Computing 12, 91—98 (1974)

A O(|V|·|E|) Algorithm for Maximum Matching of Graphs*

By

T. Kameda and **I. Munro**, Waterloo, Ontario

With 2 Figures

Received July 25, 1973

Abstract — Zusammenfassung

A O(|V|·|E|) Algorithm for Maximum Matching of Graphs. This paper presents an algorithm which finds a maximum matching in a graph with n nodes and m edges within time k_1mn+k_2, where k_1 and k_2 are constants. It is also shown that a maximum degree-constrained subgraph can be found within time $k_3m^3+k_4$, where k_3 and k_4 are constants. Use of random access computer is assumed in the computation of the time bounds.

Zur Ermittlung eines maximalen Matching in Graphen in O(|V|·|E|) Schritten. Es wird ein Algorithmus zur Bestimmung eines Matching in beliebigen Graphen mit n Knoten und m Kanten angegeben, dessen Zeitaufwand von k_1mn+k_2, wobei k_1 und k_2 Konstanten sind, beschränkt ist. Dieser Zeitaufwand entspricht der Anzahl der Schritten in einem "random access" Computer.

1. Introduction

In this paper a *graph* $G=(V, E)$ shall mean a finite undirected linear graph without loops. V is the set of *nodes* and E is the set of *edges*, which are unordered pairs of nodes. The edge (v, w), where $v, w \in V$, is said to *connect* the nodes v and w. It also *meets* (or is *incident with*) both v and w and vice versa. If two nodes (edges) meet the same edge (node), the two nodes (edges) are said to be *adjacent* to each other. A set $M \subset E$ is called a *matching*, iff no two edges in M are adjacent. A matching whose cardinality $|M|$ is maximum is called a *maximum matching*. A node is called *free* (or *exposed*) with respect to a matching M, iff it meets no edge in M. An elementary path (i.e., a path such that no node meets more than two edges in the path) such that for any consecutive pair of edges in the path, one of them is in M and the other is in $E-M$, is called an *alternating path*. A node is an *even node* relative to a fixed free node, iff there exists an alternating path of even length from the free node. An alternating path whose end-nodes are free is called an *augmenting path*. It is easy to see that if there is an augmenting path P in a graph G with a matching M, then another matching M' with $|M'|=|M|+1$ can easily be constructed by interchanging the *matching edges* (i.e., the edges belonging

* This work was supported in part by the National Research Council of Canada under grant no. A4315.

Added in proof: We recently learned that the same time bound was independently obtained by Gabor of Stanford University and Tobin of University of Iowa.

to M) of P and the non-matching edges of P. This operation is called an *augmentation*. Berge [2, 3] is credited with the discovery of the following theorem: *A matching is maximum iff there is no augmenting path.*

Based on this theorem Edmonds [4], Witzgall and Zahn [7], and Balinski [1] devised algorithms for constructing a maximum matching for a given graph. A close examination of each of these algorithms reveals that they all require time proportional to mn^2, where $m=|E|$ and $n=|V|$. In arriving at the time requirements of algorithms, we assume, as is customary, the use of so-called *random-access computer* model, in which data storage and retrieval, arithmetic and logical operations, and comparisons are all assumed to require fixed amounts of time. A graph is assumed to be represented by the *adjacency structure* [6], in which each node has a list of all nodes that are adjacent to it.

2. Informal Description

In this paper we follow the approach of Witzgall and Zahn [7] in that we search the graph in the depth-first manner, starting at a free node called the *root*. More specifically, given a graph $G(V, E)$ with a matching M, which may be empty, we first choose a free node to be the root of a labeled subgraph which is to be constructed within the graph, by labeling some nodes of the graph. We "grow" the labeled subgraph at the even nodes which are already in the subgraph. The root is labeled with the number 0 and successively increasing integers are given to subsequently encountered nodes. Consider an edge connecting an even node in the labeled subgraph and a node not in the subgraph. Since this edge is necessarily a non-matching edge, if the new node met by this edge is free, then an augmenting path has been found from the root to this new node. Otherwise there is a matching edge which connects this node and some other node, which is now found to be an even node. Thus the labeled subgraph now has two additional edges and two additional nodes.

If a non-matching edge connects an even node v to another even node u, both in the labeled subgraph, then there is a closed alternating path of odd length from a node b back to itself which contains both u and v. Such a closed path is called a *blossom*. The node b is the only node in this closed path that meets only non-matching edges of the path. b is called the *base* of the blossom. Note that all the nodes in a blossom are even nodes. We shall associate two numbers, $p_0(v)$ and $p_e(v)$, with each node v, which take integer values larger than or equal to -1. If $p_e(v) \geq 0$, then it means that there exists an alternating path of even length from r to v, i.e., v is an even node, the edge $(p_e(v), v)$ being the last edge in the alternating path. Similarly if $p_0(v) \geq 0$, then it means that there exists an alternating path of odd length from the root r to v, $(p_0(v), v)$ being the last edge or the final segment (consisting of more than one edge) of the alternating path.

In the following informal description of our algorithm, we use two first-in last-out *pushdown stacks* $S1$ and $S2$. $S1$ records (parts of) the alternating path from the root to the current node, and $S2$ stores blossom-nodes for further examinations. The *top node* means the node corresponding to the number at the

top of the stack in question. If any vague point arises in the following description, the reader is advised to follow the example in the next section to clarify the point.

A. Initialization: Pick a free node as the root r, let $u=r$, and put $i=0$ in $S1$.

B. Continue growing the labeled subgraph from u, assigning the number $i=i+1$ to each new node encountered, and putting the number in $S1$.

C. IF an augmenting path is found, THEN stop, ELSE continue.

D. IF a blossom is created, THEN go to F,
ELSE IF there is an unexamined edge (matching edge if the top node is not an even node) that meets the current top node u of $S1$, THEN go to B,
ELSE go to E.

E. IF the number on top of $S1$ has the # marker, THEN delete the top number of $S1$ and $S2$ and go to G.
ELSE delete the top number from $S1$, let $u=$the new top node of $S1$, and go to B.

F. Complete appropriately the labels (i.e., replace the -1 entry of each node with the number of its "predecessor") of all the nodes corresponding to the numbers in $S1$ lying above the number of the base of the blossom and move these numbers from $S1$ to $S2$.

G. IF $S2\neq\emptyset$, THEN let $u=$the top node of $S2$, put $NUM(u)\#$ in $S1$, and go to B,
ELSE IF $S1\neq\emptyset$, THEN let $u=$the top node of $S1$ and go to B, ELSE stop.

If the above algorithm stops at step G and there remain one or more nodes that have not been examined, restart at A, with a new free node. Then eventually either an augmenting path will be found or all the nodes will be examined without generating an augmenting path. In the latter case the matching M under consideration is maximum and in the former case we can find a new matching M' with $|M'|=|M|+1$ by augmentation. Starting with $M=\emptyset$, repeated applications of the algorithm will eventually produce a maximum matching.

3. Example

In Fig. 1, we have an example of a graph $G(V, E)$ with a matching M. The solid lines indicate the edges in M and the dotted lines the edges in $E-M$. In order to find an augmenting path, we first choose an arbitrary free node as the root, which is to be the starting point of alternating paths. Fig. 1 (a) shows the first nine nodes of an alternating path with $[p_0(v), p_e(v)]$ pair attached to each node v. Note that for the node 8, neither $p_0(8)$ nor $p_e(8)$ is -1, because of the edge (8, 4) leading to the base 4 of the blossom 456784. Stack $S1$ contains all the nodes we have visited so far in the order of the visits. We next move nodes from the top of $S1$ to $S2$, until the base appears at the top, determining at the same time $p_0(v)$ or $p_e(v)$ of each node moved. In Fig. 1 (b) we enter $p_e(7)$, $p_0(6)$, and $p_e(5)$, as the nodes 7, 6, and 5 are moved from $S1$ to $S2$. After this operation the node v at the top of $S2$ is copied on top of $S1$ with the marker #. It is then checked as to whether it meets an edge which has not been examined yet. If there is such an edge (v, u), then u is put on top of $S1$, and otherwise v is removed from $S1$ and $S2$. The nodes

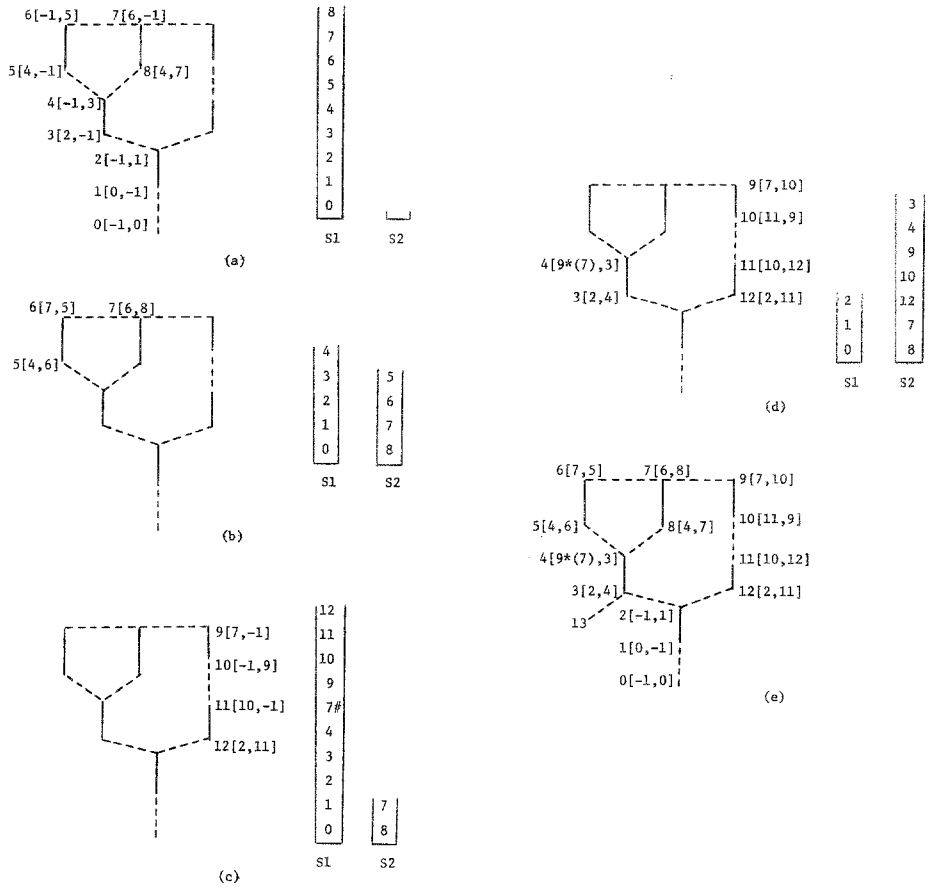

Fig. 1. Labeling process

5 and 6 are thus deleted from $S2$ and new nodes 9, 10, 11, and 12 are entered in $S1$ as shown in Fig. 1 (c). As before the edge (12, 2) leading to the base 2 of a blossom enables us to determine $p_0(12)$, $p_e(11)$, $p_0(10)$, and $p_e(9)$. Note that $p_0(4) = 9^*(7)$, the asterisk indicating the fact that an alternating path of odd length from node 4 to 0 has, as an initial segment, a path from 4 to 7 to 9. This segment can be easily determined by back-tracing from 7 to 4.

Suppose now that there is a free node 13, as shown in Fig. 1 (e). An augmenting path from 13 to 0 can be found as follows.

$$13 \rightarrow 3\,(=p_0(13)) \rightarrow 4\,(=p_e(3)) \rightarrow 9^*\,(=p_0(4)) \leftarrow 7\,(=p_0(9)) \leftarrow 8\,(=p_e(7)) \rightarrow 10\,(=p_e(9)) \rightarrow 11\,(=p_0(10)) \rightarrow 12\,(=p_e(11)) \rightarrow 2\,(=p_0(12)) \rightarrow 1\,(=p_e(2)) \rightarrow 0\,(=p_0(1))$$

4. Formal Algorithm

In the following u, x, and z represent even nodes (i.e., nodes for which there exists an alternating path of even length from r). We also use the notation $m(v)=w$, iff $(v, w) \in M$.

0. $i:=0$. Pick a free node r as a root and $NUM(r):=i$, $p_0(r):=-1$, $p_e(r):=0$. Place i in $S1$ and $u:=r$.

1. IF there is an edge $(u, v) \notin M$ that has not been examined, THEN go to 3, ELSE go to 2.

2. IF $NUM(u)$ in $S1$ has the marker $\#$, THEN delete $NUM(u)$ from $S1$ and $S2$ and go to 8,
ELSE delete the top two numbers of $S1$, let u be the new top node of $S1$ and go to 1.

3. IF $v \neq r$ and free, THEN an augmenting path from r to v has been found (stop),
ELSE IF v has not been numbered, THEN go to 4,
ELSE go to 5.

4. $NUM(v):=i:=i+1$, $p_0(v):=NUM(u)$, $p_e(v):=-1$, place i on top of $S1$, $NUM(m(v)):=i:=i+1$, $p_e(m(v)):=NUM(v)$, $p_0(m(v)):=-1$, place i on top of $S1$, $u:=m(v)$, and go to 1.

5. IF $p_e(v)=-1$, THEN go to 1,
ELSE (new blossom has been created) $z:=v$ and continue.

6. $x:=z$ and let y and z be such that $NUM(y)=$top entry of $S1$, $NUM(z)=$2nd top entry of $S1$, if they exist.
IF $NUM(y) \leq NUM(v)$, THEN go to 8,
ELSE IF $NUM(y)$ has no $\#$ marker, THEN $p_0(y):=NUM(x), p_e(z):=NUM(y)$, move $NUM(y)$ and $NUM(z)$ from $S1$ to $S2$, and repeat 6, ELSE delete $NUM(y)$ $\#$ from $S1$ and go to 7.

7. IF $NUM(z)>NUM(v)$, THEN $p_0(z):=NUM(x)*(NUM(y))$, $z:=m(z)$, and go to 6
ELSE go to 8.

8. IF $S2 \neq \emptyset$, THEN $u:=$ top node of $S2$, place $NUM(u)$ $\#$ on top of $S1$, and go to 1,
ELSE IF $S1=\emptyset$, THEN $u:=$ top node of $S1$ and go to 1,
ELSE stop.

5. Analysis of the Algorithm

In this section we shall prove the correctness of our algorithm and also establish a time bound. Since our algorithm discards a node when all the edges meeting it have been examined, we have:

Lemma 1

The algorithm of section 4 eventually terminates.

The following theorem states that even nodes are correctly identified by the algorithm.

Theorem 1

Unless the algorithm stops at step 3, *each even node v relative to the root eventually gets label* $p_e(v) \geq 0$.

Proof: If the algorithm doesn't stop at step 3, it must stop at step 8 because of Lemma 1. Therefore assume that the algorithm terminates at step 8, leaving some even nodes unlabeled. Let w be one of them with the shortest alternating path of even length from the root r. Let this path be P with length l. Then it follows that all even nodes for which there exist alternating paths of length $\leq l-2$, have labels $p_e(\ ) \geq 0$. Now consider the path P, which starts at the root r and ends at the node w. Let u be the even node which is connected by the last two edges of P to w. During the application of the algorithm $NUM(u)$ would have been placed on top of $S1$ as long as there was some unexamined edge meeting it. Thus $w=m(v)$ would have been labeled in step 4, after going through steps 1 and 3. This is a contradiction to the initial assumption. Q.E.D.

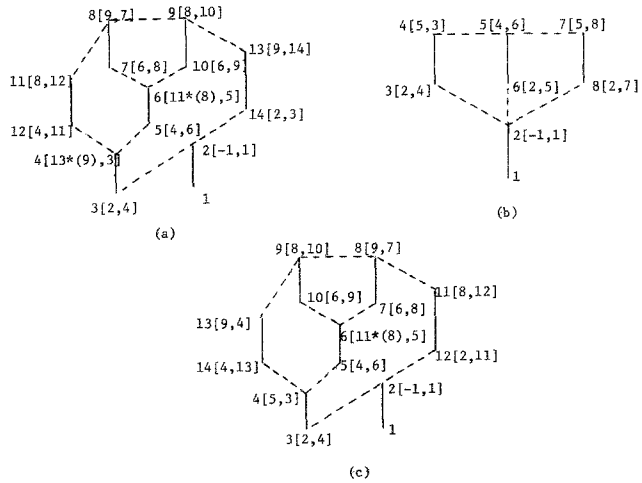

Fig. 2. Blossoms of Depth 1, 2, and 3

In order to prove our final result (Theorem 2), we have to introduce a few more definitions. In the process of growing a labeled subgraph, following the steps of our algorithm, a blossom may be created, which contains other blossoms. Take Fig. 2(a) for instance. The blossom $B1=6789106$ is created first. Then the blossom B_2 containing B_1 and nodes 4, 5, 11 and 12, is created. We call B_1 a *subblossom* of B_2, and B_2 is said to be a *nested blossom*. B_1 and B_2 have "depth" 1 and 2, respectively. Note that $p_0(6)=11^*(8)$ is determined when B_2 is created by means of the edge (12, 4). More precisely, a blossom which contains no other blossoms is said to be nested to depth 1, and a blossom *nested to depth d* (or *of depth d*) is created, if the closing of an alternating path determines the $p_e(\ )$ value of at least one subblossom and the maximum depth among those subblossoms is $d-1$. If

an alternating path leads from a node of a blossom nested to depth $d-1$ to another node of the same blossom, then the newly created blossom is nested only to depth $d-1$.
According to the above definition of the depth, the creation of a blossom by adding the edge (8, 2) to the subgraph in Fig. 2 (b) does not increase the depth. Note also that the depth of a blossom depends on the order in which nodes were labeled. For example, the outermost blossom in Fig. 2 (c) is nested to depth 2, whereas that in Fig. 2 (a) is nested to depth 3.

Lemma 2

If the altorithm stops at step 3, *then we can identify the augmenting path in time proportional to the number of edges in the path.*

We want to trace the labeled subgraph obtained during the application of the algorithm from v to r. In this process we may encounter some blossoms. We may enter a blossom via its base b or one of its non-base nodes u. Without loss of generality, we can always assume that we enter a blossom via a non-base edge, since otherwise from $p_0(b)$, which is of the form $NUM(x)*(NUM(u))$ (see step 7 of the algorithm), the path in question can be traced, starting at node u. Lemma 3 below will suffice to prove Lemma 2.

Lemma 3

Given a nested blossom B with base b and any $v(\neq b)\in B$, an alternating path of even length from u to b (within B) can be identified from the labeled graph in time proportional to the number of edges in the path.

Proof: By induction. If B is a blossom of depth 1, then trace $p_e(u)$, $p_0(p_e(u))$, $p_e(p_0(p_e(u)))$, ..., until we reach b. Assume as the induction hypothesis that the assertion is true for all blossoms of depth $\leq k-1$. Let B be a blossom of depth k. Let S be the set of nodes that were added to B, when B was created from subblossom(s) of depth $\leq k-1$. Let v be the frist node in the sequence, $p_e(u)$, $p_0(p_e(u))$, ..., such that $v\in B-S$. (In case there is no such v, b will appear in the sequence and the alternating path will be thus found.) Then v belongs to a subblossom B_1 (of B) of depth $\leq k-1$. Let b_1 be the base of B_1. There are two cases.

(a) $v\neq b_1$.

By induction hypothesis, an alternating path from v to b_1 will be found within B_1, in time proportional to the number of edges in the path.

(b) $v=b_1$.

In this case there exists a node x such that $p_0(b_1)=NUM(x)*(NUM(y))$ and $x\in S$. Therefore the problem is now reduced to finding an alternating path y to b_1 within a subblossom of depth $\leq k-1$ and another from x to b. From our construction, the path from v to y will have no node in common with the path from x to b. In either case above, we can repeat the argument until we reach b. Q.E.D.

Theorem 2

A maximum matching can be found within time $k_1 mn + k_2$, where k_1 and k_2 are constants.

Proof: Starting with a given graph $G(V, E)$ with matching $M = \emptyset$, apply our algorithm repeatedly. In order to find an augmenting path, we may have to start our algorithm more than once. But since each edge is examined at most twice (note that each edge is represented twice in the adjacency structure), the total time spent to find an augmenting path is bounded by $c_1 m$, where $m = |E|$, and c_1 is a positive constant. Also the time needed to *identify* the edges of an augmenting path, in order to augment a matching, is bounded by $c_2 m$, where c_2 is a constant (Lemma 2). A maximum matching will be reached after at most $n/2$ augmentations. Q.E.D.

A *degree-constrained subgraph* in a graph $G(V, E)$ is a subgraph $G'(V, M)$ of G, where $M \subset E$, such that for each $v \in V$, v meets no more than $c(v)$ edges in M, where $c: V \to N$ (the set of natural numbers) is a given function. Such a set M is called a *c-matching* in [3]. Thus the matching we defined before is a special case where $c(v) = 1$ for all $v \in V$. Berge [3] shows that the problem of finding a maximum degree-constrained subgraph in a graph $G(V, E)$ with $n = |V|$ and $m = |E|$ can be reduced to the maximum matching problem in a graph with less than $4m$ nodes and $4m(4m-1)/2$ edges. The following is also valid for *multigraphs* with multiple edges allowed between any pair of nodes but no loops.

Corollary: A maximum degree-constrained subgraph can be found within time $k_3 m^3 + k_4$, where k_3 and k_4 are constants.

References

[1] Balinski, M. L.: Labelling to obtain a maximum matching. Proc. Conf. Comb. Math. and Its Applications. Univ. North Carolina, April 1967.
[2] Berge, C.: Two theorems in graph theory. Proc. Natl. Acad. Sci. **43**, 842—844 (1957).
[3] Berge, C.: Alternating chain methods: A survey, in "Graph Theory and Computing" (Read, R. C., ed.), New York-London: Academic Press. 1972.
[4] Edmonds, J., Paths, trees, and flowers. Can. J. Math. **19**, 449—467 (1965).
[5] Hopcroft, J. E., and R. M. Karp: A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. Record of the 12th IEEE Annual Symp. on Switching and Automata, pp. 122—125. Oct. 1971.
[6] Tarjan, R. E.: Depth-first search and linear graph algorithms. SIAM J. Computing **1**, 146—159 (1972).
[7] Witzgall, C., and C. T. Zahn, Jr.: Modification of Edmonds' maximum matching algorithm. J. Res. Nat. Bur. Stds. **69B**, 91—98 (1965).

Prof. Dr. T. Kameda
Department of Electrical Engineering
Prof. Dr. I. Munro
Department of Applied Analysis and
Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

Eigentümer, Herausgeber und Verleger: Springer-Verlag, A-1011 Wien, Mölkerbastei 5. — Für den Textteil verantwortlich: Dr. W. Schwabl, A-1011 Wien, Mölkerbastei 5. — Für den Anzeigenteil verantwortlich: Bruno Schweder, A-1130 Wien, Schweizertalstraße 8–10. — Satz: Austro-Filmsatz Richard Gerin, A-1020 Wien, Fischergasse 1. Druck: Paul Gerin, A-1021 Wien, Zirkusgasse 13.
Printed in Austria.