DETERMINISTIC AND PROBABILISTIC ALGORITHMS FOR MAXIMUM BIPARTITE MATCHING VIA FAST MATRIX MULTIPLICATION *

Oscar H. IBARRA and Shlomo MORAN
Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, U.S.A.

Received 22 January 1981; revised version received 15 June 1981

Probabilistic algorithm, maximum bipartite matching, fast matrix multiplication

1. Introduction

Let $G = (S, T, E)$ be a bipartite graph, where $S \cup T$ is the set of nodes ($S \cap T = \emptyset$) and E is the set of edges, $E \subseteq S \times T$. Let $S = \{u_1, ..., u_s\}$, $T = \{v_1, ..., v_t\}$ ($s \leqslant t$), and $|E| = e$. An (S, T) matching is a subset M of E such that no two edges in M have a common endpoint.

A maximum matching is a matching of maximum cardinality. The set of nodes which take part in such a maximum matching is denoted by Nodes(G) and the cardinality of the matching is denoted by Card(G). Note that Nodes(G) is not unique, but Card(G) is unique. There is an $O(e\sqrt{s})$ algorithm to find a maximum matching [3]. When $e = O(st)$ (i.e., the graph is dense), $O(e\sqrt{s}) = O(s^{1.5}t)$.

Let $A = (a_{ij})$ be an $s \times t$ matrix over a given ring. A matching of cardinality r in A is a set of r nonzero entries of A, with at most one entry chosen from each row and column. The cardinality of a maximal matching in A is denoted by Card(A).

Let F be any field, and let $G = (S, T, E)$ be given. With each edge (u_i, v_j) in E we associate a variable x_{ij}, and with the graph G we associate a matrix $A_G = (a_{ij})$ defined by:

$$a_{ij} = \begin{cases} x_{ij} & \text{if } (u_i, v_j) \text{ in E,} \\ 0 & \text{otherwise.} \end{cases}$$

Note that there is an obvious one-to-one correspondence between the matchings in G and A_G. In particular, Card(G) = Card(A_G).

Let $F[x_{i_1 j_1}, ..., x_{i_e j_e}]$ denote the ring of polynomials with e variables over F. Then one can easily verify the following claim.

Claim 1. A_G has a matching of cardinality r if and only if the rank of A_G over $F[x_{i_1 j_1}, ..., x_{i_e j_e}]$ is greater or equal to r. The rows and columns of a maximal nonsingular square submatrix of A_G correspond to the nodes of a maximal matching in G.

In this paper, we present two algorithms for finding a maximal nonsingular square submatrix of A_G (or, equivalently, Card(G) and Nodes(G)) in $O(s^{\beta-1}t)$ arithmetic operations, where $O(n^\beta)$ is the complexity of matrix multiplication. The best bound on β currently claimed is 2.49+, and recent developments may reduce this bound further [5]. Our algorithms use the following result in [4]:

Theorem 1. There is an $O(s^{\beta-1}t)$ algorithm to find a maximal nonsingular square submatrix of any $s \times t$ matrix ($s \leqslant t$).

Note that a direct application of Theorem 1 to the matrix A_G yields an algorithm whose complexity is larger than $O(s^{\beta-1}t)$, since the elements of A_G are variables and not numbers.

In Section 2, we present a deterministic algorithm which replaces each x_{ij} in A_G by an integer, thus ob-

* This research was supported in part by NSF Grant MCS78-01736.

0020-0190/81/0000–0000/$02.75 © 1981 North-Holland

taining an integer matrix A'_G satisfying $\text{rank}(A'_G) = \text{rank}(A_G)$. The integers in A'_G are, however, very large, and hence the bitwise complexity of the algorithm (using exact arithmetic [1]) is large. In Section 3, we develop a probabilistic algorithm (for the same task) which generates random numbers for A'_G. The numbers are relatively small, and $\text{rank}(A'_G) = \text{rank}(A_G)$ with probability $>\frac{1}{2}$. Both algorithms use Theorem 1 to find a maximal nonsingular square submatrix of A'_G. The dimension and the set of rows and columns making up the maximal nonsingular square submatrix correspond to Card(G) and Nodes(G), respectively. We note that our algorithms, like other algorithms which solve graph problems by fast matrix multiplication (e.g. the shortest path problem [7,8]), do not provide the full constructive solution to the maximal matching problem since they do not construct the arcs which take part in a maximal matching. (In [7,8], $O(n^\beta)$ algorithms to find the set of all minimum distances of an n-node graph, with integer distances and without negative-cost cycles, are given. The algorithms, which also use exact arithmetic on very small numbers do not find the set of shortest paths.)

2. The deterministic algorithm

Definition. A sequence $(c_1, ..., c_e)$ of integers is a 2-*sequence* if it satisfies the following:

(a) $c_e \geqslant 2$,

(b) $c_{i-1} \geqslant c_i^2$ $(i = 2, ..., e)$.

A 2-*matrix* is a matrix whose nonzero elements, when ordered row by row, form a 2-sequence. Note that every submatrix of a 2-matrix is a 2-matrix.

Example. The following is a 2-matrix:

$$\begin{bmatrix} 0 & 18 \\ 4 & 2 \end{bmatrix}.$$

Lemma. Let A be an r × r 2-matrix such that Card(A) = r. Then rank(A) = r (i.e., A is nonsingular) *.

Proof. The proof is an induction on r, and for each r, an induction on k = number of nonzero elements in A. Clearly, for each r, $k \geqslant r$, and if k = r then A is nonsingular (since, in this case, $|\det(A)| = \Pi_{i=1}^{r} a_i$, where $a_1, ..., a_r$ are the nonzero elements of A).

* A is nonsingular if and only if $\det(A) \neq 0$.

Assume that the lemma holds for r and k $(k < r^2)$. We shall prove that it also holds for r and k + 1. To simplify the discussion we shall assume that $b = a_{11}$ is the largest element in A. Denote by A_{ij} the minor obtained by deleting row i and column j from A. We consider 3 cases:

Case 1. $\det(A_{11}) = 0$. In this case, by the induction hypothesis for r − 1, $\text{Card}(A_{11}) < r - 1$, which means that a_{11} does not belong to any matching of cardinality r in A. Hence, by substituting $a_{11} = 0$, a matrix A′ is obtained and $\text{Card}(A') = \text{Card}(A) = r$, $\det(A') = \det(A)$. Since A′ has only k nonzero entries, the lemma holds for A′ by the induction on k, and hence it also holds for A.

Case 2. a_{11} is the only nonzero element in row A_1. In this case, $|\det(A)| = a_{11}|\det(A_{11})|$, and $\text{Card}(A_{11}) = r - 1$, which implies (by induction on r) that $\det(A_{11}) \neq 0$. Hence $\det(A) \neq 0$.

Case 3. $\det(A_{11}) \neq 0$ and there are at least 2 nonzero elements in A_1. In this case (note that $b = a_{11}$):

$$|\det(A)| \geqslant b|\det(A_{11})| - \sum_{j=2}^{r} a_{1j}|\det(A_{1j})|.$$

Since $|\det(A_{11})| \geqslant 1$, $\det(A) \neq 0$ if $b > \Sigma_{j=2}^{r} a_{1j} \times |\det(A_{1j})|$.

Let $\bar{A} = (\bar{a}_{ij})$ be the matrix obtained by substituting $a_{11} = 0$ in A. Let $\ell_i = \bar{a}_{i_1} + \cdots + \bar{a}_{ir}$, and let p_i be the product of the nonzero elements in row i of $\bar{A}$. Since all the nonzero $\bar{a}_{ij}$'s are at least 2, we have that $\ell_i \leqslant p_i$. Also, since all the $\bar{a}_{ij}$'s are nonnegative

$$\sum_{j=2}^{r} a_{1j}|\det(A_{1j})| \leqslant \prod_{i=1}^{r} \ell_i .$$

Combining the inequalities, we get

$$\sum_{j=2}^{r} a_{1j}|\det(A_{1j})| \leqslant \prod_{i=1}^{r} \ell_i \leqslant \prod_{i=1}^{r} p_i$$

$$\leqslant \prod_{\bar{a}_{ij} \neq 0} \bar{a}_{ij} < b^{1/2+1/4+\cdots} = b.$$

Corollary 1. Let A be an m × n 2-matrix. Then Card(A) = rank(A). Moreover, if A′ is any r × r submatrix of A with $\text{Card}(A') = r$, then $\text{rank}(A') = r$.

Proof. Since every submatrix of a 2-matrix is a 2-matrix, Lemma 1 applies to any submatrix A′ satisfying the hypothesis above.

Let A_G be the matrix defined in the previous section. By the discussion following Theorem 1 and by Corollary 1, if the matrix A'_G obtained by replacing the variables of A_G by integers is a 2-matrix, then Card(G) and Nodes(G) can be found in $O(s^{\beta-1}t)$ arithmetic operations. Thus, we have:

Theorem 2. There is an algorithm which finds, for each bipartite graph G(S, T, E), Card(G) and Nodes(G) in $O(s^{\beta-1}t)$ arithmetic operations.

Proof. Let $x_1, ..., x_e$ be the nonzero elements in A_G, ordered by rows. Compute the integers $c_1, ..., c_e$ by: $c_e = 2$, $c_{i-1} = c_i^2$ $(i = e, e-1, ..., 2)$, and let A'_G be the matrix obtained by replacing each x_i in A_G by c_i. The result follows from the discussion above, noting that A'_G is a 2-matrix.

3. The probabilistic algorithm

Definition. Let $p(x_1, ..., x_e)$ be a polynomial with e variables. Then p is of *semi-degree 1* if it is of degree at most 1 in each of its variables.

Example. $x + xy$ is of semi-degree 1, while $x^2 + y$ is not.

Let $\bar{c} = (c_{i_1 j_1}, ..., c_{i_e j_e})$ be an e-tuple of constants from F. Then $A_G(\bar{c})$ is the matrix obtained from A_G by replacing $x_{i_k j_k}$ by $c_{i_k j_k}$ for $k = 1, 2, ..., e$.

Claim 2. Let $\text{rank}(A_G) = r$. Then there is a nonzero polynomial $p(x_{i_1 j_1}, ..., x_{i_e j_e})$ of semi-degree 1 such that for $\bar{c} = (c_{i_1 j_1}, ..., c_{i_e j_e})$, if $p(\bar{c}) \neq 0$ then $\text{rank}(A_G(\bar{c})) = r$.

Proof. Let p be the sum of all the determinants of $r \times r$ submatrices of A_G.

Lemma 2. Let $p(x_1, ..., x_e)$ be of semi-degree 1. Let C be a set of constants in F with $|C| = m$, and let $C^e = C \times \cdots \times C$ (e times). Then $p(\bar{c}) \neq 0$ for at least $(m-1)^e$ elements in C^e.

Proof. The proof is an induction on e. For $e = 1$, the lemma follows trivially from the observation that a one-variable polynomial of degree 1 has 1 root.

Assume the lemma holds for $e \geqslant 1$, and consider a nonzero polynomial $p(x_1, ..., x_e, x_{e+1})$ of semi-degree 1. Then there exists an e + 1 tuple $(c_1, ..., c_e, c_{e+1})$ such that $p(c_1, ..., c_e, c_{e+1}) \neq 0$. Hence, the polynomial $p(x_1, ..., x_e, c_{e+1}) = \hat{p}(x_1, ..., x_e)$ is nonzero and is of semi-degree 1. Hence, by induction hypothesis, there are at least $(m-1)^e$ e-tuples $(c_1, ..., c_e)$ in C^e such that $\hat{p}(c_1, ..., c_e) = p(c_1, ..., c_e, c_{e+1}) \neq 0$. For each such tuple, the one-variable polynomial $\tilde{p}(x_{e+1}) = p(c_1, ..., c_e, x_{e+1})$ is nonzero and of semi-degree 1. Hence, there are at least $m - 1$ elements c_{e+1} in C such that $\tilde{p}(c_{e+1}) = p(c_1, ..., c_e, c_{e+1}) \neq 0$. The lemma follows.

Note. It was pointed out by the referee that a result similar to Lemma 2 appeared in [2].

Corollary 2. Let G, A_G be as above, and let C be of cardinality at least 2e. Let $\bar{c}$ be an element chosen at random from C^e. Then $\text{rank}(A_G(\bar{c})) = r$ with probability $\geqslant 1/2$.

Proof. Let $|C| = m$. Then by Claim 2 and Lemma 2 above, $\text{rank}(A_G(\bar{c})) = r$ for at least $(m-1)^e$ elements in C^e. Assuming that each $\bar{c}$ has the same probability of being chosen, we get that this probability is not smaller than

$$\frac{(m-1)^e}{m^e} = \left(1 - \frac{1}{m}\right)^e \geqslant \left(1 - \frac{1}{2e}\right)^e \geqslant 1/2.$$

The probabilistic algorithm for maximal bipartite matching is given below. F is a field containing at least 2e elements, and $C \subseteq F$, $|C| = 2e$.

Step 1. Generate a random $\bar{c}$ in C^e.

Step 2. Find a maximal nonsingular square submatrix in $A_G(\bar{c})$. Denote this submatrix by $\hat{A}$.

Step 3. The dimension and the set of rows and columns of $\hat{A}$ correspond to Card(G) and Nodes(G), respectively. Stop.

The probability that the output matching is maximum $\geqslant 1/2$. This follows from the fact that $\text{rank}(A_G(\bar{c})) = \text{rank}(A_G)$ with probability $\geqslant 1/2$, and that the cardinality of the matching is equal to $\text{rank}(A_G(\bar{c}))$.

By running the algorithm above k times we have

Theorem 3. There is a probabilistic algorithm which

finds for a bipartite graph, G = (S, T, E), Card(G) and Nodes(G) with probability $\geqslant(1 - 1/2^k)$ in time $O(ks^{\beta-1}t)$, where s = | S | and t = | T |.

Remark. The bitwise complexity of the algorithm above (i.e., the number of bit operations required to execute it – see [1]) may depend on the field F: Step 2 of the algorithm may require inversion of order s matrices [4]. Hence, if we take F to be the field of rational numbers (with C = {1, 2, ..., 2e}, say), then numbers whose representations are as large as sloge may occur during the computation. Perhaps a better idea is to take F = GF(p) for some prime $p \geqslant 2e$. To do this we first generate random numbers between 2e and $2e^2$ and test them for primality by a probabilistic algorithm (see, e.g., [6]), until a prime p is found. This requires time which is polynomial in log e. The computation is then carried out modulo p. The bitwise complexity of multiplication (mod p) is at most $O(\log^2 p)$. Finding a^{-1} (mod p) can be carried out by applying a gcd algorithm which finds integers x and y such that ax + py = 1. a^{-1} (mod p) is equal to x (mod p). Using Euclid's algorithm for gcd requires $O(\log^3 p)$ bit operations, which gives an $O(n^\beta \log^3 n)$ time bound for the probabilistic algorithm. This bound can be improved slightly by using fast algorithms for integer arithmetic (see e.g. [1], sects. 7, 8).

Acknowledgement

We would like to thank the referee for suggestions which improve the presentation of our results.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, The Design and Analysis of Computer Algorithms (Addisson-Wesley, Reading, MA, 1974).

[2] M. Blum, A.K. Chandra and M.N. Wegman, Equivalence of free Boolean graphs can be decided probabilistically in polynomial time, Information Processing Lett. 10 (2) (1980) 80–82.

[3] J.E. Hopcroft and R.M. Karp, An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs, SICOMP 2 (4) (1973) 225–231.

[4] O.H. Ibarra and S. Moran, A generalized fast matrix decomposition algorithm and applications, University of Minnesota, Dept. of Computer Science, Tech. Rep. No. 80-20 (1980).

[5] V. Pan, New combinations of methods for the acceleration of matrix multiplication, unpublished manuscript.

[6] M.O. Rabin, Probabilistic algorithms, in: J. Traub, Ed., Algorithms and Complexity: New Directions and Recent Results (Academic Press, New York, 1976) pp. 21–39.

[7] F. Romani, Shortest path problem is not harder than matrix multiplication, Information Processing Lett. 11 (3) (1980) 134–136.

[8] G. Yuval, An algorithm for finding all shortest paths using $N^{2.81}$ infinite precision multiplications, Information Processing Lett. 4 (6) (1976) 155–156.