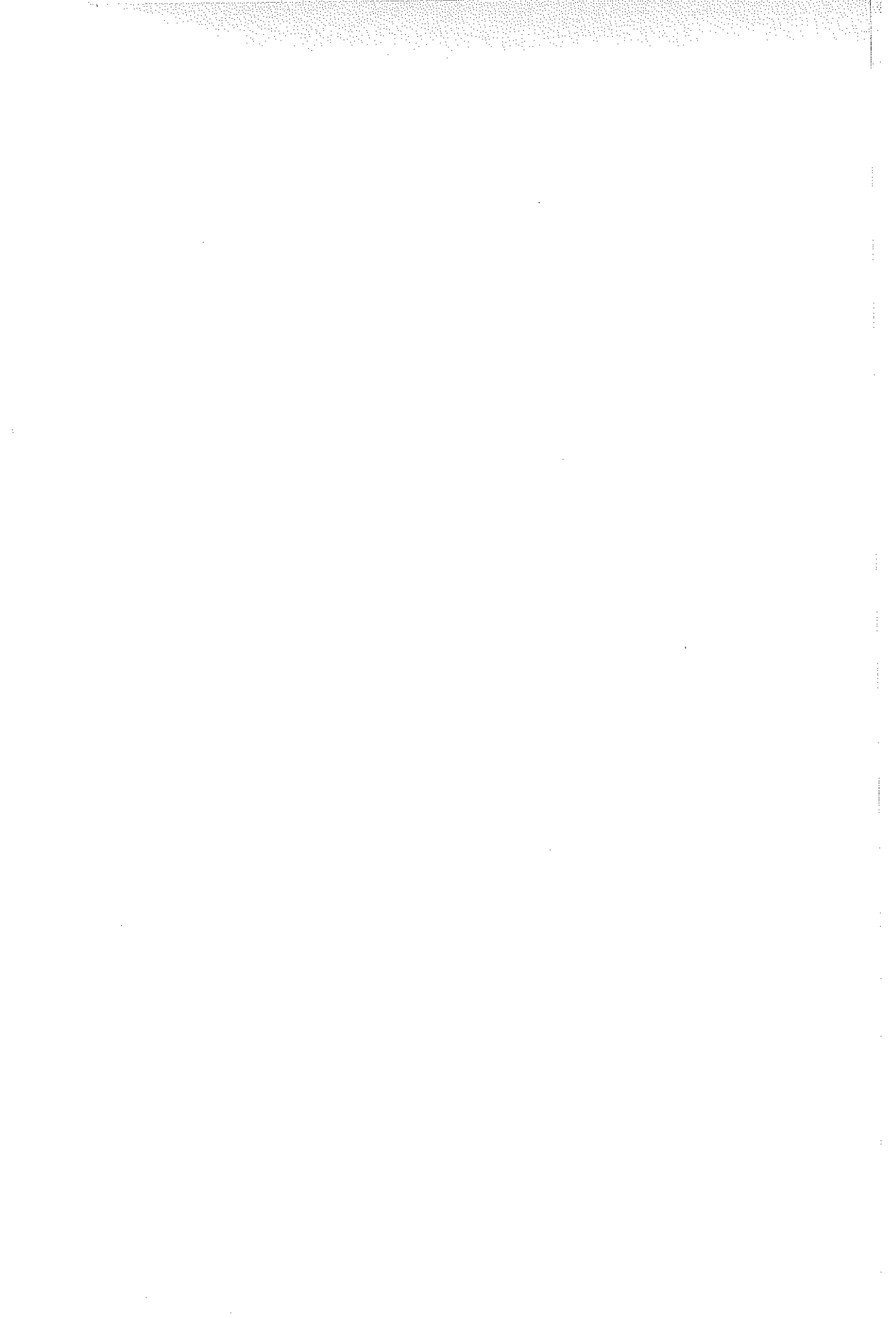
PATH: PROC(A,P) RECURSIVE;
    DCL A,P,W,R;
    P1: IF A=P THEN RETURN;
        K=LINK(P); IF K>M THEN K=K-MM; R=E(K); W=E(-K);
    P2: NEWMATE(R)=W; NEWMATE(W)=R;
    P3: CALL PATH(A,R);
    P4: IF W=MATE(P) THEN CALL PATH(MATE(P),W);
    P5: RETURN;
END PATH;
END RPATH;
END STAGE4;

/* PRINT RESULTS */
RESULTS: BEGIN;
    DCL EN PIC'ZZZ9'; DCL EE(N) CHAR(4);
    PUT PAGE EDIT('RESULTS',(20)'-')(X(20),A,SKIP,X(15),A);
    NR=(N-NN)/2;
    PUT SKIP(5) EDIT('NUMBER OF MATCHED EDGES:',NR,'NUMBER OF EXP'
        , 'USED VERTICES:',NN)(X(20),A,F(6),SKIP,X(20),A,A,F(6));
    DO I=1 TO N; IF MATE(I)<0 THEN EE(I)='NONE';
        ELSE DO; EN=MATE(I); EE(I)=EN; END; END;
    PUT SKIP(3) EDIT('MATCHING',(20)'-')(X(20),A,SKIP,X(15),A);
    I1=1;
    RS1: I2=I1+14; IF I2>N THEN I2=N; I3=I2+1-I1;
        PUT SKIP(2) EDIT((I DO I=I1 TO I2),('|' DO I=I1 TO I2),(EE(I)
            DO I=I1 TO I2))((I3)(X(4),F(4)),SKIP,(I3)(X(7),
                A),SKIP,(I3)(X(4),A(4)));
        IF I2=N THEN DO; I1=I2+1; GO TO RS1; END
        K1=0;
        DO K=1 TO M; IF MATE(E(K))=E(-K) THEN DO;
            K1=K1+1; LEV(K1)=K; END; END;
        PUT SKIP EDIT('LIST OF MATCHED EDGES',(30)'-')(X(20),A,SKIP
            ,X(15),A);
        PUT SKIP EDIT((LEV(K) DO K=1 TO K1))(F(8));
    END RESULTS;
END MATCH;

```

REFERENCES

1. Edmonds, J., "Paths, Trees and Flowers"; Canadian J. 1965, Vol. 17, pp. 449-467. "Maximum Matching and Polyhedron with 0,1 vertices"; Journal of Research of the National Bureau of Standards, Jan.-June 1965, Vol. 69B, pp. 125-130.
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, Vol. 69B, pp. 91-98.
3. Gabow, H., "An Efficient Implementation of Edmonds' Maximum Matching Algorithm"; June 1972, Technical Report No. 31, Stan-CS, 72-328, to be published JACM. "Implementation of Algorithms for Maximum Matching on Non-Bipartite Graphs"; Ph.D. dissertation, Stanford University, 1973.
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, Vol. 43, pp. 842-844.
5. Hopcroft, J.E. and Karp, R.M., "An  $n^{\frac{5}{2}}$  Algorithm for Maximum Matching in Bipartite Graphs"; SIAM J. on Comp. 2, December 1973, pp. 225-231.

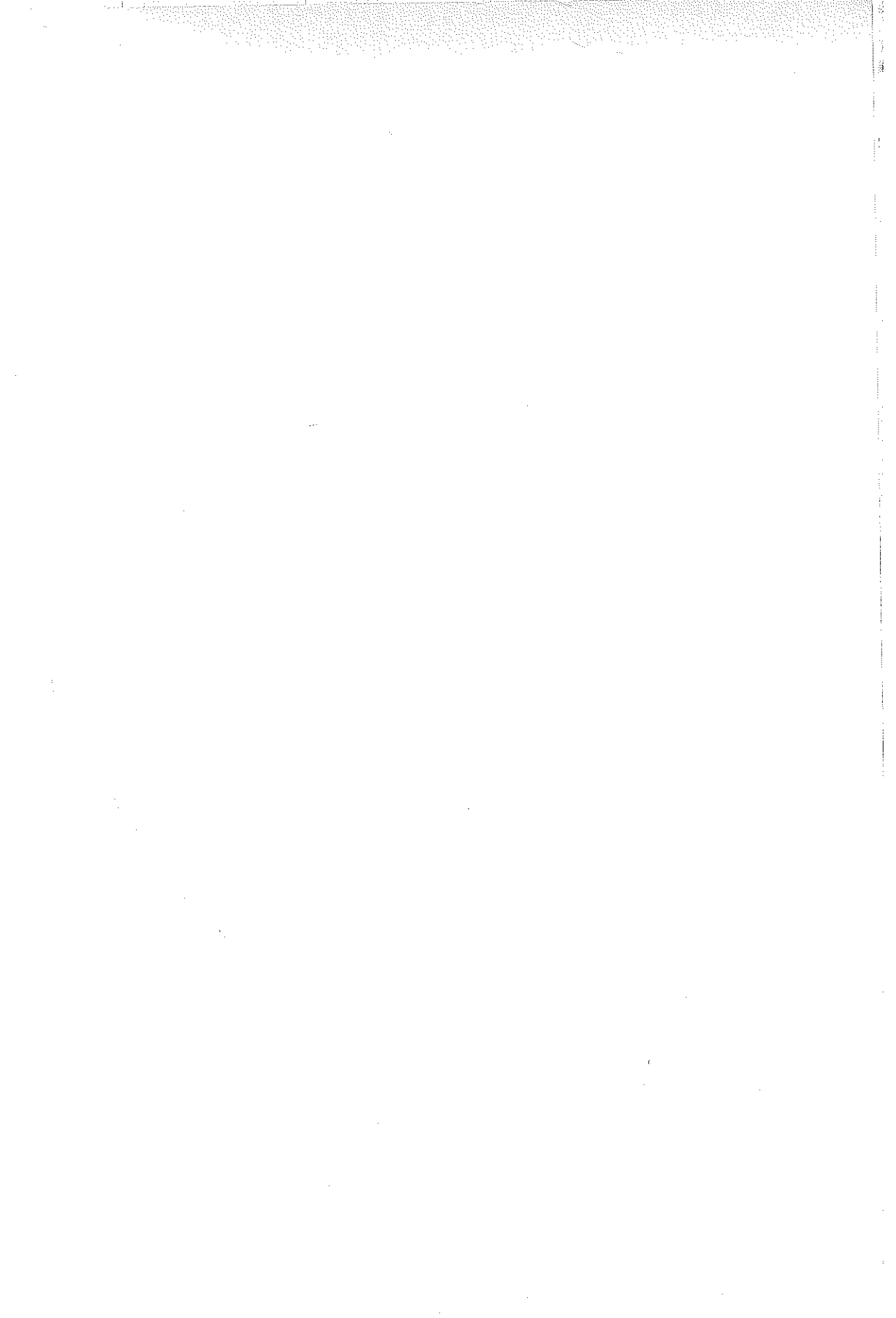




6. Aho, A.V., Hopcroft, J.E. and Ullman, J.D., "The Design and Analysis of Computer Algorithms"; Addison-Wesley Publishing Company, 1974, Chapter 4, pp. 124-163.
7. Emde Boas, P. Van, "Preserving Order in a Forest in Less Than Logarithmic Time"; Proceedings of the 16th Annual Symposium on Foundations of Computer Science (FOCS), Berkeley 1975, pp. 75-84.
8. Balinski, M.L., "Labelling to Obtain a Maximum Matching"; in R.C. Bose and T.A. Dowling ed., Proc. of the Conference on Combinatorial Mathematics and Its Applications, Univ. of North Carolina, April 1967, pp. 582-602.
9. Kameda, T. and Munro, I., "A  $O(|V| \cdot |E|)$  Algorithm for Maximum Matching of Graphs"; Computing 1974, Vol. 12, pp. 91-98.
10. Lawler, E.L., "Combinatorial Optimization Theory", to be published.
11. Even, S. and Kariv, O., "An  $O(n^{2.5})$  Algorithm for Maximum Matching in General Graphs", Proceedings of the 16th Annual Symposium on Foundations of Computer Science (FOCS), Berkeley 1975, pp. 100-112.







מתאימה לקבוצה מקסימלית של מסלולי-שיפור זרים חוקיים ומינימליים בגרף המקורי.

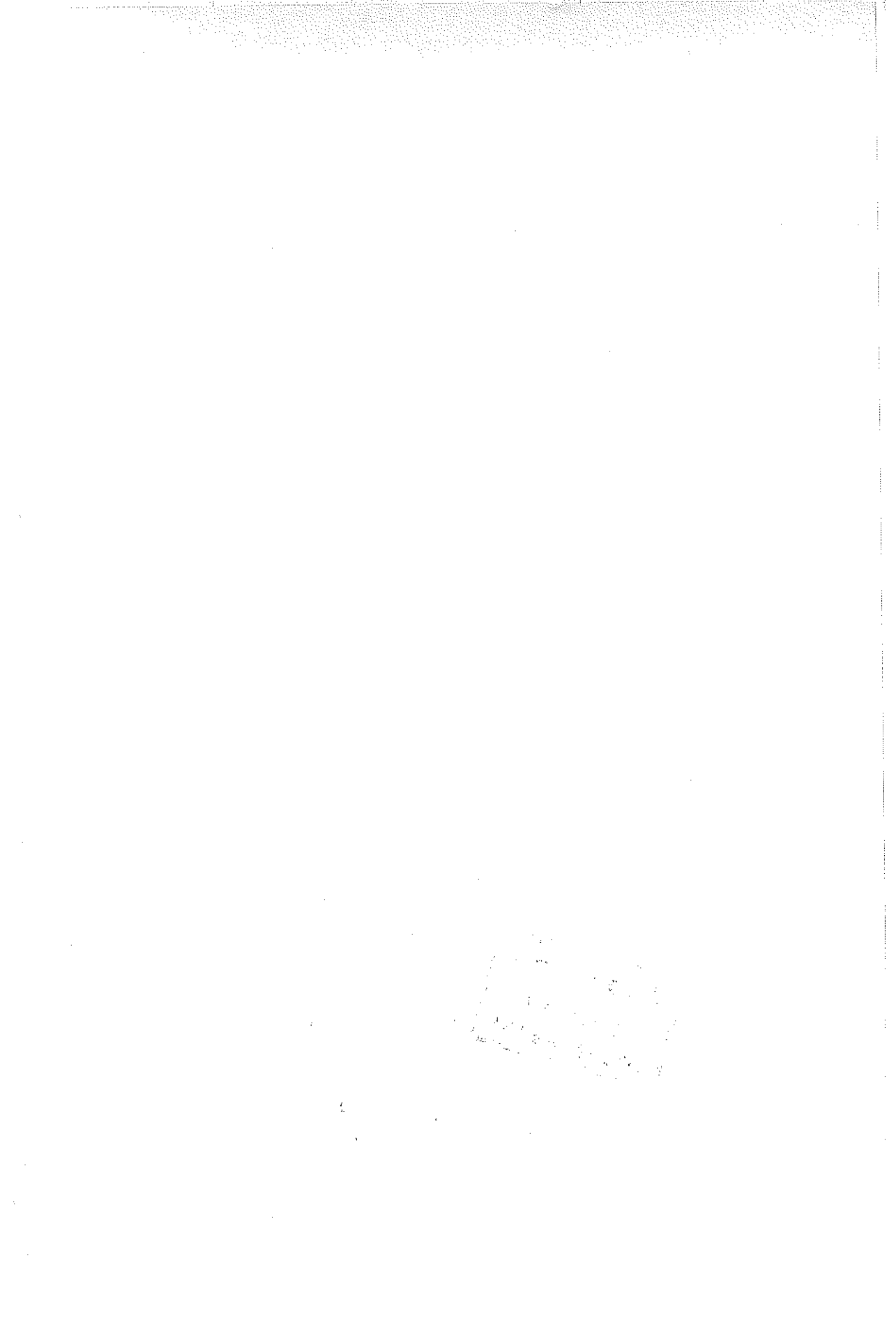
תכונה זו של הגרף המצומצם מנוצלת בשלב השלישי של התהליך שבו אנו סורקים לעומק (Depth First Search) את הגרף המצומצם ומוציאים בו קבוצה מקסימלית של מסלולי-שיפור זרים חוקיים בעלי אורך שרירותי. השמטת הדרישה בדבר מינימליות המסלולים מאפשרת ביצוע תהליך זה ב- $n^2$  צעדים, תוך שימוש ברמות שהוגדרו לצמתים בשלב הראשון ובמבני-נתונים הדומים לאלה של השלב הראשון.

בשלב הרביעי אנו משחזרים את מסלולי-השיפור שמצאנו בגרף המצומצם למסלולי-שיפור מינימליים של הגרף המקורי. כתוצאה משחזור זה מתקבלת קבוצה מקסימלית של מסלולי-שיפור זרים חוקיים ומינימליים בגרף המקורי שעליה אנו מבצעים את התהליך של Berge. לאחר שיפור הצימוד אנו חוזרים לשלב הראשון להתחיל תהליך חדש.

ע"י שינוי במבני-הנתונים אפשר לקבל גם תהליכים הפועלים ב-  
 $O(m \lg \lg n)$  צעדים או ב- $(m/\bar{n} + n^{1.5+\epsilon})$  צעדים [כאשר  $\epsilon$  הוא ערך חיובי שרירותי].

כנספח לעבודה מופיעה תכנית מחשב PL/1 המבוססת על התהליך המתואר

בעבודה עצמה.

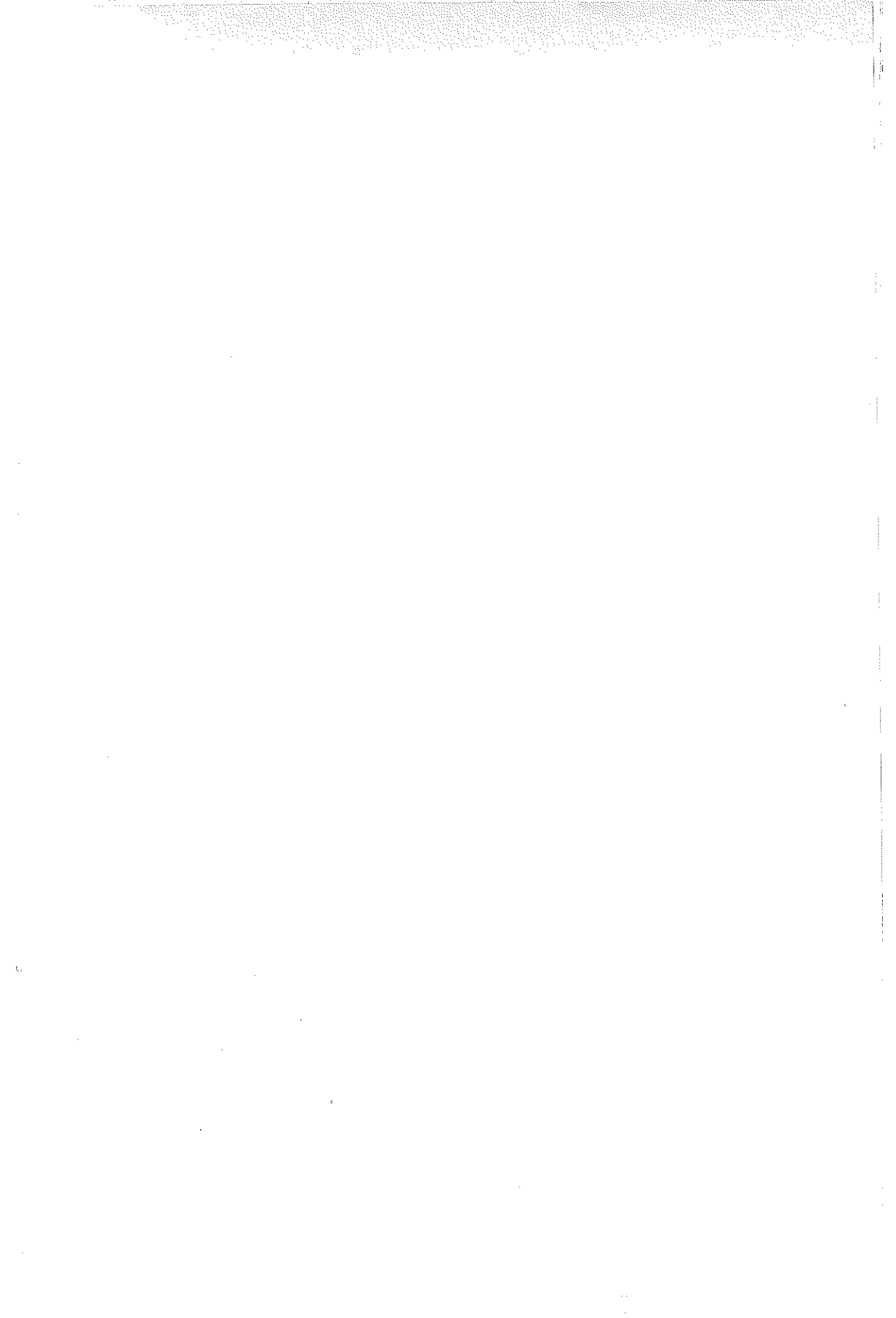


הקושי העיקרי במציאת מסלולי-שיפור בגרף כללי הוא שעל המסלולים הללו להיות "חוקיים" (כלומר פשוטים, חסרי לולאות). בעוד שבגרף דו-צדדי כל מסלול-שיפור שהוא מינימלי הוא גם חוקי, הרי בגרף כללי בהחלט יתכן שמסלול שיפור מינימלי יכיל לולאות. עובדה זו הופכת את הבעיה של מציאת קבוצה מקסימלית של מסלולי-שיפור זרים חוקיים ומינימליים לבעיה מסובכת שלשם פתירתה יש להשתמש במגוון של השיטות הנהוגות בתורת הגרפים.

התהליך המתואר בעבודה זו מוצא קבוצה מקסימלית של מסלולי-שיפור זרים חוקיים ומינימליים ב- $n^2$  צעדים. התהליך מבוצע ב-4 שלבים:

בשלב הראשון (שהוא "לב" התהליך) מתבצעת טריקה לרוחב של הגרף (Breadth First Search) המתחילה בבת אחת מכל הצמתים המבודדים. לכל צומת של הגרף אנו מוצאים מסלול שיפור חוקי ומינימלי המוליך אליו מאחד הצמתים המבודדים. ארכו של מסלול זה מוגדר בתור רמתו של הצומת. בסופו של שלב זה אנו יודעים אם קיים מסלול שיפור חוקי בגרף ואם כן - מהו ארכו של מסלול השיפור החוקי המינימלי (אם לא קיים בגרף מסלול שיפור חוקי הרי שהצימוד הנוכחי הינו מקסימלי והתהליך נפסק). במהלך שלב זה אנו משתמשים במבני-נתונים מיוחדים בהם אנו מאכסנים מידע המאפשר לנו לבנות במספר קבוע של צעדים מסלולי-שיפור חדשים במקרה שמתברר כי חלק ממסלולי-השיפור הקיימים אינם חוקיים.

בשלב השני אנו מאתרים (בעזרת הרמות שהוגדרו בשלב הראשון) את כל הצמתים והקשתות שאינם נמצאים על מסלולי-שיפור מינימליים ואנו מסלקים אותם מהגרף. בשלב זה אנו גם "מכווצים" את כל הלולאות הנמצאות על מסלולי-שיפור לא-חוקיים שארכו קטן מאורך מסלולי-השיפור המינימליים החוקיים. כתוצאה מתהליך זה מתקבל גרף מצומצם בעל התכונה הבאה: כל קבוצה מקסימלית של מסלולי-שיפור זרים חוקיים מאורך כלשהו בגרף המצומצם



תהליך ב- $n^{2.5}$  צעדים למציאת צימוד מקסימלי בגרף כללי

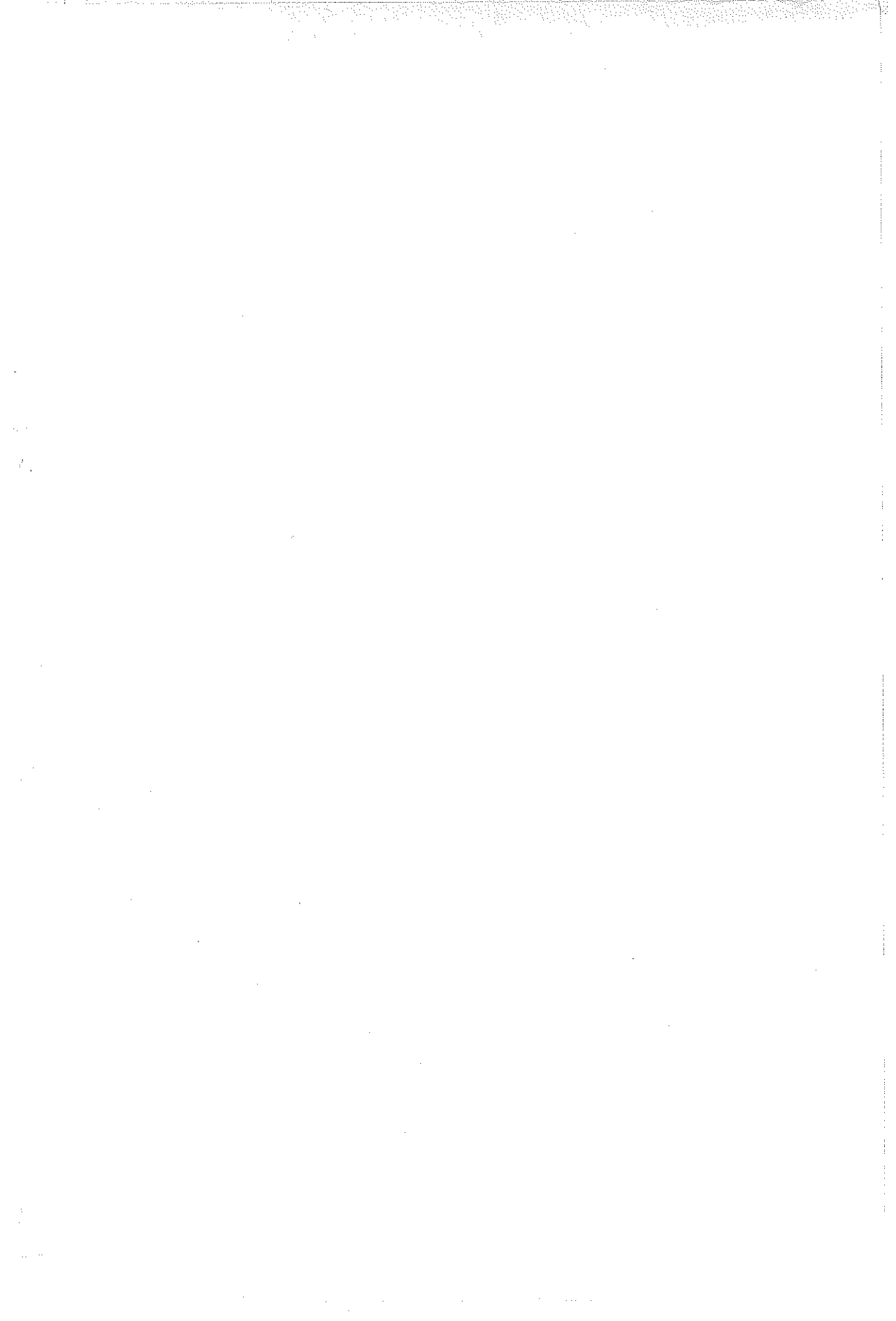
עבודה זו נעשתה תחת השגחתו של פרופסור שמעון אבן, במחלקה למתמטיקה שמושית, מכון ויצמן למדע, רחובות, ישראל.

### תקציר

צימוד (matching) בגרף כללי בעל  $n$  צמתים ו- $m$  קשתות זו תת-קבוצה של קשתות הגרף כך שלכל שתי קשתות השייכות לצימוד אין צומת משותף. הבעיה של מציאת צימוד המכיל מספר מקסימלי של קשתות נדונה ע"י מספר חוקרים (ראה [1], [2], [3], [8], [9], [10]) אולם התהליכים הטובים ביותר שהושגו עד כה היו תהליכים ב- $n^3$  צעדים.

כל התהליכים הידועים לנו מתבססים על משפטו של Berge ([4]), האומר כי צימוד נתון בגרף אינו מקסימלי אם ורק אם קיים בגרף מסלול שיפור חילופי (שבו הקשתות הן לסירוגין קשתות השייכות לצימוד ושאינן שייכות לו), המחבר שני צמדים מבודדים (כלומר צמתים שאינם נמצאים על קשתות מהצימוד). במקרה זה, ע"י הוצאתן מהצימוד של כל הקשתות הנמצאות על המסלול יהשייכות לצימוד והכנסתן לצימוד של כל הקשתות האחרות הנמצאות על המסלול, יגזל הצימוד ב-1.

Hopcroft ו-Karp ([5]) מצאו שאם נבצע את תהליך השיפור הנייל בבת אחת על קבוצה מקסימלית של מסלולי - שיפור זרים שכולם בעלי אורך מינימלי — אזי מספר פעולות השיפור הדרושות עד למציאתו של צימוד מקסימלי יהיה לכל היותר  $\sqrt{n}$ . לפיכך, אם נוכל למצוא קבוצה מקסימלית של מסלולי-שיפור זרים מינימליים בתהליך הפועל ב- $n^2$  צעדים הרי שקבלנו תהליך למציאת צימוד מקסימלי בגרף הפועל ב- $n^{2.5}$ . Hopcroft ו-Karp הצליחו למצוא תהליך כזה לגרף דו-צדדי (bipartite).



# תהליך ב-2.5 ח צעדים למציאת צימוד מקסימלי בגרף כללי

חיבור לשם קבלת התואר  
דוקטור לפילוסופיה

מאת

## עודד קריב

הוגש למועצה המדעית של מכון ויצמן למדע

רחובות

מרץ 1976

אדר ב' תשל"ו