SOME MATCHING PROBLEMS

Alon Itai
Technion, Israel Institute of Technology, Haifa, Israel
Michael Rodeh
IBM Israel Scientific Center, Haifa, Israel

ABSTRACT

In certain applications it is required to find in a bipartite graph a perfect matching which satisfies some additional properties. For one such type of restrictions the problem is proven to be NP-complete. If for a given subset of edges no more than r edges may be included in the matching then an $O(ne)$ algorithm is suggested.

Finally, an efficient algorithm to find 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.

1. INTRODUCTION

This paper is motivated by an attempt to find practical solutions to the school-scheduling problem. This problem is known to be NP-complete [EIS]. (A discussion on NP-completeness appears in [AHU].) A solution can be considered as a matching between classes and teachers which 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 [E] for the properties of graphs and matchings.) This problem is shown to be NP-complete. If no restrictions are present, then a maximum matching may be found in $O(n^{1/2}e)$ time [HK] (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 [FF]. In our case the algorithm due to Edmonds and Karp [EK] may be simplified and implemented more efficiently using the shortest path finding techniques of Wagner [W].

The restricted matching problem 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 [GB] where $O(n^3)$ time may elapse without getting a solution. Tanimoto has developed a method for enumeration of maximum matchings for a general graph in order to analyze some biomedical images [T]. His 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 [R] for the properties of the permanent.

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 set $x \in X$ 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 [HK] 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:

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

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 $\psi = C_1 \ldots C_p$ 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:

$$X = \{C_1,\ldots,C_p\} ;$$

$$Y = \{x_1,\ldots,x_q,\bar{x}_1,\ldots,\bar{x}_q\} \times \{1,\ldots,p\};$$

$$E = \{(C_j,(x_i,j)) \mid x_i \text{ appears in } C_j\} \cup$$

$$\{(C_j,(\bar{x}_i,j)) \mid \bar{x}_i \text{ appears in } C_j\} .$$

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 $E' = \{(v_i, y) \mid v_i \in X', y \in Y\}$. 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

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

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 [FF]. 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) $A = \{(s,x) \mid x \in X\} \cup \{(y,t) \mid y \in Y\} \cup \{(x,y) \mid (x,y) \in E, x \in X, y \in Y\}$;

(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 [FF] and [EK].

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) $G^f = (V, A^f)$;

$$A^f = \{(u,v) \mid (u,v) \in A, f(u,v) = 0\} \cup \{(u,v) \mid (v,u) \in A, f(v,u) = 1\};$$

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

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

Theorem 3: ([FF], 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 wieghts. Since it is more efficient to find shortest paths in a graph with nonnegative weights, Edmonds and Karp [EK] 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 inaccessable from s
                                  then σ^k(u):=∞
                    end
          end
end
```

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

Theorem 4: ([EK]). For each k and u, $\pi^k(u)$ is the weight of a shortest path from s to t 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 $0 \le \sigma^k(u) \le n$. Q.E.D.

Following Dijkstra's algorithm [D], 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 [LS]. We take advantage of the fact that in our case the distances are integers between 0 and n. Following Wagner [W] we keep the vertices of $\bar{S}$ in a vector of buckets (the i-th 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. 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.

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 [HK]. 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 non-trivial 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 $E' = \{(u,v) \mid u,v \in X, u \neq v, \exists w \in Y: (u,w) \in M$ 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.

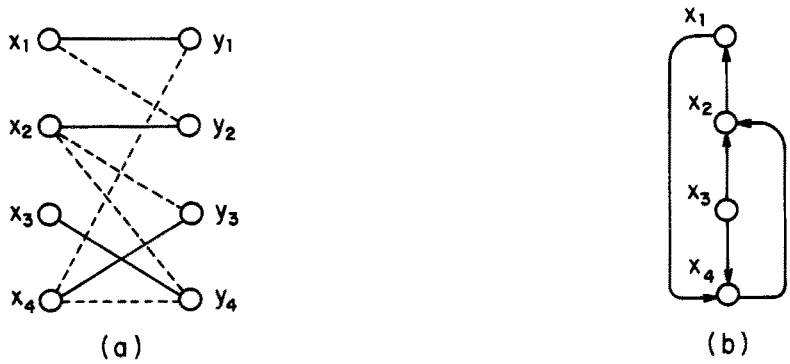

Fig. 1

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 [AHU].)

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. Every perfect matching of B is obtained by adding the set of edges L to the perfect matchings 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 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 exists 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 $O(ne)$. 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 $A = (a_{ij})_n$ be a real matrix, then its permanent is defined by $\mathrm{perm}(A) = \sum_{\pi} \prod_{i=1}^{n} a_{i,\pi(i)}$. 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 $a_{ij} = 0$. $S_\pi = \prod_{i=1}^{n} a_{i,\pi(i)}$ is nonzero if and only if $M_\pi = \{(x_i, y_{\pi(i)}) \mid i = 1, \ldots, n\} \subset E$ 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.)

5. CONCLUSIONS

Certain aspects of the school-scheduling problem may be formalized in terms of perfect matchings with restrictions. However, in practice the number of teachers is not equal to the number of classes. In this case maximum matchings should be found. An algorithm for finding maximum matchings in a general graph will appear in a forthcoming paper by the authors and S.L. Tanimoto.

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.

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 [IR].

REFERENCES

[AHU] A.V.Aho, J.E.Hopcroft and J.D. Ullman, "The Design and Analysis of Computer Algorithms", Addison-Wesley (1974).

[B] C. Berge, "Graphs and Hypergraphs", North-Holland (1973).

[D] E.W. Dijkstra, "A Note on Two Problems in Connexion with Graphs", Numerische Mathematik 1 (1959), 269-271.

[EK] J. Edmonds and R.M.Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems", J. ACM, 19 (1972), 248-264.

[E] S. Even, "Algorithmic Combinatorics", MacMillan (1973).

[EIS] S. Even, A. Itai and A. Shamir, "On the Complexity of Timetable and Multi-Commodity Flow", SIAM J. on Computing, 5 (1976), 691-703.

[FF] C.R. Ford Jr. and D.R. Fulkerson, "Flows in Networks", Princeton University Press (1962).

[GB] S. Gal and Y. Breitbart, "A Method for Obtaining all the Solutions of a Perfect Matching Problem", IBM Israel Scientific Center TR-16 (1974).

[HK] J.E. Hopcroft and R.M. Karp, "An $n^{5/2}$ Algorithm for Maximum Matching in Bipartite Graphs", SIAM J. on Computing, 2 (1973), 225-231.

[IR] A. Itai and M. Rodeh, "Finding a Minimum Circuit in a Graph", Proc. of the 1977 ACM Symp. on Theory of Computing, Boulder, Colorado (May 1977).

[LS] L.L. Lang and J.D. Starkey, "An $O(e\log n)$ Shortest Path Algorithm for Sparse Graphs", Proc. on the Symp. on Algorithms and Complexity, Carnegie-Mellon University (April 1976).

REFERENCES (cont'd)

[R] J.R. Ryser, "Combinatorial Mathematics", The Mathematical Association of America, distributed by John Wiley & Sons (1963).

[T] S.L. Tanimoto, "Analysis of Biomedical Images Using Maximal Matching", Proc. of the 1976 IEEE Conf. on Decision and Control Adaptive Processes, Clearwater Beach, Florida (Dec. 1976).

[W] R.A. Wagner, "A Shortest Path Algorithm for Edge-Sparse Graphs", J. ACM, 23 (1976), 50-57.