11.3 MATCHINGS AND ASSIGNMENTS

Douglas R. Shier, Clemson University

Introduction

In an undirected graph, the maximum matching problem requires finding a set of nonadjacent edges having the largest total size or largest total weight. This graph optimization problem arises in a number of applications, often involving the optimal pairing of a set of objects.

11.3.1 Matchings

Matchings are defined on undirected graphs, in which the edges can be weighted. Matchings are useful in a wide variety of applications, such as vehicle and crew scheduling, sensor location, snowplowing streets, scheduling on parallel machines, among others.

DEFINITIONS

D1: Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E. Each edge $e \in E$ has an associated ***weight*** w_e.

D2: A ***matching*** in $G = (V, E)$ is a set $M \subseteq E$ of pairwise nonadjacent edges.

D3: A ***vertex cover*** in G is a set C of vertices such that every edge in G is incident on at least one vertex in C.

D4: A ***perfect matching*** in $G = (V, E)$ is a matching M in which each vertex of V is incident on exactly one edge of M.

TERMINOLOGY: A perfect matching of G is also called a ***1-factor*** of G; see §5.3.

D5: The ***size*** (***cardinality***) of a matching M is the number of edges in M, written $|M|$. The ***weight*** of a matching M is $wt(M) = \sum_{e \in M} w_e$.

D6: A ***maximum-size matching*** of G is a matching M having the largest size $|M|$.

D7: A ***maximum-weight matching*** of G is a matching M having the largest weight $wt(M)$.

D8: Relative to a matching M in $G = (V, E)$, edges $e \in M$ are ***matched*** edges, while edges $e \in E - M$ are **free** edges. Vertex v is ***matched*** if it is incident on a matched edge; otherwise vertex v is **free** (or ***unmatched***).

D9: Every matched vertex v has a ***mate***, the other endpoint of the matched edge incident on v.

D10: With respect to a matching M, the ***weight*** $wt(P)$ of path P is the sum of the weights of the free edges in P minus the sum of the weights of the matched edges in P.

D11: An ***alternating*** path has edges that are alternately free and matched. An ***augmenting*** path is an alternating path that starts at a free vertex and ends at another free vertex.

NOTATION: Throughout this section, edges are represented as ordered pairs of vertices, and when discussing matchings, paths are represented as edge sets.

EXAMPLES

E1: Figure 11.3.1 shows a graph G together with the matching $M_1 = \{(2,3),(4,5)\}$ of size 2; the matched edges are highlighted. The mate of vertex 2 is vertex 3, and the mate of vertex 5 is vertex 4. Relative to matching M_1, vertices 1 and 6 are free vertices, and an augmenting path P from 1 to 6 is given by $P = \{(1,2),(2,3),(3,4),(4,5),(5,6)\}$. The matching M_1 is not a maximum-size matching; the matching $M_2 = \{(1,2),(3,4),(5,6)\}$ of size 3 is a maximum-size matching, in fact a perfect matching.

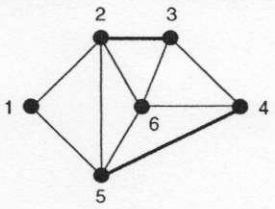

Figure 11.3.1 A matching in a graph.

E2: Figure 11.3.2 shows a graph G with 6 vertices. The matching $M = \{(2,4),(3,5)\}$ displayed is a maximum-size matching, of size 2. This graph G does not have a perfect matching.

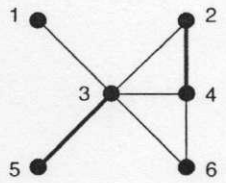

Figure 11.3.2 A maximum-size matching that is not perfect.

E3: In the weighted graph G of Figure 11.3.3 below, the weight w_e is shown next to each edge e. The weight of matching $M = \{(1,2),(3,5)\}$ is $wt(M) = 7$. Relative to this matching, the path $P = \{(1,2),(2,5),(3,5),(3,6)\}$ is an alternating path with $wt(P) = 7 + 1 - 2 - 5 = 1$. The path $\{(1,4),(1,2),(2,3),(3,5),(5,6)\}$ is an augmenting path, joining the free vertices 4 and 6.

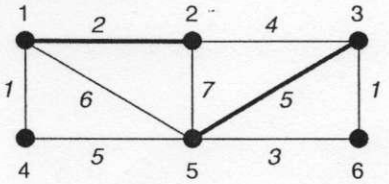

Figure 11.3.3 A matching in a weighted graph.

Some Fundamental Results

FACTS

F1: If M is a matching of $G = (V, E)$, then the number of matched vertices is $2|M|$ and the number of free vertices is $|V| - 2|M|$.

F2: If M is any matching in G, then $|M| \leq \left\lfloor \frac{|V|}{2} \right\rfloor$.

F3: (Weak Duality) The size of any vertex cover of G is an upper bound on the size of any matching in G.

F4: Every augmenting path has an odd number of edges.

F5: If M is a matching and P is an augmenting path with respect to M, then the symmetric difference $M \Delta P$ is a matching of size $|M| + 1$.

NOTE: The symmetric difference $M \Delta P$ is taken with respect to the *edge sets* defining M and P.

F6: (Augmenting Path Theorem) M is a maximum-size matching if and only if there is no augmenting path with respect to M. (See [Pe1891], [Be57], [NoRa59].)

F7: If M is a matching and P is an augmenting path with respect to M, then $wt(M \Delta P) = wt(M) + wt(P)$.

F8: Suppose M is a matching having maximum weight among all matchings of a fixed size k. If P is an augmenting path of maximum weight with respect to M, then $M \Delta P$ is a maximum-weight matching among all matchings of size $k + 1$.

F9: Let M_i be a maximum-weight matching among all matchings of a fixed size i, $i = 1, 2 \ldots, k$, and let P_i be a maximum-weight augmenting path with respect to M_i. Then $wt(P_1) \geq wt(P_2) \geq \cdots \geq wt(P_k)$.

F10: An immediate consequence of Facts 8 and 9 is a weighted-matching analogue to the Augmenting Path Theorem: A matching M is of maximum weight if and only if the weight of every augmenting path relative to M is nonpositive.

EXAMPLES

E4: In Figure 11.3.1, the path $P = \{(1,2),(2,3),(3,4),(4,5),(5,6)\}$ is augmenting with respect to the matching $M_1 = \{(2,3),(4,5)\}$. As guaranteed by Fact 4, path P has an odd number of edges. The new matching $M_2 = M_1 \Delta P = \{(1,2),(3,4),(5,6)\}$ has size one greater than M_1, and is a maximum-size matching. There are other maximum-size matchings, such as $\{(1,2),(3,6),(4,5)\}$ and $\{(1,5),(2,6),(3,4)\}$.

E5: In Figure 11.3.2, the set $S = \{3,4\}$ is a vertex cover of G. Thus, by Fact 3, the size of any matching M satisfies $|M| \leq 2 = |S|$. On the other hand, $S = \{2,3,4,5\}$ is a (minimum cardinality) vertex cover of the graph in Figure 11.3.1, yet a maximum-size matching M for this graph satisfies $|M| = 3 < |S|$.

E6: Figure 11.3.4(a) below shows a matching M_1 of size 1, with $wt(M_1) = 7$. Since edge $(2,5)$ has maximum weight among all edges, M_1 is a maximum-weight matching of size 1. Relative to M_1, the augmenting path $P_1 = \{(1,5),(2,5),(2,3)\}$ has weight $wt(P_1) = 6 + 4 - 7 = 3$, whereas the augmenting path $P_2 = \{(3,6)\}$ has weight 1. It can be verified that P_1 is a maximum-weight augmenting path relative to M_1. Illustrating

Fact 8, $M_2 = M_1 \Delta P_1 = \{(1,5),(2,3)\}$ is a maximum-weight matching of size 2, with $wt(M_2) = wt(M_1) + wt(P_1) = 10$ (see Figure 11.3.4(b)). Relative to M_2 there are several augmenting paths between the free vertices 4 and 6:

$$Q_1 = \{(1,4),(1,5),(5,6)\}, \quad wt(Q_1) = 1+3-6 = -2,$$
$$Q_2 = \{(1,4),(1,5),(2,5),(2,3),(3,6)\}, \quad wt(Q_2) = 1+7+1-6-4 = -1,$$
$$Q_3 = \{(4,5),(1,5),(1,2),(2,3),(3,6)\}, \quad wt(Q_3) = 5+2+1-6-4 = -2.$$

Path Q_2 is a maximum-weight augmenting path and so (by Fact 8) $M_3 = M_2 \Delta Q_2 = \{(1,4),(2,5),(3,6)\}$ is a maximum-weight matching of size 3, with $wt(M_3) = 9$. All augmenting paths relative to M_2 have negative weight, and so (by Fact 10) M_2 is a maximum-weight matching in G.

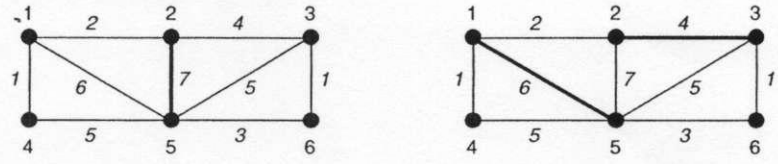

Figure 11.3.4 Maximum-weight matchings of sizes 1 and 2.

REMARKS

R1: Fact 6 was obtained independently by C. Berge [Be57] and by R. Z. Norman and M. O. Rabin [NoRa59]. This result was also recognized in an 1891 paper of J. Petersen [Pe1891].

R2: An historical perspective on the theory of matchings is provided in [Pl92].

R3: Plummer [Pl93] describes a number of variations on the standard matching problem, together with their computational complexity.

11.3.2 Matchings in Bipartite Graphs

Bipartite graphs arise in a number of applications (such as in assigning personnel to jobs or tracking objects over time). See the surveys [AhMaOr93], [AhMaOrRe95], and [Ge95] as well as the text [LoPl86] for additional applications. This section describes properties and algorithms for maximum-size and maximum-weight matchings in bipartite graphs.

DEFINITIONS

D12: Let $G = (X \cup Y, E)$ be a bipartite graph with n vertices, m edges, and edge weights w_e.

D13: If $S \subseteq X$ then $\Gamma(S) = \{y \in Y \mid (x,y) \in E \text{ for some } x \in S\}$ is the set of vertices in Y adjacent to some vertex of S.

D14: A ***complete*** (or ***X-saturating***) ***matching*** of $G = (X \cup Y, E)$ is a matching M in which each vertex of X is incident on an edge of M. Such a matching is also called an ***assignment*** from X to Y.

APPLICATIONS

A1: A drug company is testing n antibiotics on n volunteer patients in a hospital. Some patients have known allergic reactions to certain of these antibiotics. To determine if there is a feasible assignment of the n different antibiotics to n different patients, construct the bipartite graph $G = (X \cup Y, E)$, where X is the set of antibiotics and Y is the set of patients. An edge $(i, j) \in E$ exists when patient j is *not* allergic to antibiotic i. A complete matching of G is then sought.

A2: There are n applicants to be assigned to n jobs, with each job being filled with exactly one applicant. The weight w_{ij} measures the suitability (or productivity) of applicant i for job j. Finding a valid assignment (matching) achieving the best overall suitability is a weighted matching problem on the bipartite graph $G = (X \cup Y, E)$, where X is the set of applicants and Y is the set of jobs.

A3: The movements of n objects (such as submarines or missiles) are to be followed over time. The locations of the group of objects are known at two distinct times, though without identification of the individual objects. Suppose $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$ represent the spatial coordinates of the objects detected at times t and $t + \Delta t$. If Δt is sufficiently small, then the Euclidean distance between a given object's position at these two times should be relatively small. To aid in identifying the objects (as well as their velocities and directions of travel), a pairing between set X and set Y is desired that minimizes the overall sum of Euclidean distances. This can be formulated as a maximum-weight matching problem on the complete bipartite graph $G = (X \cup Y, E)$, where edge (i, j) indicates pairing position x_i with position y_j. The weight of this edge is the negative of the Euclidean distance between x_i and y_j. A maximum-weight matching of size n in G then provides an optimal (minimum distance) pairing of observations at the two times t and $t + \Delta t$.

FACTS

F11: (König's Theorem) For a bipartite graph G, the maximum size of a matching in G is the minimum cardinality of a vertex cover in G. Thus for bipartite graphs the general inequality stated in Fact 3 can always be satisfied as an equality. (See [Bo90].)

F12: (Hall's Theorem) $G = (X \cup Y, E)$ has a complete matching if and only if $|\Gamma(S)| \geq |S|$ holds for every $S \subseteq X$. In words, a complete matching exists precisely when every set of vertices in X is adjacent to at least an equal number of vertices in Y. (See [Bo90, Gr99].)

F13: Suppose there exists some k such that $deg(x) \geq k \geq deg(y)$ holds in $G = (X \cup Y, E)$ for all $x \in X$ and $y \in Y$. Then G has a complete matching. (See [Gr99].)

EXAMPLES

E7: In the bipartite graph of Figure 11.3.5 below, $S = \{1, 3, b, d\}$ is a vertex cover of minimum cardinality, and $M = \{(1, a), (3, c), (4, b), (5, d)\}$ is a maximum-size matching. As guaranteed by König's Theorem, $|M| = |S|$. Also, by choosing $A = \{2, 4, 5\}$ we have $\Gamma(A) = \{b, d\}$. Since $|\Gamma(A)| < |A|$ holds, Hall's Theorem shows that there is no complete matching with respect to the set $X = \{1, 2, 3, 4, 5\}$. In fact, the maximum matching M above has size $4 < 5$.

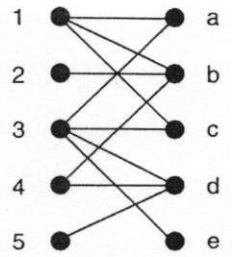

Figure 11.3.5 Covers and matchings in a bipartite graph.

E8: In the chessboard of Figure 11.3.6, we are to place non-taking rooks at certain allowable positions, those marked with an X. For example, we can place rooks at the *independent* positions $(1,3),(2,4),(4,1)$: no two selected positions are in the same row or column. It turns out that three is the maximum number of rooks that can be so placed with regard to the allowable positions X. Also, notice that row 2, row 4, and column 3 are three *lines* in the chessboard containing all X entries; in fact, no fewer number of lines suffice. Here the maximum number of non-taking rooks among the X entries equals the minimum number of lines containing all the X entries. This is a manifestation of König's Theorem, obtained by constructing the bipartite graph $G = (X \cup Y, E)$ where X contains the rows $\{1,2,3,4\}$ and Y contains the columns $\{1,2,3,4,5\}$; edge $(i,j) \in E$ indicates an X in row i and column j. In this context, independent positions correspond to a matching and covering lines correspond to a vertex cover in G.

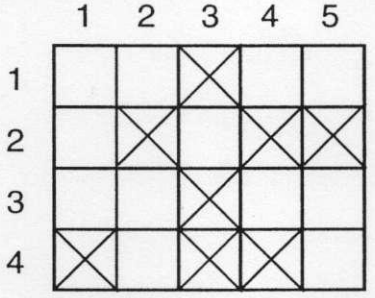

Figure 11.3.6 A chessboard with allowable X entries.

REMARKS

R4: König's Theorem and Hall's Theorem can be derived from the Max-Flow Min-Cut Theorem of §11.1.

R5: Maximum-size matching problems in bipartite graphs can be formulated as maximum flow problems in unit capacity networks and solved using maximum flow algorithms (§11.1).

R6: Maximum-weight matching problems in bipartite graphs can be formulated as minimum cost flow problems in two-terminal flow networks and solved using minimum cost flow algorithms (§11.2).

Bipartite Maximum-Size Matching Algorithm

Algorithm 11.3.1, based on Fact 6, produces a maximum-size matching of the bipartite graph $G = (X \cup Y, E)$. Each iteration involves a modified breadth-first search of G,

starting with all free vertices in set X. The vertices of G are structured into levels that alternate between free and matched edges. Algorithm algnum can be implemented to run in $O(nm)$ time (see [PaSt82]).

Algorithm 11.3.1: Bipartite Maximum-Size Matching

Input: Bipartite graph $G = (X \cup Y, E)$.
Output: Maximum-size matching M.

```
M := ∅
DONE := FALSE
While NOT DONE
    Let FREE consist of all the free vertices of G.
    S_X := X ∩ FREE
    SEEN := ∅
    STILL_LOOKING := TRUE
    While STILL_LOOKING {for an augmenting path}
        S_Y := {y | y ∉ SEEN and (x, y) ∈ E, x ∈ S_X}
        If S_Y ∩ FREE ≠ ∅ {an augmenting path exists}
            Construct an augmenting path P to y*.  [*]
            M := MΔP
            STILL_LOOKING := FALSE
        Else {continue looking for an augmenting path}
            SEEN := SEEN ∪ S_Y
            S_X := {x | (y, x) ∈ M, y ∈ S_Y}
            If S_X = ∅
                STILL_LOOKING := FALSE
                DONE := TRUE
```

REMARK

R7: The augmenting path at step [*] is constructed in reverse, starting at the free Y-vertex y^*. Choose a vertex $x \in S_X$ (adjacent to y^*) by which y^* was defined to be an element of S_Y. Then choose the vertex $y \in S_Y$ that is matched to x in M. Vertices from X and Y are alternately chosen in this way until an x is chosen from the initial S_X, which means that it is a free vertex.

EXAMPLE

E9: Algorithm 11.3.1 can be used to find a maximum-size matching in the bipartite graph of Figure 11.3.7 below. We begin with the matching $M = \{(1,a),(2,b)\}$ of size 2, shown in Figure 11.3.7(a). At the next iteration, $S_X = \{3,4\}$ and $S_Y = \{a,b\}$. Since both vertices of S_Y are matched, the algorithm continues with $S_X = \{1,2\}$ and $S_Y = \{c\}$. Since $c \in S_Y$ is free, with augmenting path $P = \{(3,a),(a,1),(1,c)\}$, the new matching produced is $M = \{(1,c),(2,b),(3,a)\}$; see Figure 11.3.7(b). The next iteration produces $S_X = \{4\}$, $S_Y = \{b\}$; $S_X = \{2\}$, $S_Y = \{a,c\}$; finally $S_X = \{1,3\}$, $S_Y = \emptyset$, $S_X = \emptyset$. No further augmenting paths are found, and Algorithm 11.3.1 terminates with the maximum-size matching $M = \{(1,c),(2,b),(3,a)\}$.

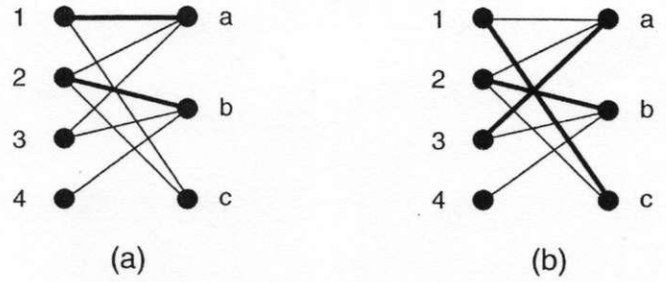

Figure 11.3.7 Maximum-size matching in a bipartite graph.

Bipartite Maximum-Weight Matching Algorithm

Algorithm 11.3.2, based on Facts 8, 9, and 10, produces a maximum-weight matching of $G = (X \cup Y, E)$. Each iteration finds a maximum-weight augmenting path relative to the current matching M. The algorithm terminates when the path has nonpositive weight. A straightforward implementation of Algorithm 11.3.2 runs in $O(n^2m)$ time.

NOTATION: The tentative largest weight of an alternating path from a free vertex in X to vertex j is maintained using the label $d(j)$.

Algorithm 11.3.2: Bipartite Maximum-Weight Matching

Input: Bipartite graph $G = (X \cup Y, E)$.
Output: Maximum-weight matching M.

$M := \emptyset$
$DONE :=$ FALSE
While NOT $DONE$
  Let S_X consist of all the free vertices of X.
  Let $d(j) := 0$ for $j \in S_X$ and $d(j) := -\infty$ otherwise.
  While $S_X \neq \emptyset$
    $S_Y := \emptyset$
    For each edge $(x, y) \in E - M$ with $x \in S_X$
      If $d(x) + w_{xy} > d(y)$
        $d(y) := d(x) + w_{xy}$
        $S_Y := S_Y \cup \{y\}$
    $S_X := \emptyset$
    For each edge $(y, x) \in M$ with $y \in S_Y$
      If $d(y) - w_{yx} > d(x)$
        $d(x) := d(y) - w_{yx}$
        $S_X := S_X \cup \{x\}$
  Let y be a free vertex with maximum label $d(y)$
   and let P be the associated path.
  If $d(y) > 0$
    $M := M \Delta P$
  Else
    $DONE :=$ TRUE

EXAMPLE

E10: Algorithm 11.3.2 can be used to find a maximum-weight matching in the bipartite graph of Figure 11.3.8. If we begin with the empty matching, then the first iteration yields the augmenting path $P_1 = \{(3,a)\}$, with $wt(P_1) = 6$, and the maximum-weight matching (of size 1) $M = \{(3,a)\}$, with $wt(M) = 6$; see Figure 11.3.8(a). The next iteration starts with $S_X = \{1,2\}$. The labels on vertices a, b, c are then updated to $d(a) = 4$, $d(b) = 4$, $d(c) = 5$, so $S_Y = \{a, b, c\}$. Using the matched edge $(a, 3)$, vertex 3 has its label updated to $d(3) = -2$ and $S_X = \{3\}$. No further updates occur, and the free vertex c with maximum label $d(c) = 5$ is selected. This label corresponds to the augmenting path $P_2 = \{(2,c)\}$, with $wt(P_2) = 5$. The new matching is $M = \{(2,c),(3,a)\}$, with $wt(M) = 11$; see Figure 11.3.8(b). At the next iteration, $S_X = \{1\}$ and vertices a, b receive updated labels $d(a) = 4$, $d(b) = 1$. Subsequent updates produce $d(3) = -2$, $d(c) = 3$, $d(2) = -2$, $d(b) = 2$. Finally, the free vertex b is selected with $d(b) = 2$, corresponding to the augmenting path $P_3 = \{(1,a),(a,3),(3,c),(c,2),(2,b)\}$, with $wt(P_3) = 2$. This gives the maximum-weight matching $M = \{(1,a),(2,b),(3,c)\}$, with $wt(M) = 13$, shown in Figure 11.3.8(c). As predicted by Fact 9, the weights of the augmenting paths are nonincreasing: $wt(P_1) \geq wt(P_2) \geq wt(P_3)$.

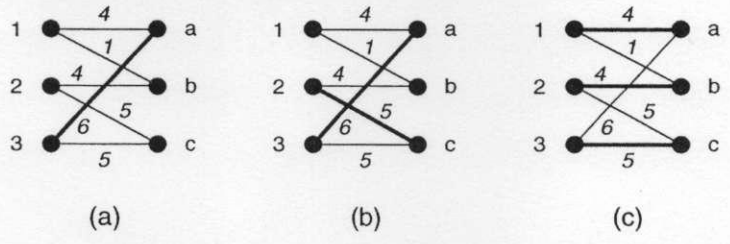

Figure 11.3.8 Maximum-weight matchings of sizes 1, 2, and 3.

11.3.3 Matchings in Nonbipartite Graphs

This section discusses matchings in more general (nonbipartite) graphs. Algorithms for constructing maximum-size and maximum-weight matchings are considerably more intricate than for bipartite graphs. The important new concept is that of a "blossom".

DEFINITIONS

D15: Suppose P is an alternating path from a free vertex s in graph $G = (V, E)$. Then a vertex v on P is ***even*** if the subpath P_{sv} of P joining s to v has even length; it is ***odd*** if P_{sv} has odd length.

D16: Suppose P is an alternating path from a free vertex s to an even vertex v and edge $(v, w) \in E$ joins v to another even vertex w on P. Then $P \cup \{(v,w)\}$ contains a unique cycle, called a ***blossom***.

D17: A ***shrunken blossom*** results when a blossom B is collapsed into a single vertex b, whereby any edge (x, y) with $x \notin B$ and $y \in B$ is transformed into the edge (x, b). The reverse of this process gives an ***expanded blossom***.