Some Matching Problems for Bipartite Graphs

ALON ITAI

Technion–Israel Institute of Technology, Haifa, Israel

MICHAEL RODEH

IBM Israel Scientific Center, Haifa, Israel

AND

STEVEN L. TANIMOTO

University of Connecticut, Storrs, Connecticut

ABSTRACT Certain applications require finding a perfect matching in a bipartite graph that satisfies some additional properties For one such type of restriction the problem is proven to be NP-complete. If for a single subset of edges no more than r edges may be included in the matching then an $O(ne)$ algorithm is given

An efficient algorithm for finding all perfect matchings is presented It requires $O(e)$ time per matching and a total of $O(e)$ space This algorithm may be used to calculate the permanent of a matrix Finally, the algorithm is generalized to find all maximum matchings

KEY WORDS AND PHRASES matching, labeling, bipartite graph, maximum matching, restricted matching, flow network, permanent of a matrix, NP-complete, maximum matchings generator

CR CATEGORIES 5 25, 5 32

1. *Introduction*

Matching problems arise in several areas of automatic data processing, including analysis of images, artificial intelligence, and the solution of school scheduling problems. In attempts to find practical solutions to problems of interpreting chest X-ray images, matching problems may be posed in terms of pairing labels (designating organ names—lungs, heart, ribs, etc.) with regions (detected by a segmentation algorithm). In such an approach a bipartite graph is constructed that describes feasible pairings between labels and regions [19]. For example, for region r and label *heart* the edge $(r, heart)$ is included if the size s_r of r satisfies $a_{heart} \leq s_r \leq b_{heart}$. A solution to the image analysis problem can be considered as a maximum matching (in the constructed graph) which satisfies some additional constraints (based upon relationships among interpreted regions).

In the artificial intelligence literature, solution methods based on a labeling technique have been described for several problems, including the n-queens problem, detecting graph isomorphism, and finding consistent labelings for the junctions in line drawings. The technique used is commonly referred to as relaxation labeling (also Waltz filtering or

The work of the third author was supported by the National Science Foundation under Grant ENG76-09924

Authors' present addresses A Itai, Computer Science Department, Technion–Israel Institute of Technology, Haifa, Israel, M Rodeh, IBM Israel Scientific Center, Haifa, Israel, S L Tanimoto, Department of Computer Science, University of Washington, Seattle, WA 98195

constraint propagation) [21, 17, 9]. Relaxation labeling is a technique whereby the sets of potential labels for each entity are successively replaced with smaller sets according as some labels are found to be inconsistent with neighboring entities. Relaxation labeling is efficient when the constraints to be satisfied are local (e.g. the constraints involve at most two entities at a time). However, the enforcement of certain global constraints cannot be done efficiently using relaxation labeling. For example, determination of a labeling satisfying the global constraint "exactly m of the n entities shall be labeled x" (where x is one of d possible labels) requires $O(\binom{n}{m}d^m)$ space using the general algorithm of Freuder for any $m \leq n/2$, since all partial solutions of size m are consistent with the global constraint and must be carried forward toward computing a complete solution [9]. Such a constraint may arise when an industrial robot is given knowledge that m out of n parts appearing on the conveyor belt are piston rods, for example. Global constraints on the number of objects receiving a given label can often be handled efficiently with matching techniques by simply providing exactly m labels of type x. In any perfect matching, the m labels will all be assigned. Consequently matching techniques are sometimes to be preferred over relaxation techniques for solving problems in artificial intelligence.

Another source of matching problems arises in attempts to find practical solutions to the school-scheduling problem (the timetable problem). This problem is known to be NP-complete [6]. (A discussion of NP-completeness appears in [1].) A solution can be considered as a matching between classes and teachers that satisfies certain restrictions (e.g. not more than two labs at the same time). Having this in mind, we look for a maximum matching in a bipartite graph with restrictions. (See [5] for the properties of graphs and matchings.) This problem is shown to be NP-complete, and offers an explanation why matching approaches to scheduling are unsuccessful. If no restrictions are present, then a maximum matching may be found in $O(n^{1/2}e)$ time [13] (n is the number of vertices and e the number of edges). For the case of a single restriction, an $O(ne)$ algorithm is presented. The algorithm finds a minimum cost maximum flow for an auxiliary network by augmenting along minimum weight paths [8]. In our case the algorithm due to Edmonds and Karp [4] may be simplified and implemented more efficiently using the shortest path finding techniques of Wagner [20].

The restricted matching problem and also certain matching problems which result from image analysis may be solved by enumeration. Hence, an algorithm is presented for producing a sequence of all perfect matchings. It has the property that at most $O(e)$ time passes between the emission of two successive matchings. This compares favorably with Gal and Breitbart's algorithm [12] where $O(n^3)$ time may elapse without getting a new solution. The third author has developed a method for enumeration of maximum matchings for a general graph in order to analyze some biomedical images [19]. However, that algorithm may require exponential space and need not produce a new maximum matching within polynomial time.

The above enumeration technique may be used to obtain an algorithm for finding the permanent of a matrix. See Ryser [18] for the properties of the permanent.

Finally, we generalize the enumeration algorithm to yield an algorithm for finding all maximum matchings in a bipartite graph. The behavior of this generalized algorithm is similar to that of finding perfect matchings.

2. *The Restricted Matching Problem Is NP-Complete*

A graph $B = (V, E)$ is *bipartite* if V is partitioned into two disjoint sets, X and Y; all edges have one endpoint in X and one endpoint in Y. A set $M \subseteq E$ is a *matching* if no vertex is incident with more than one edge of M. The size of a matching M is the number of its edges. If $|X| \leq |Y|$ and every vertex $x \in X$ is incident with an edge of M then the matching is *complete*. If in addition $|X| = |Y|$ then the matching is *perfect*. Hopcroft and Karp [13] find a *maximum matching* (a matching of maximum size) in $O(n^{1/2}e)$ time.

Let $E_1, \ldots, E_k$ be subsets of E, $r_1, \ldots, r_k$ positive integers. The *restricted complete*

matching problem (RCM) is to determine whether there exists a complete matching M for B which also satisfies the restrictions:

$$|M \cap E_j| \leq r_j \quad \text{for } j = 1, \ldots, k.$$

THEOREM 1. *RCM is NP-complete*

PROOF. It is easy to see that RCM belongs to the class NP. We show a reduction from the satisfiability problem of Boolean expressions to RCM. Let $\psi = C_1 \cdots C_p$ be a Boolean expression in conjunctive normal form the variables of which are $(x_1, \ldots, x_q)$. Define the bipartite graph $B = (X \cup Y, E)$ as follows:

$X = \{C_1, \ldots, C_p\}$;
$Y = \{x_1, \ldots, x_q, \bar{x}_1, \ldots, \bar{x}_q\} \times \{1, \ldots, p\}$;
$E = \{(C_j, (x_i, j)) | x_i \text{ appears in } C_j\} \cup \{(C_j, (\bar{x}_i, j)) | \bar{x}_i \text{ appears in } C_j\}$.

Let M be a complete matching in B If M does not include two edges $(C_j, (x_i, j))$, $(C_k, (\bar{x}_i, k))$ then it corresponds to an assignment which satisfies ψ. Therefore, the following restrictions are imposed:

$$E_{ijk} = \{(C_j, (x_i, j)), (C_k, (\bar{x}_i, k))\} \cap E,$$
$$r_{ijk} = 1$$

Those E_{ijk} whose cardinality is less than 2 may be ignored.

We now show that the RCM problem B with the restrictions E_{ijk} has a solution if and only if ψ is satisfiable.

(i) Assume that ψ is satisfiable. For each clause C_j choose a single literal y ($y = x_i$ or $y = \bar{x}_i$) such that y is true If $y = x_i$ then include the edge $(C_j, (x_i, j))$ in the matching M. If $y = \bar{x}_i$ then include $(C_j, (\bar{x}_i, j))$ in M. Thus, M is a complete matching. Since either x_i or $\bar{x}_i$ is true but not both $|M \cap E_{ijk}| \leq 1$.

(ii) Let M be a solution to the RCM problem. The truth value of x_i is assigned as follows If $(C_j, (x_i, j)) \in M$ for some j then x_i is given the value true. Otherwise, x_i is given the value false. The restrictions imply that the value of x_i is well defined (x_i and $\bar{x}_i$ cannot both be true) Every vertex C_j is matched in M with some vertex y. From the construction, the literal y is true and the clause C_j is satisfied. Thus, ψ is satisfiable. Q.E.D.

The *restricted perfect matching problem* (RPM) is a special case of RCM when the graph $B = (X \cup Y, E)$ satisfies $|X| = |Y|$.

THEOREM 2. *RPM is NP-complete.*

PROOF We show a reduction from RCM to RPM.

Let $B = (X \cup Y, E)$ and $E_1, \ldots, E_k, r_1, \ldots, r_k$ be an instance of RCM Define the bipartite graph $B' = ((X \cup X') \cup Y, E \cup E')$ as follows: $X' = \{v_1, \ldots, v_t\}$ is a set of new vertices where $t = |Y| - |X|$ and $E' = \{(v_i, y) | v_i \in X', y \in Y\}$. It is easy to see that B has a complete matching if and only if B' has a perfect matching Q.E.D.

Given an integer $p \leq |X|$, the *restricted maximum matching problem* (RMM) is the problem of determining whether there exists a matching M such that

$$|M| \geq p, \quad |M \cap E_j| \leq r_j, \quad j = 1, \ldots, k.$$

It is easy to see that RMM is also NP-complete

3. *Finding a Complete Matching with a Single Restriction*

The general RCM problem is NP-complete. However, if there exists only one restriction (the RCM1 problem) a solution can be found in $O(ne)$ time. Let $B = (X \cup Y, E)$ and E_1, r_1 be an instance of RCM1. We reformulate the problem in terms of a network N for which a maximum flow of cost less than or equal to r_1 is sought [8]. N consists of a directed graph $G = (V, A)$, capacities $c(u, v)$, and costs $d(u, v)$ assigned to the edges:

(i) $V = X \cup Y \cup \{s, t\}$ where $s, t \notin X \cup Y$;
(ii) $A = \{(s, x) | x \in X\} \cup \{(y, t) | y \in Y\} \cup \{(x, y) | (x, y) \in E, x \in X, y \in Y\}$;

(iii) $c(u, v) = 1$ for $(u, v) \in A$;

(iv) $d(u, v) = \begin{cases} 1 & (u, v) \in E_1, \\ 0 & \text{otherwise.} \end{cases}$

Let f be a flow function; its value $|f|$ is $\sum_{x \in X} f(s, x)$ and its cost is $\sum_{(u,v) \in A} f(u, v)d(u, v)$.

A complete matching corresponds to a flow of value $|X|$ from s to t. The cost of the flow is equal to the number of edges of E_1 in the matching. Therefore, a minimum cost maximum flow for the network N yields a matching which uses the minimum number of edges of E_1. This number is less than or equal to r_1 if and only if there exists a solution to the given RCM1 problem.

In order to solve the minimum cost maximum flow problem we follow [8] and [4].

A flow f is *extreme* if its cost is minimum among all flows of the same value. The zero flow is extreme.

For a given flow f, the network N^f is defined as follows:

(i) $G^f = (V, A^f)$;
$A^f = \{(u, v) | (u, v) \in A, f(u, v) = 0\} \cup \{(u, v) | (v, u) \in A, f(v, u) = 1\}$;

(ii) $c^f(u, v) = 1$, $(u, v) \in A^f$;

(iii) $\Delta(u, v) = \begin{cases} d(u, v) & (u, v) \in A, \\ -d(v, u) & (v, u) \in A. \end{cases}$

THEOREM 3 ([8, p. 121]). *If f is extreme and P a path of minimum weight in N^f from s to t then a flow f' obtained by augmenting along P is extreme.*

The above theorem suggests a method for solving the minimum cost maximum flow problem: Start with an extreme flow $f^0 = 0$; compute f^{k+1} from f^k ($k = 0, 1, \ldots$) by augmenting along any one of the shortest paths from s to t in N^{f_k} (with respect to the weights Δ). N^{f_k} might contain negative weights. Since it is more efficient to find shortest paths in a graph with nonnegative weights, Edmonds and Karp [4] introduce auxiliary weights $\Delta^k(u, v) \geq 0$. These weights are obtained from the original weights $\Delta(u, v)$ and a vertex labeling function $\pi^k(u)$ to be defined in the following algorithm:

```
procedure MIN_COST_MAX_FLOW_FOR_RCM1,
begin f^0 := zero flow; π^0 := zero labeling function,
  for k := 0 step 1 until n − 1 do
    begin determine f^{k+1} by augmenting along any one of the shortest paths from s to t in N^{f_k} with respect to the
          (nonnegative) weights

                    Δ^k(u, v) = π^k(u) + Δ(u, v) − π^k(v);

      for v ∈ V do
        begin σ^k(v) := the weight of the shortest path from s to v,
          π^{k+1}(u) := π^k(u) + σ^k(u),
          comment if u is inaccessible from s then σ^k(u) = ∞
        end
    end
end
```

Finding shortest paths is the most time consuming part of the algorithm. Edmonds and Karp [4] present an $O(n^2)$ solution. The following discussion leads to an $O(e)$ algorithm.

THEOREM 4 [4]. *For each k and u, $\pi^k(u)$ is the weight of a shortest path from s to u in N^{f_k} with respect to the weights $\Delta(u, v)$ and $\pi^{k+1}(u) \geq \pi^k(u)$.*

COROLLARY. *During the execution of MIN_COST_MAX_FLOW_FOR_RCM1 the weights of shortest paths (σ^k) are bounded by n.*

PROOF. By Theorem 4, $\pi^k(u)$ may be expressed as a sum of at most n weights Δ. Since $\Delta(u, v) \leq 1$, $\pi^k(u) \leq n$. Moreover, $0 \leq \pi^k(u)$. By the construction $\sigma^k(u) = \pi^{k+1}(u) - \pi^k(u)$ Thus $0 \leq \sigma^k(u) \leq n$. Q.E.D.

Following Dijkstra's algorithm [3], let S be a set of vertices whose distance from s is known. For the remaining vertices $\bar{S}$ only a tentative distance is known To start $S := \emptyset$;

$\bar{S} := V$; $\delta(s) := 0$; $\delta(v) := \infty$ for $v \in \bar{S} - \{s\}$ At each stage, we find a vertex $v \in \bar{S}$ whose tentative distance $\delta(v)$ is minimum and transfer it to S. Then we use $\delta(v)$ to update the tentative distances of the vertices of $\bar{S}$ adjacent to v.

If the weights are nonnegative, then $\delta(u)$ is the weight of a minimum path from s to u. The main problem is to find a vertex $v \in \bar{S}$ whose tentative distance is minimum. Dijkstra suggested searching sequentially through $\bar{S}$. This yields an $O(n^2)$ algorithm. Using a balanced tree for keeping the tentative distances leads to an $O(e \log n)$ algorithm [16]. We take advantage of the fact that in our case the distances are integers between 0 and n. Following Wagner [20] we keep the vertices of $\bar{S}$ in a vector of buckets (the ith bucket contains a list of vertices whose tentative distance is equal to i). We search through the vector for the first nonempty bucket. A vertex v of minimum tentative distance is found in that bucket. Since the distances are nonnegative the index of the first nonnegative bucket does not decrease. Therefore, the entire search through the vector requires $O(n)$ time. Updating tentative distances is done by moving vertices from buckets, an operation which takes constant time and occurs at most e times. Hence, the algorithm requires at most $O(e + n)$ time and $O(n)$ space in addition to the input.

Once the distances to all vertices are known, a shortest path from s to t may be found in linear time. Since there may be no more than $|X|$ augmenting paths we have:

THEOREM 5. *MIN_COST_MAX_FLOW_FOR_RCM1 requires at most $O(ne)$ time.*

RCM may be defined for general graphs. For such graphs RCM1 can be solved in $O(n^3)$ time by finding a maximum matching with a minimum cost [11].

4. *Finding All Perfect Matchings*

Let $B = (X \cup Y, E)$ be a bipartite graph such that $|X| = |Y| = n$. A maximum matching may be found in $O(n^{1/2}e)$ time [13]. A matching is perfect if it contains n edges. We use circuits to produce all perfect matchings one by one. A circuit C in B is an *M-alternating circuit* if for any two adjacent edges of C exactly one is in M.

LEMMA 1. *A perfect matching M is not unique if and only if there exists an M-alternating circuit.*

PROOF. Let M' be another perfect matching. Consider the graph $H = (X \cup Y, M' \oplus M)$ ($\oplus$ denotes the symmetric difference). Since $M \neq M'$ there exists a vertex v of positive degree in H. The degree of v is at most two (each perfect matching can contribute at most one to the degree of v). If the degree of v in H is one then only one edge in $M \oplus M'$ is incident with v. Assume that this edge belongs to M. Then it does not belong to M'. Moreover, none of the edges of M' is incident with v. This contradicts the hypothesis that both matchings are perfect. Consequently, the nontrivial connected components of H consist of disjoint M-alternating circuits.

If M is a perfect matching and C an M-alternating circuit then $M' = M \oplus C$ is another perfect matching. Q.E.D.

We use the auxiliary directed graph $D = (X, E')$ to find an M-alternating circuit in B where $E' = \{(u, v) | u, v \in X, \exists\, w \in Y: (u, w) \in M \text{ and } (v, w) \in E\}$. (An edge in E' originates from two adjacent edges in E, the first of which belongs to M.) A bipartite graph B with a perfect matching M and the corresponding directed graph D are illustrated in Figure 1.

B contains an M-alternating circuit through $(x, y) \in M$ if and only if D contains a directed circuit through x. We may find a directed circuit in D in $O(e)$ time. (The algorithm for finding strongly connected components may be employed [1].)

The procedure *NEW_SOLUTIONS*(G, M, C, L) below accepts a bipartite graph G which is a subgraph of B, a perfect matching M of G, and an M-alternating circuit C. It finds all the additional perfect matchings of G A perfect matching of B is obtained by adding the set of edges L to a perfect matching of G. Note that *NEW_SOLUTIONS* is invoked only when M is not unique.

Method of Operation. Let $(x, y) \in M$ be an edge of the M-alternating circuit C. The

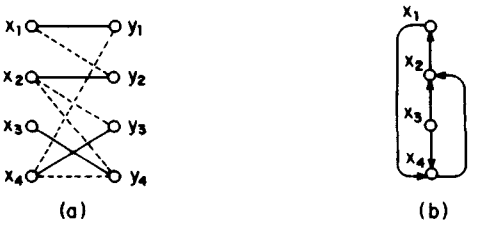

FIG 1

perfect matchings of G fall into two disjoint categories:

(a) Matchings which do not contain (x, y): The perfect matching $M_e = M \oplus C$ does not contain the edge (x, y). Let $G_e = G - \{(x, y)\}$. There exist additional matchings which do not contain (x, y) if and only if there exists an M_e-alternating circuit C_e in G_e. These matchings are found by invoking $NEW\_SOLUTIONS(G_e, M_e, C_e, L)$ recursively.

(b) Matchings which contain (x, y). Let $M_v = M - \{(x, y)\}$. Then there exist additional matchings which contain the edge (x, y) if there exists an M_v-alternating circuit C_v in $G_v = G - \{x, y\}$. (M_v is a perfect matching in G_v.) These matchings are found by invoking $NEW\_SOLUTIONS(G_v, M_v, C_v, L \cup \{(x, y)\})$ recursively.

```
    procedure NEW_SOLUTIONS(G, M, C, L),
    begin comment Me, Ge, Ce, Mv, Gv, Cv are local variables,
1     Let (x, y) ∈ M ∩ C,
      Me := M ⊕ C,
2     Ge = G − {(x, y)}, (delete the edge (x, y) from G). Find an alternating circuit Ce in Ge with respect to Me, if
      none exists then Ce := nil,
3     Mv := M − {(x, y)};
      Gv = G − {x, y},   } (delete the vertices x, y from G and M)
      Find an alternating circuit Cv in Gv with respect to Mv; if none exists then Cv = nil,
4     Output Me ∪ L,
5     if Ce ≠ nil then call NEW_SOLUTIONS(Ge, Me, Ce, L),
6     if Cv ≠ nil then call NEW_SOLUTIONS(Gv, Mv, Cv, L ∪ {(x, y)})
    end
```

LEMMA 2. *The procedure NEW_SOLUTIONS finds a new matching in $O(e)$ time.*

PROOF. On entering *NEW_SOLUTIONS*, lines 1–3 are executed requiring at most $O(e)$ time. A new matching is output in line 4. Consequently, a new matching is output $O(e)$ time after entering *NEW_SOLUTIONS*. If $C_e \neq$ **nil** or $C_v \neq$ **nil** then *NEW_SOLUTIONS* is called recursively and a new matching is found in $O(e)$ time. If both C_e and C_v are equal to **nil** then a return from the recursion occurs without finding a new matching. Checking $C_e \neq$ **nil** and $C_v \neq$ **nil** takes constant time. Thus the total time to exit the recursion is bounded by a constant times the depth of the recursion, which is at most e. Q.E.D

The procedure *ALL_SOLUTIONS* finds all perfect matchings using *NEW_SOLUTIONS*.

```
procedure ALL_SOLUTIONS(B),
begin Find a perfect matching M,
  if none exists then return,
  Output M;
  Find an alternating circuit C in B with respect to M,
  if none exists then return,
  call NEW_SOLUTIONS(B, M, C, ∅)
end
```

Note that the algorithm halts at most $O(e)$ time after the last matching is found

We may summarize:

THEOREM 6. *(a) The procedure ALL_SOLUTIONS finds all perfect matchings; (b) the time delay to find a new matching is O(e); (c) if a bipartite graph contains m perfect matchings they are found by ALL_SOLUTIONS in O(e(n^{1/2} + m)) time*

Since the depth of the recursion is at most e and each circuit may have at most n edges, the space requirements in the above formulation are bounded by $O(ne)$, provided G_e and G_v are produced by deleting edges and vertices and restoring them later. By recomputing the alternating circuits instead of storing them then the algorithm may be implemented within $O(e)$ space. (The execution time may be doubled.)

The procedure *ALL_SOLUTIONS* may be used to compute the *permanent* of a matrix. Let $A = (a_{ij})$ be a real matrix; then its permanent is defined by $perm(A) = \sum_\pi \prod_{i=1}^n a_{i,\pi(i)}$. The summation is over all permutations π on n letters. Let $B = (X \cup Y, E)$ be a bipartite graph where $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_n\}$, and $(x_i, y_j) \in E$ if and only if $a_{ij} \neq 0$. $S_\pi = \prod_{i=1}^n a_{i,\pi(i)}$ is nonzero if and only if $M_\pi = \{(x_i, y_{\pi(i)}) | i = 1, \ldots, n\} \cap E$ is a perfect matching. The value of the permanent is equal to the sum of S_π over all permutations which correspond to perfect matchings. Thus, we may use *ALL_SOLUTIONS* to find the value of the permanent in $O(e(n^{1/2} + m))$ time, and $(n - 1)m$ multiplications (m is the number of perfect matchings). The standard way to compute the permanent requires $O(2^n)$ time [18].

5. *Finding All Maximum Matchings*

In the previous section the problem of finding all perfect matchings in a bipartite graph was considered. Now we turn to generalize the techniques and find all maximum matchings instead of just perfect matchings. As before an initial maximum matching may be found in $O(n^{1/2}e)$.

If the matching M is maximum but not perfect then there exist *exposed vertices*—vertices which are not incident with any edge of M. Let z be an exposed vertex and (z, y) an edge; then y is matched since otherwise M is not maximum. Let $(x, y) \in M$. Then other maximum matchings can be found from M not only by alternating circuits but also by replacing (x, y) by (y, z). A path (x, y, z) is an *M-transposition* if $(x, y) \in M$ and z is an exposed vertex. The following lemma is a generalization of Lemma 1 and is proven similarly.

LEMMA 3. *A maximum matching M is not unique if and only if there exists either an M-alternating circuit or an M-transposition.*

M-alternating circuits are found as before.

As in the case of perfect matchings, B contains an M-alternating circuit through $(x, y) \in M$ if and only if D contains a directed circuit through x.

To find an M-transposition choose any nonisolated exposed vertex z, a vertex y adjacent to z, and that vertex x to which y is matched. (Such a vertex x always exists since M is a maximum matching.) Searching for an M-transposition takes at most $O(n)$ time.

Now we outline the method of operation of an algorithm for finding all maximum matchings of a bipartite graph B. First find an initial maximum matching. Then search for an M-transposition (x, y, z). If the search succeeds then define $M_e = M \cup \{(y, z)\} - \{(x, y)\}$, $B_e = B - \{(x, y)\}$ and proceed recursively to look for new maximum matchings in B_e. Also define $M_v = M - \{(x, y)\}$, $B_v = B - \{x, y\}$ and check whether M_v is a unique maximum matching in B_v. Appending (x, y) to these matchings yields maximum matchings of B. If no M-transposition exists, search for an M-alternating circuit and proceed as in *NEW_SOLUTIONS*

We have omitted the bookkeeping details but it is quite obvious that the behavior of the generalized algorithm is similar to that of *ALL_SOLUTIONS.*

We summarize:

THEOREM 7. *All maximum matchings in a bipartite graph may be found such that the time delay to find a new matching is at most O(e). The entire algorithm requires O(e) space*

6. *Conclusions*

For the case of a single restriction an $O(ne)$ solution has been presented. However, it is not known whether there exists a polynomial solution even for two restrictions. One of the referees suggested a variant to Theorem 1, in which the number of restrictions is $O(n)$, the maximum degree is two, and the restriction on the number of edges that may appear in a matching is either greater than or equal or less than or equal to one. His proof is similar to ours.

If M is a matching which satisfies most of the restrictions, then in many cases there exists a matching M' such that M and M' have many edges in common and M' satisfies all the requirements. Therefore, it might be worthwhile to find all matchings such that successive matchings are close to one another. This may be done by using minimum length alternating circuits [14].

It is unknown whether the maximum matchings generator for bipartite graphs can be extended for arbitrary graphs with the time delay still bounded by $O(e)$ The following techniques of the bipartite case are applicable also for arbitrary graphs:

(i) An initial matching may be found in $O(n^{2.5})$ time [7].
(ii) An M-transposition may be found in $O(n)$ time
(iii) The procedures *ALL_SOLUTIONS* and *NEW_SOLUTIONS* are valid.

However, the missing link in the process is finding an M-alternating circuit in $O(e)$ time. Such a circuit can be found in $O(ne)$ time: Choose a matched edge (x, y) and delete it. Then try to find an augmenting path from x to y. If a path P is found, $P_v(x, y)$ is an alternating cycle; otherwise repeat the procedure for the other matched edges. Finding an augmenting path is $O(e)$ [10], using breath-first search, or [15].

ACKNOWLEDGMENT. The authors wish to thank Dr. O. Kariv for helpful discussions, and the referees for many useful suggestions

REFERENCES

(Note Reference [2] is not cited in the text)

1 AHO, A V, HOPCROFT, J E, AND ULLMAN, J D *The Design and Analysis of Computer Algorithms* Addison-Wesley, Reading, Mass, 1974
2 BERGE, C *Graphs and Hypergraphs* North-Holland, Amsterdam, 1973.
3 DIJKSTRA, E W A note on two problems in connexion with graphs *Numer Math 1* (1959), 269-271
4 EDMONDS, J, AND KARP, R M Theoretical improvements in algorithmic efficiency for network flow problems. *J ACM 19*, 2 (April 1972), 248-264
5 EVEN, S *Algorithmic Combinatorics* MacMillan, New York, 1973
6 EVEN, S, ITAI, A, AND SHAMIR, A On the complexity of timetable and multi-commodity flow *SIAM J Comptng. 5* (1976), 691-703
7 EVEN, S, AND KARIV, O An $O(n^{2.5})$ algorithm for maximum matchings in general graphs Proc 16th Symp on Foundations of Comptng, 1975, pp. 382-399
8 FORD JR, C R, AND FULKERSON, D R *Flows in Networks* Princeton U. Press, Princeton, N J, 1962
9 FREUDER, E C Synthesizing constraint expressions A I Memo 370, Artif Intell Lab, M I T, Cambridge, Mass, July 1976
10 GABOW, H N An efficient implementation of Edmonds' algorithm for maximum matching on graphs *J ACM 23*, 2 (April 1976), 221-234
11 GABOW, H N, AND LAWLER, E An efficient implementation of Edmonds' algorithm for maximum matching on graphs Rep CV-CS-075-75, Dept Comptr Sci, U of Colorado, Boulder, Colo, 1975
12 GAL, S, AND BREITBART, Y A method for obtaining all the solutions of a perfect matching problem TR-16, IBM Israel Scientific Ctr, Haifa, Israel, 1974
13 HOPCROFT, J E, AND KARP, R M An $n^{5/2}$ algorithm for maximum matching in bipartite graphs *SIAM J Comptng 2* (1973), 225-231
14 ITAI, A, AND RODEH, M Finding a minimum circuit in a graph Proc 1977 ACM Symp Theory of Comptng, Boulder, Colo, May 1977, pp 1-10
15 KAMEDA, T, AND MUNRO, I A $O(|V| |E|)$ algorithm for maximum matchings of graphs *Computing 12* (1974), 91-98
16 LANG, L L, AND STARKEY, J D An $O(e \log n)$ shortest path algorithm for sparse graphs (abstract) In Traub, J F. (Ed), Proc Symp Algorithms and Complexity, Carnegie-Mellon U, April 1976, p 476

17 MACKWORTH, A K Consistency in networks of relations TR 75-3, Comptr Sci Dept, U. of British Columbia, Vancouver, B C, Canada, 1975
18 RYSER, J R *Combinatorial Mathematics* Math Assoc Amer, dist John Wiley, New York, 1963
19 TANIMOTO, S L Analysis of biomedical images using maximal matching Proc 1976 IEEE Conf Decision and Control Adaptive Processes, Clearwater Beach, Fla, Dec 1976, pp 171–176
20 WAGNER, R A A shortest path algorithm for edge-sparse graphs *J ACM 23*, 1 (Jan 1976), 50–57
21 WALTZ, D Understanding line drawings of scenes with shadows In *The Psychology of Computer Vision*, P H Winston, Ed, McGraw-Hill, New York, 1975, pp 19–91

RECEIVED JULY 1977, REVISED DECEMBER 1977