Contents

10 Blossom Algorithms for 1-Matching/Edge Covering Problems in Undirected Networks 669

10.1 The 1-Matching Problems 681

10.1.1 Blossom Algorithm for the Maximum Cardinality Matching Problem 714

10.1.2 The Minimum Cost Perfect Matching Problem . 718

10.1.3 The Minimum Cost Matching Problem 746

10.1.4 The Convex Hulls of Matching and Perfect Matching Vectors 753

10.1.5 Other Algorithms for Minimum Cost Matching Problems 754

10.1.6 Integer Valued Optimum Dual Solutions 755

10.2 1-Edge Covering Problems 757

10.2.1 Minimum Cost Edge Covers 760

10.3 Minimum Cost 1-Matching/Edge Coverings 779

10.4 The Minimum Cost 1-M/EC Problem with Specified Cardinality 807

10.5 Degree Constrained Subnetworks, *b*-Matching Problems, and the General Matching Problem 808

10.6 Exercises 809

10.7 References 820

Chapter 10

Blossom Algorithms for 1-Matching/Edge Covering Problems in Undirected Networks

In this chapter we consider only undirected networks. Let G $= (\mathcal{N}, \mathcal{A}, c = (c_{ij}))$ be a given undirected network, with c as the vector of edge weights or edge cost coefficients, and $|\mathcal{N}| = n, |\mathcal{A}| = m$. There may be parallel edges, but we assume that there are no self loops or isolated nodes in G. Each c_{ij} could be positive, 0, or negative. As defined in Section 1.2.2, a **matching** (*or* **1-matching** to be specific, also called **an independent set of edges**) in G is a subset of edges $\mathbf{M} \subset \mathcal{A}$ that contains at most one edge incident at node i, for each $i \in \mathcal{N}$. The "1" in the name "1-matching" refers to the fact that the degree in $\mathbf{M}$ of every node is required to be $\leqq 1$. Hence, every pair of distinct edges in a matching are node disjoint. In Figure 10.1, the set of thick edges is a matching. The cardinality of a maximum cardinality matching in G is known as the **edge independence number** of G, clearly this is $\leqq n/2$. The cost of a matching $\mathbf{M}$ is defined to be $\sum(c_{ij}$: over $(i; j) \in \mathbf{M})$.

A **perfect matching** (*or* **perfect 1-matching** to be specific, also called a **one factor**) in G is a matching $\mathbf{M}$ that contains exactly one edge incident at each $i \in \mathcal{N}$. If a perfect matching $\mathbf{M}$ exists in G, n must be even, and $|\mathbf{M}| = n/2$. The network in Figure 10.1 has 12 nodes, and it has the wavy perfect matching.

When referring to a matching $\mathbf{M}$, edges in it are called **matching edges**, and those not in it are called **nonmatching edges**. A node i is said to be a **matched node** in $\mathbf{M}$ if $\mathbf{M}$ contains an edge incident at i, or an **unmatched node** wrt $\mathbf{M}$ otherwise. So, $\mathbf{M}$ is a perfect matching iff there are no unmatched nodes wrt it. If, $(i; j) \in \mathbf{M}$, nodes i, j are called **mates of each other** in $\mathbf{M}$.

The algorithm for finding a minimum length postman's route (an ECR) in an undirected network G discussed in Section 1.3.8 required a minimum cost perfect matching on the complete undirected network H defined on the set of odd degree nodes in G. This is a very important application of a matching problem in routing. The minimum cost perfect matching problem is also used in an algorithm for finding a near optimum tour whose cost is guaranteed to be no more than 50% greater than the cost of an optimum tour in the Euclidean distance traveling salesman problem, see Christofides [1976]. For other applications of matching problems see Fujii, Kasami, and Ninomiya [1969], and Montreuil, Ratliff, and Goetschalckx [1987].

We will discuss efficient primal-dual algorithms called **blossom algorithms** for several different types of matching problems, the **maximum cardinality matching problem**, the **minimum cost perfect matching problem**, the problem of finding a **minimum cost matching** among all matchings, and the **parametric matching problem** of finding a minimum cost matching among all matchings of cardinality γ treating γ as an integer parameter varying between 1 and $\lfloor n/2 \rfloor$. To solve a **maximum weight matching problem** wrt an edge weight vector, define the edge cost vector to be the negative of the edge weight vector, and then find the minimum cost matching subject to the same conditions. Both problems have the same set of optimum matchings, and the minimum cost matching algorithms are shown to work for arbitrary cost vectors.

A subset of edges $\mathbf{E} \subset \mathcal{A}$ is said to **cover the nodes** in G if each node in $\mathcal{N}$ is incident to at least one edge in $\mathbf{E}$, and such a set is called an **edge cover** *or* **1-edge cover** to be specific, or a **dominating**

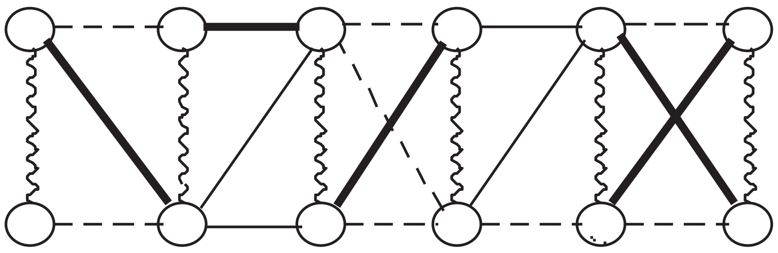

Figure 10.1: The symbols $\overset{\geq}{=}, \overset{\leq}{=}, =, 0$ inside the nodes identify the subset $\mathcal{N}^{\overset{\geq}{=}}, \mathcal{N}^{\overset{\leq}{=}}, \mathcal{N}^{=}, \mathcal{N}^{0}$ to which they belong. The set of thick edges is a 1-M/EC.

edge set in G because the edges in **E** cover or dominate all the nodes. For the network in Figure 10.1, the set of dashed edges is an edge cover. A matching is an edge cover iff it is a perfect matching. Since we assumed that there are no isolated nodes in G, $\mathcal{A}$ itself is an edge cover in G. We will discuss efficient algorithms to find minimum cardinality edge covers, and minimum cost edge covers.

Another type of covers usually discussed in graph theory are **node covers**. These are subsets of nodes $\mathbf{X} \subset \mathcal{N}$ that cover all the edges, i.e., every edge in $\mathcal{A}$ is incident to at least one node in **X**. From Exercise 1.28 we know that the node-edge incidence matrix of an undirected network is totally unimodular iff the network is bipartite. Because of this property, we can solve minimum cost node cover problems in G by linear programming methods if G is bipartite. However, if G is not bipartite, i.e., if it contains odd cycles, to find either a minimum cardinality or minimum cost node cover, are difficult problems for which no polynomially bounded algorithms are known. In contrast, in a general undirected network a minimum cost edge cover can be found with a computational effort of at most $O(n^{\mathbf{3}})$ using the blossom algorithms discussed in this chapter. We do not discuss any algorithms for node covers.

1-Matching/Edge Coverings

Let $(\mathcal{N}^{\overset{\leq}{=}}, \mathcal{N}^{=}, \mathcal{N}^{\overset{\geq}{=}}, \mathcal{N}^{0})$ be a given partition of the node set $\mathcal{N}$,

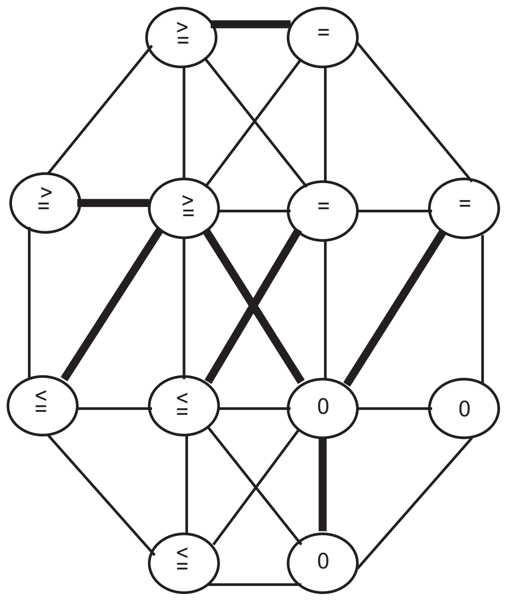

Figure 10.2: The symbols $\geqq, \leqq, =, 0$ inside the nodes identify the subset $\mathcal{N}^{\geqq}, \mathcal{N}^{\leqq}, \mathcal{N}^{=}, \mathcal{N}^0$ to which they belong. The set of thick edges is a 1-M/EC.

where some of these subsets may be empty. A subset of edges $\mathbf{E} \subset \mathcal{A}$ is called a **1-matching/edge covering** *or* **1-M/EC** in short, wrt the partition $(\mathcal{N}^{\leqq}, \mathcal{N}^{=}, \mathcal{N}^{\geqq}, \mathcal{N}^0)$ of $\mathcal{N}$ if -

$$\begin{aligned} \text{every node } i &\in \mathcal{N}^{\leqq} \text{ is incident to at most one edge in } \mathbf{E} \\ \text{every node } i &\in \mathcal{N}^{=} \text{ is incident to exactly one edge in } \mathbf{E} \\ \text{every node } i &\in \mathcal{N}^{\geqq} \text{ is incident to at least one edge in } \mathbf{E}. \end{aligned}$$

There are no constraints on the degrees of nodes in $\mathcal{N}^0$ in $\mathbf{E}$. The notation for the subsets in the partition of $\mathcal{N}$ is very suggestive. See Figure 10.2. One can easily see that 1-matching corresponds to $\mathcal{N} = \mathcal{N}^{\leqq}, \mathcal{N}^{=} = \mathcal{N}^{\geqq} = \mathcal{N}^0 = \emptyset$; 1-perfect matching corresponds to $\mathcal{N} = \mathcal{N}^{=}, \mathcal{N}^{\leqq} = \mathcal{N}^{\geqq} = \mathcal{N}^0 = \emptyset$; and 1-edge covering corresponds to $\mathcal{N} =$

$\mathcal{N}^{\geqq}, \mathcal{N}^{\leqq} = \mathcal{N}^{=} = \mathcal{N}^{0} = \emptyset$. Hence all these are special cases of 1-M/EC. We discuss efficient primal-dual blossom algorithms for the minimum cost 1-M/EC problem.

Let $\mathbf{E} \subset \mathcal{A}$ be an arbitrary subset of edges in G. Its incidence vector is the 0-1 vector $x^{\mathbf{E}} = (x^{\mathbf{E}}_{ij})$ defined on $\mathcal{A}$, where $x^{\mathbf{E}}_{ij} = 1$ if $(i; j) \in \mathbf{E}$, 0 otherwise. Conversely, given a 0-1 vector $x = (x_{ij})$ defined on $\mathcal{A}$, it is the incidence vector of (and is therefore said to correspond to) the subset of edges $\mathbf{E}^x = \{(i; j) : x_{ij} = 1\}$.

A 0-1 vector $x = (x_{ij})$ defined on $\mathcal{A}$ is said to be a **matching vector, perfect matching vector, edge covering vector,** *or* **1-M/EC vector**, if it is the incidence vector of a matching, perfect matching, edge covering, or 1-M/EC, respectively.

In this chapter an **edge vector** in G always refers to a 0-1 vector $x = (x_{ij})$ defined on $\mathcal{A}$. The variable x_{ij} associated with edge $(i; j)$ is known as the **decision variable associated with that edge** in the vector x. Given the edge vector x, we define for each $i \in \mathcal{N}$

$$x(i) = \sum (x_{ij} : \text{ over } j \text{ such that } (i; j) \in \mathcal{A}) \tag{10.1}$$

Thus an edge vector x is

$$\begin{array}{rl} \text{a matching vector} & \text{iff } x(i) \leqq 1, \text{ for each } i \in \mathcal{N} \\ \text{a perfect matching vector} & \text{iff } x(i) = 1, \text{ for each } i \in \mathcal{N} \\ \text{an edge covering vector} & \text{iff } x(i) \geqq 1, \text{ for each } i \in \mathcal{N} \\ \begin{array}{l}\text{a 1-M/EC vector wrt parti-} \\ \text{tion } (\mathcal{N}^{\leqq}, \mathcal{N}^{=}, \mathcal{N}^{\geqq}, \mathcal{N}^{0})\end{array} & \text{iff } x(i) \left\{ \begin{array}{l} \leqq 1, \text{ for } i \in \mathcal{N}^{\leqq} \\ = 1, \text{ for } i \in \mathcal{N}^{=} \\ \geqq 1, \text{ for } i \in \mathcal{N}^{\geqq} \end{array} \right. \end{array} \tag{10.2}$$

An Application of the Minimum Cost 1-M/EC Problem in Integer Programming

Consider the following pure 0-1 integer programming problem

$$\text{Minimize} \quad cy$$

$$\begin{array}{lll} \text{subject to} \quad Ay & \square\ \mathbf{e} & \\ y_j = 0 \text{ or } 1 & \text{for all } j \end{array} \qquad (10.3)$$

where $A = (a_{ij})$ is a 0-1 matrix of order $p \times q$, $\mathbf{e}$ is the column vector of all 1's in $\mathbb{R}^q$, and $\square$ denotes the vector of either $\leqq, =,$ or $\geqq$ for each constraint. The well known set covering problem with many applications in airline crew scheduling, facility location, vehicle routing, scheduling, etc., is a special case of (10.3) when $\square$ consists of all $\geqq$ symbols. (10.3) has many applications.

First consider the case in which there are at most two nonzero entries in each column of A. In this case, (10.3) is a 1-M/EC problem. To see this, let $\mathcal{N} = \{1, \ldots, p\}$ if each column of A contains exactly two nonzero entries, $\mathcal{N} = \{1, \ldots, p, p+1\}$ if some columns of A contain only one nonzero entry. For $i = 1$ to p, node i in $\mathcal{N}$ is associated with constraint i in the problem. Associate each column of A with an edge. If $A_{.j}$ contains only one nonzero entry, in row i say, associate it with the edge $(i; p+1)$ and make the cost of this edge equal to c_j. If $A_{.j}$ contains two nonzero entries; in rows h and w say; associate it with the edge $(h; w)$ and make the cost of this edge equal to c_j. Let $\mathcal{A}$ be the set of q edges associated with the columns in A. Define $\mathcal{N}^{\leqq}, \mathcal{N}^{=}, \mathcal{N}^{\geqq}$ to be the set of rows $i = 1$ to p in which the entry in $\square$ is $\leqq, =, \geqq$ respectively. Define $\mathcal{N}^0 = \emptyset$ if every column of A contains two nonzero entries, or $= \{p+1\}$ if there are some columns of A with a single nonzero entry. Let $\mathrm{G} = (\mathcal{N}, \mathcal{A})$. Then, it can be verified that a minimum cost 1-M/EC vector in G wrt the partition $(\mathcal{N}^{\leqq}, \mathcal{N}^{=}, \mathcal{N}^{\geqq}, \mathcal{N}^0)$ of $\mathcal{N}$ is an optimum solution of (10.3) and vice versa, in this case. As an example consider the following instance of (10.3). Each column of the coefficient matrix contains at most two nonzero entries among the constraints. So, this instance is equivalent to a 1-M/EC problem. The corresponding network, constructed as mentioned above, is given in Figure 10.3. The variable y_j in the problem corresponds to the edge with c_j marked on it in Figure 10.3, and c_j is the cost of this edge in the 1-M/EC problem.

Now consider the general case where there are columns in A with 3 or more nonzero entries. The best known methods for solving (10.3)

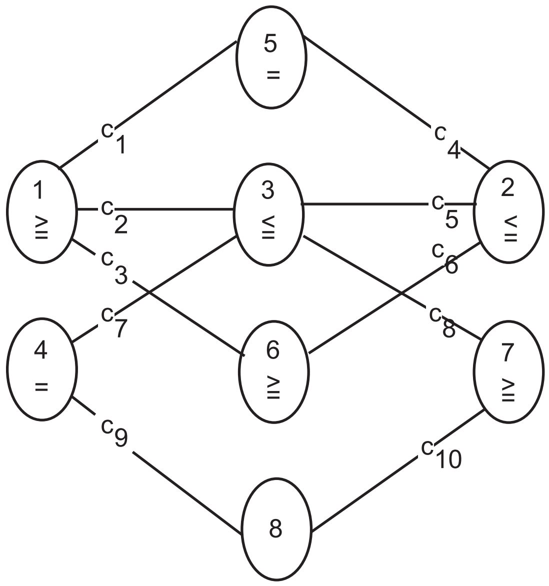

Figure 10.3: Node i is associated with constraint i. The symbol $\geqq, =, \leqq$ inside a node represents the type of constraint associated with it, and the subset $\mathcal{N}^{\geqq}, \mathcal{N}^{=}, \mathcal{N}^{\leqq}$ to which it belongs.

Constraint	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}		
1	1	1	1								$\geqq$	1
2				1	1	1					$\leqq$	1
3		1			1		1	1			$\leqq$	1
4							1		1		$=$	1
5	1			1							$=$	1
6			1			1					$\geqq$	1
7								1		1	$\geqq$	1
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	Min.	

in this case are partial enumeration methods known as branch and bound methods (see Murty [1976 of Chapter 1]). A basic operation in this method is the bounding operation whose main function is to compute a lower bound for the minimum objective value in the problem. There is a lower bounding strategy for (10.3) based on the minimum cost 1-M/EC problem. It consists of identifying a subset $\mathbf{I} \subset \{1, \dots, p\}$, so that every column of A contains at most two nonzero entries among rows in the subset $\mathbf{I}$. There may be many subsets $\mathbf{I}$ with this property, the best among these is one with as large a cardinality as possible. In branch and bound methods, a subset $\mathbf{I}$ like this is initially selected by inspection, maximizing its cardinality to the extent possible. Once $\mathbf{I}$ is selected, constraints in it are said to be *included constraints*, and those outside it are called *unincluded constraints*. In the candidate problems generated after one or more branching operations, the corresponding set of included constraints is obtained automatically through the branching strategy.

The lower bounding strategy relaxes the unincluded constraints using Lagrangian relaxation. Let $\bar{\mathbf{I}} = \{1, \dots, p\} \setminus \mathbf{I}$; $A_{\mathbf{I}}, A_{\bar{\mathbf{I}}}$ the submatrices of A consisting of all the rows in $\mathbf{I}$, $\bar{\mathbf{I}}$ respectively; and $\mathbf{e}_{\mathbf{I}}, \mathbf{e}_{\bar{\mathbf{I}}}$ the subvectors of $\mathbf{e}$ corresponding to $\mathbf{I}$, $\bar{\mathbf{I}}$ respectively. Let $u_{\bar{\mathbf{I}}} = (u_i : i \in \bar{\mathbf{I}})$ be a vector of Lagrange multipliers associated with the unincluded constraints satisfying

$$u_i \begin{cases} \leqq 0, \text{ if the } i\text{th constraint is } \leqq \\ \geqq 0, \text{ if it is } \geqq \\ \text{unrestricted, if it is } = \end{cases} \tag{10.4}$$

Then the relaxed problem is

$$\begin{array}{rrcl} \text{Minimize} & L(y, u_{\bar{\mathbf{I}}}) & = & cy - u_{\bar{\mathbf{I}}}(A_{\bar{\mathbf{I}}}y - \mathbf{e}_{\bar{\mathbf{I}}}) \\ \text{subject to} & A_{\mathbf{I}}y & \boxed{\mathbf{I}} & \mathbf{e}_{\mathbf{I}} \\ y \text{ is a} & 0, 1 & & \text{vector} \end{array} \tag{10.5}$$

where $\boxed{\mathbf{I}}$ is the subvector of the original $\square$ corresponding to the constraints in $\mathbf{I}$. In (10.5), the u_i for $i \in \bar{\mathbf{I}}$ are given constants satisfying (10.4), and only the y are the decision variables. Since each column in $A_{\mathbf{I}}$ has at most two nonzero entries, (10.5) can be solved directly as a 1-M/EC problem as discussed above. It can be shown that the optimum objective value in (10.5) is a lower bound for that in (10.3) for any $u_{\bar{\mathbf{I}}}$ satisfying (10.4). In the branch and bound method one uses a subgradient optimization procedure to select a $u_{\bar{\mathbf{I}}}$ satisfying (10.4) to make the optimum objective value in (10.5) as large as possible, in order to get a high quality lower bound for the optimum objective value in (10.3), see Chapter 15 in Murty [1976 of Chapter 1].

Eliminating Parallel Edges

In the 1-M/EC problem associated with a partition of nodes ($\mathcal{N}^{\leqq}$, $\mathcal{N}^{=}, \mathcal{N}^{\geqq}, \mathcal{N}^{0}$) of $\mathcal{N}$, an optimum solution set of edges can contain at most one edge joining a pair of nodes i, j with $i \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{=}$ and $j \in \mathcal{N}$. Hence for each such pair of nodes, if there are parallel edges joining them, keep only one of these parallel edges corresponding to the minimum cost and delete all the rest. If $\mathcal{N}^{\geqq} \cup \mathcal{N}^{0} \neq \emptyset$, consider a pair of nodes i, j in this set. If there are parallel edges joining this pair, define $\mathbf{U}(i; j)$ to be the subset of these parallel edges consisting of all of them with cost coefficient $\leqq 0$ if some of them belong to this category, or the one associated with the minimum cost among these parallel edges if all of them have cost coefficients > 0. Replace all the parallel edges with a single edge $(i; j)$ representing the set of parallel edges $\mathbf{U}(i; j)$ in the original network, and make its cost coefficient equal to the sum of the original cost coefficients of edges in $\mathbf{U}(i; j)$. If this edge $(i; j)$ is contained in an optimum 1-M/EC in the transformed network, replace it with the set of parallel edges $\mathbf{U}(i; j)$ to get an optimum 1-M/EC in the original network. So, in the sequel we assume that the network G on which our problems are defined contains no parallel edges.

The Minimum Cost 1-M/EC Problem

The minimum cost 1-M/EC problem in the network $\mathrm{G} = (\mathcal{N}, \mathcal{A}, c)$ with the partition $(\mathcal{N}^{\leqq}, \mathcal{N}^{=}, \mathcal{N}^{\geqq}, \mathcal{N}^{0})$ of $\mathcal{N}$, is the following problem (10.6), (10.7); and all the matching, edge covering problems that we consider are special cases of this problem.

The coefficient matrix of the variables in (10.6) is a submatrix ofthe node-edge incidence matrix of G. First consider the case where G is bipartite. In this case, the coefficient matrix in (10.6) is totally unimodular (Exercise 1.28) and all the BFSs of (10.6) are themselves integer vectors without even the integer constraint (10.7). Hence we can ignore

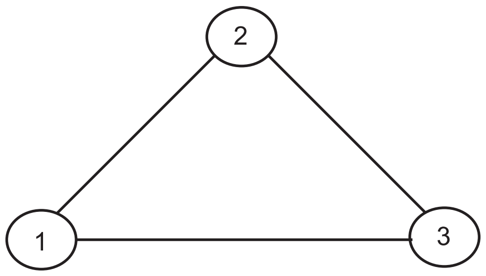

Figure 10.4:

the integer restriction (10.7) and solve (10.6) as a continuous variable LP by any algorithm that is guaranteed to terminate with an optimum BFS when feasible solutions exist, and this provides an optimum solution satisfying (10.7) automatically. Thus all these problems can be solved directly by linear programming algorithms (for example, variants of the primal-dual Hungarian method of Chapter 3) in the case when G is bipartite.

$$\begin{array}{rcl} \text{Minimize} \quad \sum(c_{ij}x_{ij} & : & \text{over } (i;j) \in \mathcal{A}) \\ \text{Subject to} \quad x(i) & & \left\{ \begin{array}{l} \leqq 1, \text{ for } i \in \mathcal{N}^{\leqq} \\ = 1, \text{ for } i \in \mathcal{N}^{=} \\ \geqq 1, \text{ for } i \in \mathcal{N}^{\geqq} \end{array} \right. \\ 0 \leqq x_{ij} \leqq 1 & & \text{for all } (i;j) \in \mathcal{A} \end{array} \tag{10.6}$$

$$\text{and} \quad x_{ij} \text{ integer} \qquad \text{for all } (i;j) \in \mathcal{A} \tag{10.7}$$

Now consider the case where G is not bipartite, i.e., it contains some odd cycles. In this case (10.6) may have extreme point solutions which violate (10.7); so (10.6), (10.7) is a genuine integer programming problem. As an example, consider (10.8), (10.9), the minimum cost perfect matching problem in the network in Figure 10.4, with all edge cost coefficients equal to -1.

$$\text{Minimize} \quad -x_{12} - x_{13} - x_{23}$$

$$\begin{array}{rrcl} \text{subject to} & x_{12} + x_{13} \qquad & = & 1 \\ & x_{13} + x_{23} & = & 1 \\ & x_{12} \qquad + x_{23} & = & 1 \\ & x_{12}, x_{13}, x_{23} & \geqq & 0 \end{array} \tag{10.8}$$

$$x_{12}, x_{13}, x_{23} \qquad \text{integer} \tag{10.9}$$

Ignoring (10.9), the unique optimum solution of (10.8) is $\overline{x} = (\overline{x}_{12}, \overline{x}_{13}, \overline{x}_{23})^T = (1/2, 1/2, 1/2)^T$, which is a noninteger BFS of (10.8). Actually, (10.8), (10.9) together have no feasible solution.

Again consider the minimum cost edge covering problem in the network in Figure 10. 4, with a cost coefficient of 1 for each edge. This problem is

$$\begin{array}{rrcl} \text{Minimize} & x_{12} + x_{13} + x_{23} & & \\ \text{subject to} & x_{12} + x_{13} \qquad & \geqq & 1 \\ & x_{13} + x_{23} & \geqq & 1 \\ & x_{12} \qquad + x_{23} & \geqq & 1 \\ & 0 \leqq x_{12}, x_{13}, x_{23} & \leqq & 1 \end{array} \tag{10.10}$$

$$x_{12}, x_{13}, x_{23} \qquad \text{integer} \tag{10.11}$$

It can be verified that $\overline{x} = (1/2, 1/2, 1/2)^T$ with an objective value of 3/2, is again the unique optimum solution of (10.10) without (10.11). The optimum objective value in (10.10), (10.11) together is 2. These examples indicate that matching and edge covering problems in nonbipartite networks are nontrivial integer programs. The thing that makes them nontrivial is the presence of odd cycles in nonbipartite networks.

The approach taken to solve these integer programs is to develop additional linear constraints (usually called **blossom constraints**) such that every extreme point solution of the system consisting of (10.6) and the blossom constraints, is a feasible solution of (10.6), (10.7), and vice versa. One then relaxes the integer restrictions (10.7), and solves the LP (10.6) with the additional blossom constraints, by primal-dual or primal methods leading to extreme point optimum solutions when the problem is feasible. The primal-dual algorithms of this type are called **blossom algorithms**.

The Symmetric Assignment Problem

Consider the minimum cost perfect matching problem in the undirected network $\mathrm{G} = (\mathcal{N}, \mathcal{A}, c)$ with $|\mathcal{N}| = n$ even. If G is a bipartite network with $(\mathcal{N}_1, \mathcal{N}_2)$ as its bipartition, where $|\mathcal{N}_1| = |\mathcal{N}_2| = n/2$, we have seen in Chapter 3 that the minimum cost perfect matching problem in G is an assignment problem of order $n/2$. Even when G is nonbipartite, the minimum cost perfect matching problem in G can still be posed as an assignment problem, but with additional constraints. In this case the formulation leads to an assignment problem of order n (in comparison to the bipartite case where we had an assignment problem of order $n/2$), and the additional constraints are symmetry constraints, hence it is called a **symmetric assignment problem**.

Set up an $n \times n$ transportation array. In this array associate both row i and column i with node i in G, for $i = 1$ to n. For $i \neq j$, associate the edge $(i; j)$ in G with the pair of cells (i, j) and (j, i) in the array. Hence, in this formulation, the cell (i, j) in the array is never individually considered by itself, it is always considered together with the cell (j, i). If the edge $(i; j)$ is included in the matching, both the cells (i, j) and (j, i) will be cells with allocations in the corresponding assignment, and vice versa. With this convention any perfect matching in G corresponds to an assignment $y = (y_{ij})$ of order n, in which $y_{ij} = y_{ji}$ for all i, j. These are the symmetry constraints, and assignments satisfying them are called **symmetric assignments**. Every perfect matching in G corresponds to a symmetric assignment y with no allocations along the main diagonal (i.e., $y_{ii} = 0$ for all $i = 1$ to n) and vice versa. Let d_{ij} denote the cost coefficient in cell (i, j) in the array. To guarantee that the cost of any perfect matching in G will be the same as that of the symmetric assignment corresponding to it in the array, we need to make sure that $d_{ij} + d_{ji} = c_{ij}$ for each edge $(i; j) \in \mathcal{A}$. In practice one may prefer to take $d_{ij} = d_{ji} = (1/2)c_{ij}$ for every $(i; j)$. With this cost matrix $d = (d_{ij})$ defined on the array, the minimum cost perfect matching problem in G is equivalent to the symmetric assignment problem in the array.

From the results in Chapter 1, we know that the matrix of coefficients of the usual assignment constraints (for example, see (3.1) in Chapter 3) is totally unimodular. Unfortunately, when the symmetry constraints are added to this system, this total unimodularity property is lost. Because of this, there is no guarantee that all the BFSs of the symmetric assignment problem will be integer vectors. Hence in the symmetric assignment problem, the integer requirements on the variables have to be taken into consideration and cannot be ignored. Thus, the formulation of the minimum cost perfect matching problem as a symmetric assignment problem, does not lead to any direct linear programming approaches to solve it.

10.1 The 1-Matching Problems

In this section we discuss results on the 1-matching problems on G $= (\mathcal{N}, \mathcal{A}, c)$ with $|\mathcal{N}| = n, |\mathcal{A}| = m$, and efficient algorithms of worst case computational complexity O($n^{\mathbf{3}}$) for solving them.

Alternating and Augmenting Paths, APs, Alternating Trees

In Chapter 3 we have seen that alternating paths are a key ingredient in the Hungarian method for the assignment problem, which is the minimum cost perfect matching problem in a bipartite network. By the 1950's it was realized that alternating paths play a key role in algorithms for matching problems in nonbipartite networks as well. Let **M** be a given matching in G. A simple path $\mathcal{P}$ in G is said to be an **alternating path** wrt **M**, or an **M-alternating path** if it satisfies the following conditions (10.12). Usually the matching **M** is understood from the context, then we simply refer to the path as being alternating. By definition, all nodes on an alternating path, except possibly the end nodes, are matched nodes.

Edges in $\mathcal{P}$ are alternately in **M** and not in **M**. If any of the two end nodes (these are nodes incident to only one edge of $\mathcal{P}$) is a matched node, $\mathcal{P}$ contains the matching edge incident at it. (10.12)

Let $\mathcal{P}$ be an alternating path wrt a matching $\mathbf{M}$. Let $\mathbf{A}_1, \mathbf{A}_2$ be the sets of matching, nonmatching edges on $\mathcal{P}$. Let $\mathbf{M}_1 = \mathbf{A}_2 \cup (\mathbf{M} \backslash \mathbf{A}_1)$. Clearly, $\mathbf{M}_1$ is obtained by making all the edges in $\mathbf{A}_1$ (the original matching edges on $\mathcal{P}$) into nonmatching edges, and all the edges in $\mathbf{A}_2$ (the original nonmatching edges on $\mathcal{P}$) into matching edges; and it is another matching in G. We say that $\mathbf{M}_1$ is obtained from $\mathbf{M}$ by **changing the polarity of matching and nonmatching edges on** $\mathcal{P}$, or just by **rematching M using** $\mathcal{P}$.

An **augmenting path** wrt a matching $\mathbf{M}$ is an alternating path both of whose end nodes are unmatched nodes. An augmenting path always contains an even number of nodes and an odd number of edges. If it contains $2r + 1$ edges, r are matching edges, and the remaining $r + 1$ are nonmatching edges. Thus the operation of rematching using an augmenting path, increases the cardinality of the matching by one, and hence is called an **augmentation step**. The name augmenting path stems from this fact.

As an example, consider the paths $\mathcal{P}_1$ = 2, (2, 5), 5, (5, 8), 8, (8, 9), 9 ; $\mathcal{P}_2$ = 1, (1, 2), 2, (2, 5), 5 ; $\mathcal{P}_3$ = 1, (1, 3), 3, (3, 4), 4, (4, 6), 6 in the network in Figure 10.6. All these are alternating paths wrt the wavy matching. Rematching the wavy matching using $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ leads to the matchings: $\mathbf{M}_1$ = {(5, 8), (7, 10), (3, 4)}, $\mathbf{M}_2$ = {(1, 2), (8, 9), (7, 10), (3, 4)}, $\mathbf{M}_3$ = {(1, 3), (4, 6), (7, 10), (8, 9), (2, 5)} respectively, in the network in Figure 10.6. It can be verified that among these three alternating paths, only $\mathcal{P}_3$ is an augmenting path wrt the wavy matching in Figure 10.6, and rematching using it increases the cardinality of the matching from 4 to 5.

Verify that there is no augmenting path wrt the wavy matching in the network in Figure 10.14.

An augmenting path begins at an unmatched node, and by moving alternately through nonmatching and matching edges, it reaches another unmatched node. To trace an augmenting path, it is convenient to label the points along it alternately as **outer** (labeled with a +) or **inner** (labeled with a −) nodes, beginning with an outer label for the initial unmatched node. The last node will be an unmatched node labeled as an inner node. All nodes which are reached after passing through an odd (even) number of edges are inner (outer) nodes. That's why in some books outer, inner nodes are called even, odd nodes respectively. We will now discuss a theorem due to Berge [1957] and Norman and Rabin [1959] that shows the importance of augmenting paths in the maximum cardinality matching problem.

THEOREM 10.1 *A matching* $\mathbf{M}$ *in* G *is a maximum cardinality matching iff there exists no augmenting path* wrt *it.*

Proof If there exists an augmenting path wrt $\mathbf{M}$, augmentation using it leads to a matching of cardinality $1 + |\mathbf{M}|$, and hence $\mathbf{M}$ is not a maximum cardinality matching.

Now suppose $\mathbf{M}$ is a matching in G that is not of maximum cardinality. Let $\mathbf{M}^*$ be a maximum cardinality matching in G. So, $|\mathbf{M}^*| \geqq 1+|\mathbf{M}|$. Let $\mathbf{A} = (\mathbf{M} \cup \mathbf{M}^*)\backslash(\mathbf{M} \cap \mathbf{M}^*)$. Since both $\mathbf{M}$, $\mathbf{M}^*$ are matchings, each of them contains at most one edge incident at any node. Hence $\mathbf{A}$ is the disjoint union of some simple paths and simple cycles. Each simple cycle in $\mathbf{A}$ consists alternately of an edge in $\mathbf{M}$ and an edge in $\mathbf{M}^*$, and hence contains an equal number of edges from $\mathbf{M}$ and from $\mathbf{M}^*$. Each simple path in $\mathbf{A}$ consists alternately of an edge in $\mathbf{M}$ and an edge in $\mathbf{M}^*$. Since $|\mathbf{M}^*| > |\mathbf{M}|$, these facts imply that there must exist a simple path in $\mathbf{A}$ which contains more edges from $\mathbf{M}^*$ than from $\mathbf{M}$, and clearly that path will be an augmenting path wrt $\mathbf{M}$. So, if $\mathbf{M}$ is not a maximum cardinality matching, there exists an augmenting path wrt it. ∎

As an example consider the network in Figure 10.5 and the two matchings $\mathbf{M}$(thick), $\mathbf{M}^*$(wavy) in it, with one common edge (17; 18).

The set $\mathbf{A} = (\mathbf{M} \cup \mathbf{M}^*)\backslash(\mathbf{M} \cap \mathbf{M}^*)$ consists of an even alternating cycle containing the edges (11; 16), (14; 15), (12; 13) from $\mathbf{M}$, and the edges (11; 12), (13; 14), (15; 16) from $\mathbf{M}^*$; and three augmenting paths wrt $\mathbf{M}$. They are the single edge augmenting path (8, 9); the second 1, (1, 2), 2, (2, 3), 3, (3, 4), 4; and the third 5, (5, 6), 6, (6, 7), 7, (7, 10), 10.

Hence the maximum cardinality matching problem in G can be solved by starting with an arbitrary matching (for example the empty matching), searching for an augmenting path wrt it; and if one is found, carrying out augmentation using it, and repeating the whole process

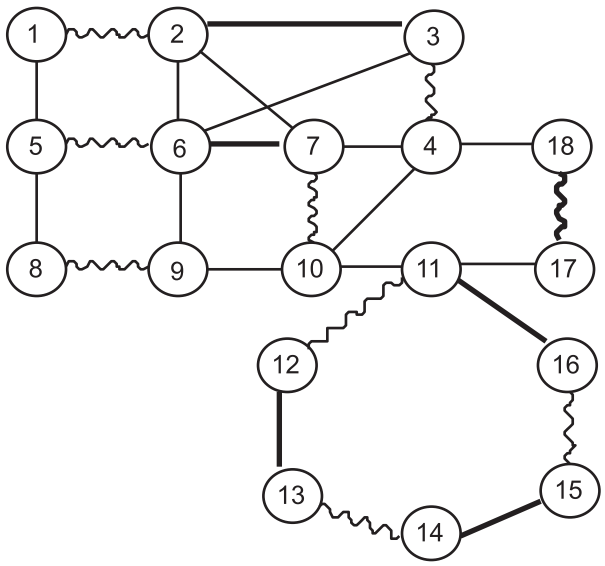

Figure 10.5: $\mathbf{M}(\mathbf{M}^*)$ is the matching of thick (wavy) edges. (17; 18) is the only edge that is in both matchings.

with the new matching, until we reach a matching wrt which there is no augmenting path. To implement this approach, we need an efficient method for finding an augmenting path if one exists, or determining that no augmenting path exists. The labeling routine discussed in Section 3.1 provided a very efficient method for doing this in bipartite networks. When the same labeling routine is applied on a nonbipartite network, we will see that odd cycles in the network pose obstacles in discovering augmenting paths.

Often, we have to deal with simple paths which have the property that edges in it are alternately from a matching $\mathbf{M}$ and not from it, we will use the symbol "**AP**" to denote such paths. So, an AP between nodes i, j wrt a matching $\mathbf{M}$ is a simple path with its edges alternately in $\mathbf{M}$ and out of $\mathbf{M}$. An AP may not be an alternating path as defined in (10.12) because it may not satisfy the second property in (10.12). For example the path 2, (2, 3), 3, (3, 4), 4, (4, 6), 6 in Figure 10.6 is an AP wrt the wavy matching, but it is not an alternating path as defined in (10.12) because it does not contain the matching edge (2; 5) incident at its end node 2. While we are developing alternating paths in matching algorithms, in intermediate steps they will actually be APs because they may not satisfy the second condition in (10.12) at that time. But any path with which we carry rematching operation will always be an alternating path as defined in (10.12).

An **alternating tree** wrt a matching **M** in G is a tree rooted at an unmatched node and satisfying the property that the predecessor path of every nonroot node in it is an AP (in the sense that it consists alternately of matching and nonmatching edges). The labeling routine of Section 3.1 to find an augmenting path, is equivalent to rooting an alternating tree at an unmatched node, and growing it. In this procedure list always refers to the set of labeled and unscanned nodes. The procedure is initiated by selecting an unmatched node, r say, and making it the root of an alternating tree by labeling it as an **outer node** with the label $(\emptyset, +)$. At this stage the list $= \{r\}$. The tree is grown by selecting nodes from the list in some order and scanning them using the following rules.

(i) Scanning an Outer Node i Find all unlabeled nodes j satisfying $(i;j) \in \mathcal{A}, (i;j) \notin \mathbf{M}$, label them as **inner nodes** with the label $(i, -)$, and include them in the list. Node i is their **immediate predecessor** *or* **predecessor index**, and all these nodes are the *immediate successors* of i. Now delete i from the list.

(ii) Scanning an Inner Node i If i is unmatched, an augmenting path has been found, and the tree is said to have become an **augmenting tree**. The augmenting path is the predecessor path of i. Terminate labeling.

If i is matched, and its mate j is unlabeled, label it with $(i, +)$ and include it in the list. In this case i is the immediate predecessor or predecessor index of j, and j is the only successor of i. Now delete i from the list.

At any stage, the set of in-tree edges is the set of edges joining the labeled nodes to their immediate predecessors. Alternating trees can be verified to satisfy the following properties: The predecessor path of any outer (inner) node consists of an even (odd) number of edges. Each inner node has at most one immediate successor, and if it has one, the edge joining them is a matching edge. If a node has two or more immediate successors, that node must be an outer node. Each inner node is incident to at most two in-tree edges, one of which must be a nonmatching edge. Each outer node is incident to any number of nonmatching in-tree edges, and one matching edge, excepting the root which meets no matching edge. If i is a non-root outer node with label (P(i), +), then P(i) must be its mate and an inner node. And the in-tree simple path from any outer node to a descendent inner node has an odd number of edges in it.

As an example, consider the network in Figure 10.6 with the wavy matching in it. An alternating tree is rooted at the unmatched node 1 in this network and grown. Nodes are scanned in the order 1, 2, 3, 4, and node labels are entered by the side of the nodes. When node 4 is scanned, the unmatched node 6 became labeled and the tree became an augmenting tree. The augmenting path is the predecessor path of node 6, it is 6, (6, 4), 4, (4, 3), 3, (3, 1), 1. All the properties mentioned above can be verified to hold in this tree.

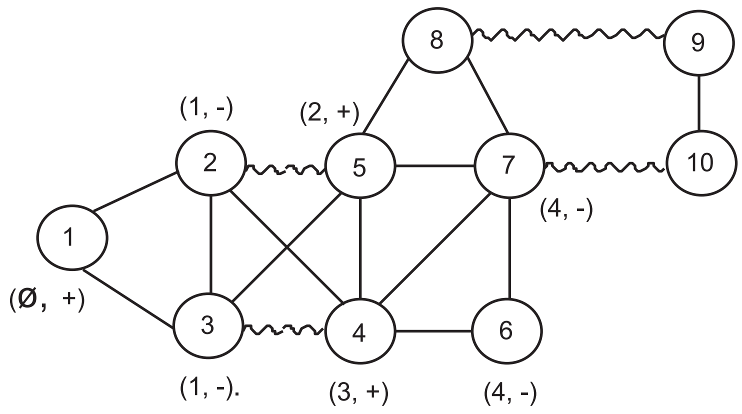

Figure 10.6: Matching edges are wavy. The tree became augmenting when unmatched node 6 is labeled.

In Section 3.1, 3.2, this scheme was applied on bipartite networks.

There, outer nodes correspond to rows of the transportation array, and inner nodes to columns. Hence, the possibility of a pair of outer nodes, or a pair of inner nodes, being joined by an edge in the network never arises there. However when this labeling scheme is applied on a nonbipartite network, it may happen that there exists an edge joining a pair of outer nodes, or a pair of inner nodes (which indicates the existence of odd cycles in the network) and this may prevent this simple labeling scheme from discovering augmenting paths even when they exist. As an example of this consider the network in Figure 10.7 with the wavy matching in it. An alternating tree is rooted at the unmatched node 1 and grown by scanning the nodes in the list in serial order. Node labels are entered by the side of the nodes. The path at the bottom of the network is an augmenting path, but the simple labeling scheme has been unable to discover it because the unmatched node 12 could not be labeled.

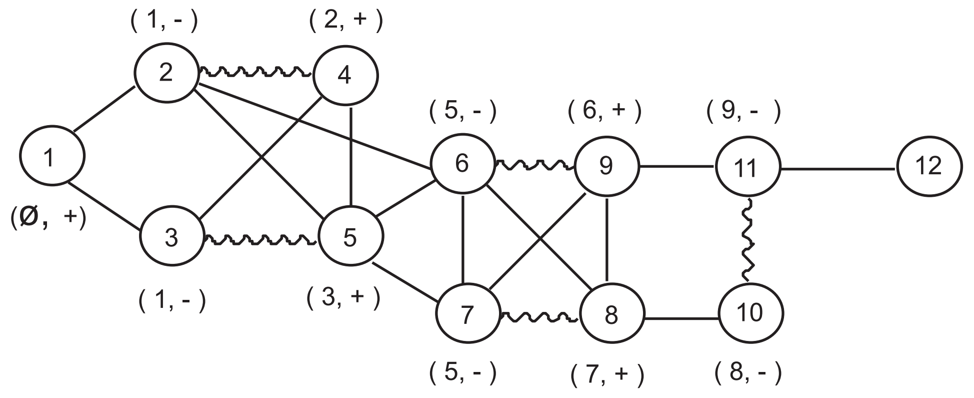

Figure 10.7: Matching edges are wavy. The augmenting path between 1 and 12 is not discovered when nodes are scanned in serial order.

Simple Blossoms

The labeling scheme failed to discover the augmenting path in the network in Figure 10.7 because of the odd alternating cycle encountered when nodes 10, 11 on the matching edge (10; 11) are both labeled as inner nodes. This is the cycle marked with thick solid lines in the following Figure 10.8.

In general, an **odd alternating cycle** is encountered whenever a pair of nodes on a matching (nonmatching) edge are both labeled as inner (outer) nodes. Such a pair is called the **identifying pair of nodes** for the odd alternating cycle. Let i, j be the identifying pair of nodes. The first common node on the predecessor paths of i, j is known as the **base node** for the odd alternating cycle, suppose it is t. Node t has at least two successors, one on each of the predecessor paths of i and j, and possibly others not on these paths. So, t must be an outer node. Hence, the two edges incident at t on the predecessor paths of i and j must be nonmatching edges. If i, j are both inner (outer) nodes, the in-tree paths from t to i, and from t to j, are edge disjoint paths, each containing an odd (even) number of edges. Hence in either case, the in-tree paths from t to i, and from t to j, together with edge $(i; j)$ form an odd alternating cycle. Let $\mathcal{N}_B$ be the set of nodes on this odd cycle, and $\mathcal{A}_B = \{(i; j) : (i; j) \in \mathcal{A}; i, j \in \mathcal{N}_B\}$. $B = (\mathcal{N}_B, \mathcal{A}_B)$ is the partial network of G determined by $\mathcal{N}_B$. B has the following properties.

B is a partial network determined by an odd subset of nodes of cardinality $\geqq 3$.

There is exactly one node in B, defined to be its **base node** which is not incident to a matching edge within B. All other nodes in B are incident to a matching edge within B. The base may be incident to a matching edge which is not an edge in B. (10.13)

There is a simple spanning cycle in B containing all the nodes in B, in which the edges incident at the base are nonmatching edges, and the others are alternately matching and nonmatching edges. This is the spanning odd alternating cycle in B.

A partial network B of G satisfying (10.13) is called a **simple blossom** wrt the present matching $\mathbf{M}$. B also satisfies the following additional property.

> The base t of B is either an unmatched node (if it is the root node); or there exists an alternating path from t beginning with a matching edge, to an unmatched node (the root node), with all its edges outside B. (10.14)

If the base node t is matched, its predecessor path meets the requirement in (10.14), and is called the **stem** of the simple blossom B. If the base node t is unmatched, it is the root node itself, and the stem is empty (i.e., contains no edges), and in this case B is called a **rooted simple blossom**. Since the base node is always an outer node, it can be verified that the stem always contains an even number (which could be 0) of edges. As an example, consider the alternating tree in the network in Figure 10.7. The two inner nodes 11, 10 joined by a matching edge are the identifying pair of nodes for a simple blossom whose base node is 5. The set of nodes on this simple blossom is $\mathcal{N}_B = \{5, 6, 9, 11, 10, 8, 7\}$. See Figure 10.8.

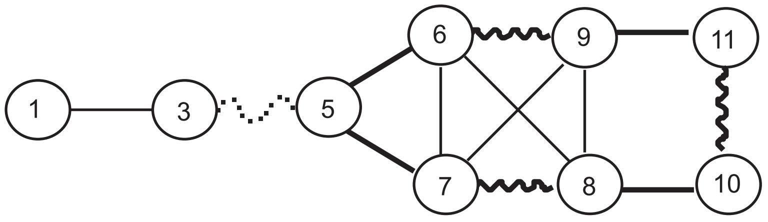

Figure 10.8: Matching edges are wavy. Simple blossom with base 5 is in solid lines. Odd cycle is thick. Dotted portion is the stem.

An odd cycle is identified whenever the pair of nodes on an edge, $(h_1; h_2)$ say, are both labeled as outer or inner nodes. Suppose h_1, h_2 are both labeled as outer nodes. If $(h_1; h_2)$ is a matching edge, both h_1, h_2 are nonroot nodes since the root is unmatched, and if $\mathrm{P}(h_1)$, $\mathrm{P}(h_2)$ are the immediate predecessors of h_1, h_2 respectively, both $\mathrm{P}(h_1)$, $\mathrm{P}(h_2)$ must be inner nodes, so $\mathrm{P}(h_1) \neq h_2$, and $(\mathrm{P}(h_1); h_1)$ must be a matching edge, a contradiction since the two edges $(h_1; h_2)$, $(\mathrm{P}(h_1); h_1)$

incident at h_1 cannot both be matching edges. So, $(h_1; h_2)$ must be a nonmatching edge, and this is a signal that a simple blossom has been identified.

Suppose h_1, h_2 are both labeled as inner nodes. In this case, if $(h_1; h_2)$ is a nonmatching edge, an odd cycle in the network is of course identified, but this odd cycle does not satisfy the last two properties in (10.13) to qualify as a simple blossom. For example, in the alternating tree in Figure 10.7, nodes 6, 7 on the nonmatching edge are both inner nodes. The odd cycle in the union of the edge (6; 7), and the predecessor paths of nodes 6, 7 is 6, (6, 5), 5, (5, 7), 7, (7, 6), 6, and this cycle does not satisfy the last two properties in (10.13). Such odd cycles do not pose obstacles in the way of the labeling scheme from identifying augmenting paths, only simple blossoms do.

Whenever a simple blossom is identified in the process of growing an alternating tree, we say that the tree has **blossomed**. By recognizing simple blossoms when they occur, and developing methods for handling them, the labeling scheme can be modified into a method that is guaranteed to find an augmenting path if one exists. The method of handling simple blossoms involves an operation called **shrinking them**, and later on expanding them when necessary.

We will now describe some of the properties of simple blossoms. Let $B = (\mathcal{N}_B, \mathcal{A}_B)$ be a simple blossom wrt the matching $\mathbf{M}$ in G, with node t as the base node. Let $\mathbf{M}_B = \mathbf{M} \cap \mathcal{A}_B$. Then $\mathbf{M}_B$ is a maximum cardinality matching in B, since all the nodes in B other than the base node t are matched in it. There can be at most one matching edge joining a node outside a simple blossom to a node inside it, and if there is such a matching edge, it must be incident at the base node.

If i is an inner node in B, the portion of the predecessor path of i up to the base node t, $\mathcal{P}_1$, is an AP between i and t, consisting of an odd number of edges, and beginning with the nonmatching edge incident at i on the odd cycle in B. By deleting all the edges on $\mathcal{P}_1$ from the odd alternating cycle in B, we are left with another AP between i and t, this one begins with the matching edge incident at i and has an even number of edges.

If i is a non-base outer node in B, the portion of the predecessor path of i up to t, $\mathcal{P}_2$, is an AP between i and t, consisting of an even number of edges and beginning with the matching edge incident at i. The AP between i and t consisting of an odd number of edges is obtained by deleting all the edges on $\mathcal{P}_2$ from the odd alternating cycle, this path begins with the nonmatching edge incident at i.

As an example, consider the non-base node 9 in the simple blossom in Figure 10.8. The AP between 9 and the base node 5 consisting of an even number of edges is 9, (9, 6), 6, (6, 5), 5; and the AP consisting of an odd number of edges is 9, (9, 11), 11, (11, 10), 10, (10, 8), 8, (8, 7), 7, (7, 5), 5.

A simple blossom has an empty stem iff its base is the root of the alternating tree, i.e., it is a rooted simple blossom. The simple blossom in Figure 10.7, with 4, 5 as the identifying pair of nodes is a rooted simple blossom.

Shrinking a Simple Blossom into a Pseudonode

Let $B = (\mathcal{N}_B, \mathcal{A}_B)$ be a simple blossom wrt a matching $\mathbf{M}$ in G, discovered during the growth of an alternating tree. At this stage we store this simple blossom (its base node, its identifying pair of nodes, and all the nodes on it together with the present labels on them). This stored information is sufficient to recognize all the current matching edges in this simple blossom (for example, if $i \in \mathcal{N}_B$ has label $(g, +)$[$(g, -)$], then $(g; i)$ is a current matching [nonmatching] edge on the odd alternating cycle in B); and the APs between any non-base node i and the base node, beginning with the matching or nonmatching edge incident at i on the odd alternating cycle in B. Then we perform an operation called **shrinking the simple blossom** B into a single new node known as a **pseudonode** (to distinguish it from the original nodes), which we denote by the same symbol B. This operation is essentially that of replacing all the present nodes in the simple blossom B by the single pseudonode B. This operation is also called **contraction** in graph theory.

Introduce the pseudonode B. For each $p \in \mathcal{N} \backslash \mathcal{N}_B$, if p is joined to one or more nodes in $\mathcal{N}_B$ by edges in $\mathcal{A}$ of which there is a matching edge, introduce a new edge $(p; B)$ and make it a matching edge. If all the edges in $\mathcal{A}$ joining p to nodes in $\mathcal{N}_B$ are nonmatching edges, make the new edge $(p; B)$ into a nonmatching edge.

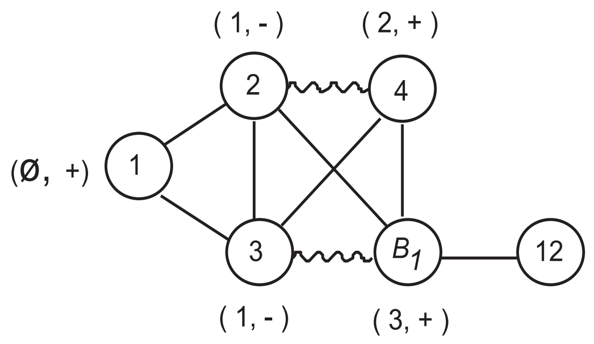

Figure 10.9: Network after shrinking the simple blossom in Figure 10.8 in the original network in Figure 10.7, into the pseudonode B_1.

Then eliminate all the nodes in $\mathcal{N}_B$, and all the edges in $\mathcal{A}$ which are incident to at least one node in $\mathcal{N}_B$. In the resulting network, give the pseudonode B the same label as that on the base of the simple blossom B, so, it will be an outer node. For each $p \in \mathcal{N} \backslash \mathcal{N}_B$ which was an immediate successor of a node in the simple blossom B before the shrinking, change the immediate predecessor of p into the pseudonode B, but leave its inner, outer status unchanged. Now the shrinking operation is completed. Let G^1 denote the resulting network, and $\mathbf{M}^1$ the remaining matching in it. G^1 is said *to have been obtained by shrinking the simple blossom B into the pseudonode B*. $\mathbf{M}^1$ is a matching in G^1, and it contains an edge incident at B iff the base in the simple blossom B was a matched node (if t was the base node of the simple blossom B, and $(t; g)$ was the matching edge incident at it in $\mathbf{M}$, $\mathbf{M}^1$ contains $(B; g)$). Define the base of the pseudonode B to be the base of the simple blossom B. The labels on nodes in G^1 define an alternating tree wrt $\mathbf{M}^1$. Thus, after the shrinking of a simple blossom, a matching and an alternating tree are available in the resulting network. G^1 is known as the **current network**, and $\mathbf{M}^1$, the **current matching** in it.

The pseudonode B is an unmatched node in G^1, iff the base of the simple blossom B was the root, in this case pseudonode B is called a **rooted pseudonode**, it becomes the root node in the alternating tree in G^1. This shows that the number of unmatched nodes in G^1 wrt $\mathbf{M}^1$, is exactly the same as the number of unmatched nodes in G wrt $\mathbf{M}$.

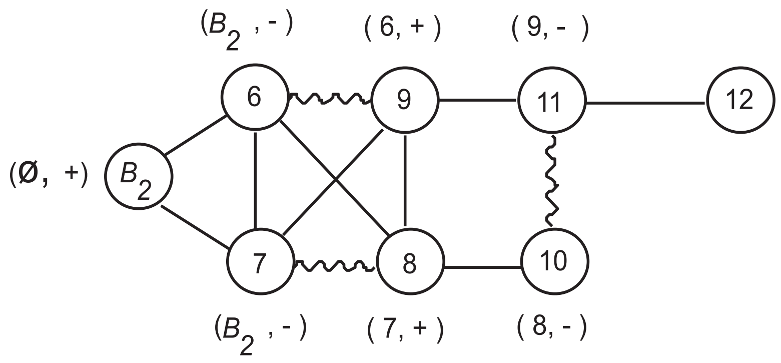

Figure 10.10: Network after shrinking the simple blossom defined by {4,2,1,3,5} in the original network in Figure 10.7, into the pseudonode B_2.

Also, the number of nodes in G^1 is $|\mathcal{N}| - |\mathcal{N}_B| + 1$, and since $|\mathcal{N}_B| \geqq 3$ and odd, this number is $\leqq |\mathcal{N}| - 2$. Thus the operation of shrinking a simple blossom into a pseudonode reduces the total number of nodes in the network by an even number $\geqq 2$, but leaves the total number of unmatched nodes unchanged.

The importance of simple blossoms and the process of shrinking them, arises from the following procedure, and Theorem 10.2 that comes later.

Procedure for Deriving an Augmenting Path $\mathcal{P}$ in G from an Augmenting Path $\mathcal{P}^1$ in G^1

Let **M** be a matching in the original network $G = (\mathcal{N}, \mathcal{A})$, and $B = (\mathcal{N}_B, \mathcal{A}_B)$ a simple blossom in G wrt **M** with base node t, which is shrunk into the pseudonode B resulting in the current network G^1 and current matching $\mathbf{M}^1$. Let $\mathcal{P}^1$ be an augmenting path in G^1 wrt $\mathbf{M}^1$. Here we discuss a procedure for obtaining an augmenting path $\mathcal{P}$ wrt **M** in G from $\mathcal{P}^1$.

If $\mathcal{P}^1$ does not contain the pseudonode B, all the nodes and edges on $\mathcal{P}^1$ are original nodes and edges in G itself, and hence $\mathcal{P}^1$ itself is an augmenting path wrt

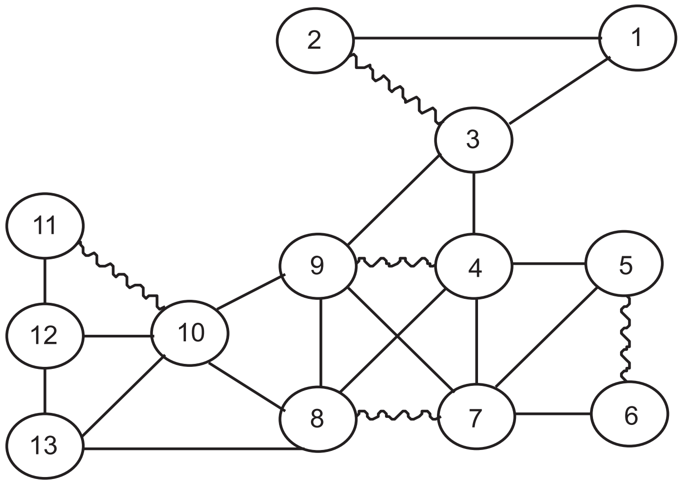

Figure 10.11: Network G. $\mathbf{M}$ is the wavy matching. B the partial network of $\mathcal{N}_B = \{5,4,9,10,8,7,6\}$, is a simple blossom.

$\mathbf{M}$ in G. Now suppose $\mathcal{P}^1$ contains the pseudonode B. We consider two cases.

Case 1 B is an end node on $\mathcal{P}^1$. Since $\mathcal{P}^1$ is an augmenting path in G^1, B must be unmatched in this case. Let $(q;B)$ be the edge in $\mathcal{P}^1$ incident at B. There must exist an $i \in \mathcal{N}_B$ such that $(q;i) \in \mathcal{A}$, and since B is unmatched, $(q;i) \notin \mathbf{M}$. If $i = t$ let $\mathcal{P}'$ be the edge $(q;t)$. If $i \neq t$, let $\mathcal{P}'$ be the path between q and t consisting of $(q;i)$, and the AP from i to t beginning with the matching edge incident at i in the odd cycle in the simple blossom corresponding to B. Replace the edge $(q;B)$ in $\mathcal{P}^1$ by the path $\mathcal{P}'$, this leads to $\mathcal{P}$, an augmenting path in G wrt $\mathbf{M}$.

Case 2 B is an intermediate node on $\mathcal{P}^1$. In this case there exists an edge $(q_1;B) \in \mathbf{M}^1$, and an edge $(B;q_2) \notin \mathbf{M}^1$ incident at B on $\mathcal{P}^1$. So, $(q_1;t) \in \mathbf{M}$, and there exists an $i \in \mathcal{N}_B$ such that $(i;q_2) \in \mathcal{A}\backslash\mathbf{M}$. If $i = t$ let $\mathcal{P}'$ be $q_1,(q_1,t),t,(t,q_2),q_2$. If $i \neq t$ let $\mathcal{P}'$ be the AP between q_1 and q_2 beginning with the matching edge $(q_1;t)$, then going from t to i using the AP on the odd cycle in the simple blossom corresponding to B ending with the matching edge incident at i, and then finally the

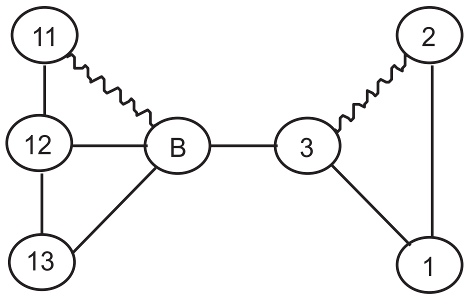

Figure 10.12: Network G^1 obtained after shrinking B in Figure 10.11 into the pseudonode B.

nonmatching edge $(i; q_2)$. Replace the pair of edges $(q_1; B), (B; q_2)$ on $\mathcal{P}^1$ by the path $\mathcal{P}'$, this converts it into an augmenting path $\mathcal{P}$ in G wrt $\mathbf{M}$.

In both cases the augmenting path $\mathcal{P}$ in G is said to have been obtained by **expanding the pseudonode** B on $\mathcal{P}^1$. This operation of expanding the pseudonode B on $\mathcal{P}^1$ uses the stored information about the simple blossom corresponding to B.

As an example consider the current network in Figure 10.10 obtained by shrinking the simple blossom determined by the set of nodes {4, 2, 1, 3, 5} in the original network in Figure 10.7 into the pseudonode B_2. B_2, (B_2, 7), (7, 8), 8, (8, 10), (10, 11), 11, (11, 12), 12 is an augmenting path wrt the wavy matching in the network in Figure 10.10. By expanding the pseudonode B_2 on it as described in Case 1 above, we get the augmenting path between nodes 1 to 12 at the bottom of the original network in Figure 10.7.

For another example consider the original network G in Figure 10.11, and the current network G^1 obtained after shrinking the simple blossom defined by the subset of nodes {5, 4, 9, 10, 8, 7, 6} in G into the pseudonode B. $\mathcal{P}^1$ = 1, (1, 2), 2, (2, 3), 3, (3, B), B, (B, 11), 11, (11, 12), 12 is an augmenting path in G^1. By expanding the pseudonode B on it as described in Case 2 above, we obtain the augmenting path 1, (1, 2), 2, (2, 3), 3, (3, 4), 4, (4, 9), (9, 10), 10, (10, 11), 11, (11, 12), 12 in G.

THEOREM 10.2 *Let* $\mathbf{M}$ *be a matching in* $\mathrm{G} = (\mathcal{N}, \mathcal{A})$*. Let* $B = (\mathcal{N}_B, \mathcal{A}_B)$ *be a simple blossom in* G wrt $\mathbf{M}$*, which is shrunk into the pseudonode* B*, resulting in the current network* G^1 *with the current matching* $\mathbf{M}^1$*.*

(i) *If there exists an augmenting path in* G^1 wrt $\mathbf{M}^1$*, then there exists an augmenting path in* G wrt $\mathbf{M}$*.*

(ii) *If* B *satisfies (10.14), and if there exists an augmenting path in* G wrt $\mathbf{M}$*, then there exists an augmenting path in* G^1 wrt $\mathbf{M}^1$*.*

Proof (i) follows from the expansion procedure discussed above. To prove (ii), let $\mathcal{P}$ be an augmenting path in G wrt $\mathbf{M}$. Let t be the base node of B. If $\mathcal{P}$ contains no nodes from $\mathcal{N}_B$, $\mathcal{P}$ itself is an augmenting path in G^1, we are done.

Now suppose $\mathcal{P}$ contains one or more nodes from $\mathcal{N}_B$. We consider two cases.

Case 1 One of the End Nodes of $\mathcal{P}$ is in $\mathcal{N}_B$ Since $\mathcal{P}$ is an augmenting path, its end nodes are unmatched. So, B must be an unmatched node in G^1, and t must be an end node of $\mathcal{P}$. The path $\mathcal{P}$ may go in and out of the simple blossom B several times, but let i be the successor of the last node in $\mathcal{N}_B$ on $\mathcal{P}$ as you travel along it beginning at t. Replace the entire portion of the path $\mathcal{P}$ from t to i by the single edge $(B; i)$, this converts $\mathcal{P}$ into an augmenting path $\mathcal{P}^1$ wrt $\mathbf{M}^1$ in G^1.

Case 2 The Only Nodes from $\mathcal{N}_B$ on $\mathcal{P}$ are Intermediate Nodes Let p_1, p_2 be the end nodes of $\mathcal{P}$. As you travel from p_1 to p_2 along $\mathcal{P}$, let q_1, q_2 be the first and last nodes of $\mathcal{N}_B$ encountered, with $(s_1; q_1)$ as the edge leading to q_1, and $(q_2; s_2)$ as the edge leading out of q_2.

If t is unmatched, replace the entire portion of the path $\mathcal{P}$ from s_1 to p_2 by the single edge $(s_1; B)$ and let the resulting path be called $\mathcal{P}^1$.

If t is a matched node and one of the two edges $(s_1; q_1), (q_2; s_2)$ is a matching edge, say $(s_1; q_1)$, replace the portion of the path $\mathcal{P}$ from s_1 to s_2 with the pair of edges $(s_1; B), (B; s_2)$ and let the resulting path be called $\mathcal{P}^1$.

Suppose t is a matched node and both the edges $(s_1; q_1), (q_2; s_2)$ are nonmatching edges. By hypothesis, B satisfies (10.14). Let $(t; i_1)$ be the matching edge incident at t. Define the following paths in G.

$\mathcal{P}_1$ = portion of the path from p_1 to s_1 on $\mathcal{P}$

$\mathcal{P}_2$ = portion of the path from s_2 to p_2 on $\mathcal{P}$

$\mathcal{P}_3$ = stem of B, beginning with $(t; i_1)$, to an unmatched node , i_u, say.

$\mathcal{P}_4$ = AP from i_1 to i_u obtained by deleting $(t; i_1)$ from $\mathcal{P}_3$

If $\mathcal{P}_1$ and $\mathcal{P}_3$ have no common nodes, let $\mathcal{P}^1$ be the path $\mathcal{P}_1 \cup \{(s_1; B), (B; i_1)\} \cup \mathcal{P}_4$.

If $\mathcal{P}_1$ and $\mathcal{P}_3$ have common nodes, but $\mathcal{P}_2$ and $\mathcal{P}_3$ have no common nodes, let $\mathcal{P}^1$ be the path ($\mathcal{P}_4$ in reverse direction from i_u to i_1) $\cup\{(i_1; B), (B; s_2)\} \cup \mathcal{P}_2$.

Now consider the case where both $\mathcal{P}_1$ and $\mathcal{P}_2$ have nodes in common with the stem $\mathcal{P}_3$. As you travel on $\mathcal{P}_2$ in reverse direction (i.e., from p_2 to s_2) let $\tilde{i}$ be the first node encountered on $\mathcal{P}_3$. Likewise let $\hat{i}$ be the first node encountered on $\mathcal{P}_3$ as you travel on $\mathcal{P}_1$ from p_1 to s_1. Since $\mathcal{P}_1$ and $\mathcal{P}_2$ are node disjoint (because $\mathcal{P}$ is a simple path) $\tilde{i} \neq \hat{i}$. As you travel from i_u to t along $\mathcal{P}_3$ suppose $\tilde{i}$ comes after $\hat{i}$ (the case where $\hat{i}$ comes after $\tilde{i}$ is handled in a symmetric way). Define

$\mathcal{P}_5$ = portion of the path from $\tilde{i}$ to p_2 on $\mathcal{P}_2$

$\mathcal{P}_6$ = portion of the path from i_u to $\tilde{i}$ on $\mathcal{P}_3$

$\mathcal{P}_7$ = portion of the path from $\tilde{i}$ to i_1 on $\mathcal{P}_3$

Verify that $\mathcal{P}_5$ and $\mathcal{P}_3$ are disjoint and $\mathcal{P}_7$ is disjoint with both $\mathcal{P}_1$ and $\mathcal{P}_2$. Also the edge on $\mathcal{P}_5$ incident at $\tilde{i}$ will be a nonmatching edge. If the edge incident at $\tilde{i}$ on $\mathcal{P}_6$ is a matching edge define $\mathcal{P}^1$ to be $\mathcal{P}_5 \cup \mathcal{P}_6$. If the edge incident at $\tilde{i}$ on $\mathcal{P}_6$ is a nonmatching edge define $\mathcal{P}^1$ to be $\mathcal{P}_1 \cup \{(s_1; B), (B; i_1)\} \cup \mathcal{P}_7 \cup \mathcal{P}_5$.

It can be verified that $\mathcal{P}^1$, so constructed in each case discussed above, is an augmenting path in G^1. This completes the proof of (ii).
∎

The result in part (ii) of Theorem 10.2 depends critically on the hypothesis that the simple blossom B satisfies (10.14). If B does not

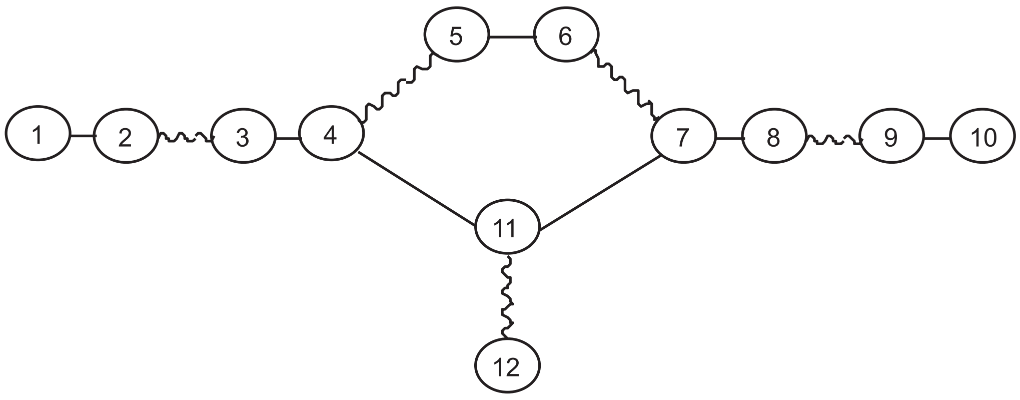

Figure 10.13: Network G^1. $\mathbf{M}^1$ is the wavy matching, it has maximum cardinality in G^1.

satisfy (10.14), there may not exist an augmenting path in G^1 wrt $\mathbf{M}^1$, even though there exists an augmenting path in G wrt $\mathbf{M}$. Consider the network G in Figure 10.13 with the wavy matching $\mathbf{M}$. The partial network determined by the subset of nodes {4, 5, 6, 7, 11} is a simple blossom in G satisfying (10.13) but not (10.14). Shrinking this simple blossom into the pseudonode B leads to the current network G^1 in Figure 10.14 with the wavy matching $\mathbf{M}^1$. The path on the top of G is an augmenting path wrt $\mathbf{M}$. It can be verified that $\mathbf{M}^1$ is a maximum cardinality matching in G^1, and hence no augmenting path exists in G^1 wrt $\mathbf{M}^1$.

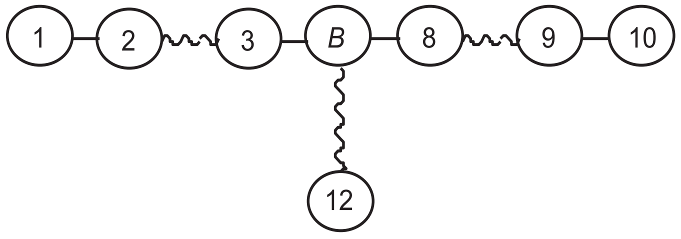

Figure 10.14: Network G^1. $\mathbf{M}^1$ is the wavy matching, it has maximum cardinality in G^1.

All simple blossoms that we encounter in the blossom algorithm for the maximum cardinality matching problem discussed later will always satisfy (10.14). However, in the blossom algorithm for the minimum cost matching problem, we sometimes encounter simple blossoms not satisfying (10.14), that algorithm has special routines for handling such simple blossoms.

Nodes and edges in the current network G^1 are called **current nodes, current edges** respectively. A current node may either be an original node or a pseudonode. As mentioned earlier, G^1 now has a rooted alternating tree wrt the current matching $\mathbf{M}^1$. Continue growing the alternating tree in G^1 by resuming labeling, using the same scheme discussed earlier.

Simple Blossoms in the Current Network, Blossoms

When labeling is resumed in G^1, a new simple blossom, say B_2, satisfying (10.13) wrt the current matching $\mathbf{M}^1$ in it, may be identified. The signal for this is the same as before, i.e., when two nodes on a current matching (nonmatching) edge are labeled as inner (outer) nodes. In this case, the alternating tree being grown is said to have blossomed again. B_2 is identified exactly as before, its base node is the first common node on the predecessor paths of the identifying pair of nodes; and the odd alternating cycle in it consists of the current edge joining the identifying pair of nodes, and the portions of their predecessor paths up to the base node. If this simple blossom does not contain the pseudonode formed earlier, say B_1, it is itself a simple blossom in G wrt $\mathbf{M}$. Otherwise, B_2 is a simple blossom in the current network containing the pseudonode B_1 as a node on it.

Let $\mathcal{N}_{B_2}$ be the set of original nodes which are either current nodes in B_2, or those contained within a pseudonode which is a current node in B_2. Let $\mathcal{A}_{B_2} = \{(i;j) : (i;j) \in \mathcal{A}, i, j \in \mathcal{N}_{B_2}\}$. Then the partial network $(\mathcal{N}_{B_2}, \mathcal{A}_{B_2})$ is called a **blossom** in the original network G wrt $\mathbf{M}$, and B_2 in the current network G^1 is the simple blossom corresponding to it.

In general, a blossom in G wrt the matching $\mathbf{M}$ is a partial network $B = (\mathcal{N}_B, \mathcal{A}_B)$ satisfying the following properties.

$|\mathcal{N}_B|$ is an odd number $\geqq 3$

$\mathbf{M}_B = \mathbf{M} \cap \mathcal{A}_B$ is a maximum cardinality matching in B that leaves exactly one node unmatched. This unique unmatched node, t say, is called the **apex** of B. Notice that t may be a matched node in G, with the matching edge incident at it not in $\mathcal{A}_B$. (10.15)

There is a simple blossom corresponding to B in the current network at the stage that B is discovered. Some of the nodes on this simple blossom may be pseudonodes formed in earlier stages. It satisfies (10.13) in the current network and current matching at the stage that it is discovered. $\mathcal{N}_B$ is the set of original nodes which are either nodes on this simple blossom, or contained within pseudonodes on it.

Whenever a new blossom is discovered, the simple blossom corresponding to it is stored by storing all the current nodes on it together with the labels on them, its identifying pair of nodes and its base node; and it is then shrunk into a new pseudonode, p say, exactly as before. Then p gets the same label as on the base node on the simple blossom, and the predecessor index of any immediate successor of that base node is changed into p. The new pseudonode p and its descendents are put in the list of labeled and unscanned nodes, and the current alternating tree is grown again by resuming labeling. In matching algorithms, blossoms may be found and shrunk repeatedly.

The **base** of a blossom B is defined to be the base node of the simple blossom corresponding to it. If the base is an original node, it is the same as the apex of the blossom. If the base is a pseudonode, the apex of the blossom is the apex of its base node. Blossoms B discovered in the maximum cardinality matching algorithm also satisfy the following additional property.

> The simple blossom corresponding to it is either a rooted simple blossom (i.e., the base is the root of the alternating tree in the current network when it is discovered), or there exists an alternating path beginning with the current matching edge incident at the base, to an unmatched node in the current network, such that none of the nodes on this path other than the base are in the simple blossom. This alternating path is the stem of this simple blossom. (10.16)

As an example consider the network in Figure 10.15 with the wavy matching in it. An alternating tree is rooted at the unmatched node 1 and grown by scanning the nodes in serial order. We show the detection and shrinking of two simple blossoms, at the end of which we have the current network shown in Figure 10.17.

Let $\mathrm{G}^1 = (\mathcal{N}^1, \mathcal{A}^1)$ be the current network at some stage after some blossoms have been shrunk. Nodes in $\mathcal{N}^1$ are called **current nodes**, they may be either pseudonodes, or original nodes in $\mathcal{N}$ not contained in any blossoms shrunk so far. Let $p \in \mathcal{N}^1$ be a current node. We will say that an original node i is **inside** p *or that* p **contains** i **inside it**, if either $i = p$, or if i is a node in the blossom corresponding to p. An original edge $(i; j) \in \mathcal{A}$ is said to be inside p if both i and j are inside p. Another pseudonode q which is not a current node is said to be inside p if the set of original nodes inside q is a subset of the set of original nodes inside p.

When a pseudonode is just formed, it is a current node in the current network at that stage. Afterwards, it might get absorbed inside another pseudonode, at which stage it is said to have become **dormant**. We will say that a pseudonode is a **current pseudonode** *or* **outermost pseudonode** if it is a current node at that stage, or a **dormant pseudonode** if it is contained inside another pseudonode.

Levels of Pseudonodes and Blossoms, and Their Nesting Properties

Original nodes refer to nodes in $\mathcal{N}$ in the original network G. They are defined to be **level 0 nodes**. For the sake of consistency, the base node and the apex node corresponding to a level 0 node is defined to be that node itself. A simple blossom, all the nodes on which are level 0 nodes is said to be a **level 1 simple blossom**, and the pseudonode into which it is shrunk is called a **level 1 pseudonode**. In general, for any $s \geqq 1$, a **level** $(s+1)$ **simple blossom** is one, at least one node on which is a level s pseudonode, and all other nodes on which are either pseudonodes of level $\leqq s$, or original nodes. For any $s \geqq 1$, the blossom (pseudonode) corresponding to a level s simple blossom is called a level s blossom (pseudonode). Outermost pseudonodes may be of any level, and at each stage there may be outermost pseudonodes of various levels in the current network. We will use the term **labeled node** to always mean a current node in the current alternating tree labeled as an outer or inner node, which may be either an original node not contained in any pseudonode, or an outermost pseudonode.

The base node of a simple blossom, pseudonode, or blossom, is the node on the corresponding simple blossom that is not contained on any matching edge within that simple blossom. Its apex node is the original node in the corresponding blossom, that is not contained on any matching edge inside that blossom. So, for any simple blossom, pseudonode, or blossom, the apex node is the same as the apex of its base node.

After some blossom shrinkings, let G^1 denote the current network at some stage. Let u be the total number of blossoms that have been shrunk into pseudonodes up to this stage. Let these blossoms, or the pseudonodes corresponding to them, be $B_1, \ldots, B_u$. For $v = 1$ to u let $\mathcal{N}_{B_v}$ be the set of original nodes contained within B_v. We will now discuss some of the properties of the blossoms at this stage.

THEOREM 10.3 *Each shrinking operation leads to a new current network in which the number of current nodes is reduced by an even number* $\geqq 2$.

Proof Each simple blossom consists of an odd number of current nodes $\geqq 3$. When it is shrunk, all the nodes in the simple blossom are eliminated and just one new pseudonode introduced. This leads to the result stated in the theorem. ∎

THEOREM 10.4 *The class of subsets $\{\mathcal{N}_{B_1}, \ldots, \mathcal{N}_{B_u}\}$ has the property that for v, w, if $\mathcal{N}_{B_v} \cap \mathcal{N}_{B_w} \neq \emptyset$, then one of them is a subset of the other.*

Proof From the manner in which blossoms are identified and shrunk, it is clear that if $\mathcal{N}_{B_v} \cap \mathcal{N}_{B_w} \neq \emptyset$, one of B_v or B_w is contained within a pseudonode on the simple blossom corresponding to the other. The theorem follows from this. ∎

A class of subsets satisfying the property discussed in Theorem 10.4 is known as a **nested class of subsets**. Thus the class of subsets of original nodes contained within the pseudonodes at any stage will always be a nested class of subsets.

THEOREM 10.5 *If $\mathcal{N}_{B_v} \supset \mathcal{N}_{B_w}$ then $|\mathcal{N}_{B_v}| \geqq 2 + |\mathcal{N}_{B_w}|$.*

Proof Since $\mathcal{N}_{B_v} \supset \mathcal{N}_{B_w}$, B_w must be contained within a pseudonode on the simple blossom corresponding to B_v. This simple blossom must have at least two other nodes besides the pseudonode containing B_w. These facts imply the theorem. ∎

THEOREM 10.6 *The total number of pseudonodes of all levels, which are either current nodes, or contained within other outermost pseudonodes is always $\leqq (n/2)$.*

Proof This theorem can be proved very easily and directly from Theorem 10.3. However, we provide here a somewhat lengthy proof giving information on bounds for the number of pseudonodes of various levels. We consider all the pseudonodes at this stage in the current network G^1, let r_t be the number of pseudonodes of level $t = 1, 2, \ldots$ among these. So, we have to prove that $\sum_t r_t \leqq (n/2)$. Each level 1 pseudonode is obtained by shrinking a simple blossom in G, which contains at least 3 nodes. So $r_1 \leqq (n/3)$. Let $\mathbf{S}_1$ denote the set of all level 1 pseudonodes, and original nodes of G which are not contained in any level 1 pseudonode. Nodes in $\mathbf{S}_1$ are the nodes on the simple blossom corresponding to any level 2 pseudonode. We have $|\mathbf{S}_1| \leqq r_1 + n - 3r_1 = n - 2r_1$. Each simple blossom corrresponding to a level 2 pseudonode consists of at least 3 nodes from $\mathbf{S}_1$, so $r_2 \leqq (n - 2r_1)/3$. So, $r_1 + r_2 \leqq ((n - 2r_1)/3) + r_1 = (n/3) + (r_1/3) \leqq (n/3) + (n/3^{\mathbf{2}})$.

In a similar manner, for $t \geqq 2$, define $\mathbf{S}_t$ to be the set of all pseudonodes of level t, and all original nodes and pseudonodes of levels $\leqq (t-1)$ which are not contained within any pseudonode of level $\leqq t$. The nodes in the simple blossom corresponding to any level $(t+1)$ pseudonode are those from the set $\mathbf{S}_t$, each of these simple blossoms contains at least three nodes from $\mathbf{S}_t$. Using the same arguments as above, and induction, it can be seen that $|\mathbf{S}_t| \leqq r_t + |\mathbf{S}_{t-1}| - 3r_t \leqq n - 2r_1 - 2r_2 - \ldots - 2r_t$. So, $r_{t+1} \leqq (n - 2r_1 - \ldots - 2r_t)/3$. So, $r_{t+1} + r_t + \ldots + r_1 \leqq ((n - 2r_1 - \ldots - 2r_t)/3) + r_t + \ldots + r_1 = (n/3) + (r_1 + \ldots + r_t)/3 \leqq (n/3) + (n/3^{\mathbf{2}}) + \ldots + (n/3^{\mathbf{t+1}})$, for any $t \geqq 2$. Hence

$$\begin{aligned}\sum_t r_t &\leqq \sum_{t=1}^{\infty} \frac{n}{3^{\mathbf{t}}} = \frac{n}{3}(1 + \frac{1}{3} + \frac{1}{3^{\mathbf{2}}} + \ldots) \\ &= \frac{n}{3}(1 - \frac{1}{3})^{-1} = \frac{n}{3} \times \frac{3}{2} = \frac{n}{2}\end{aligned}$$

This proves the theorem. ∎

THEOREM 10.7 *The number of unmatched nodes in the current network* G^1 wrt *the current matching* $\mathbf{M}^1$ *is always the same as the number of unmatched nodes in the original network* G wrt *the matching* $\mathbf{M}$ *in it.*

Proof Consider the operation of shrinking one simple blossom into a pseudo- node. All the nodes on the simple blossom, with the possible exception of its base node, are matched nodes. If the base node is a matched (the unmatched root) node, the pseudonode into which this simple blossom is shrunk will be a matched (the new unmatched root) node after it is formed. Thus each shrinking operation leaves the total number of unmatched nodes unchanged, which implies the theorem. ∎

THEOREM 10.8 *Let B be a blossom in* G wrt *the matching* $\mathbf{M}$. *Let t be the apex node of B, and $j \neq t$ an original node inside B. There exist two edge disjoint APs in B from j to t, one beginning with the matching edge incident at j, and the other beginning with a nonmatching edge incident at j.*

Proof Let B_1 be the simple blossom corresponding to the blossom B. Suppose the base node of B_1 is q, and let p be the node in B_1 that contains j inside it. First suppose $p \neq q$. Since B_1 is a simple blossom, there exist two edge disjoint APs from p to q on its alternating cycle, one beginning with the matching edge incident at p on it, and the other beginning with the nonmatching edge incident at p on it. Let these paths be $\mathcal{P}_1, \mathcal{P}_2$ respectively. Expand the pseudonodes on these paths, always expanding those of the highest level among the remaining pseudonodes first, using the procedure described earlier. This leads eventually to the alternating paths from j to t in the statement of the theorem. If $p = q$, go into the pseudonode q and repeat the same argument there. ∎

As an example consider the simple blossom in Figure 10.16 corresponding to the pseudonode B_2 in the current network in Figure 10.17, or the blossom in the original network in Figure 10.15 defined by the subset of original nodes {4, 5, 8, 9, 7, 12, 11, 10, 6} corresponding to B_2. The base node of B_2 is B_1, and its apex node is 4. 11 is an original node inside B_2. 11 is a node on this simple blossom and the AP in it beginning with the matching edge incident at 11 to the base B_1 is 11, (11, 12), 12, (12, B_1), B_1. We now expand the pseudonode B_1 on this path. The simple blossom corresponding to B_1 can be seen from Figure 10.15, node 12 is connected only to node 5 in this simple blossom by a nonmatching edge. So, expanding the pseudonode B_1 on the above path leads to the AP 11, (11, 12), 12, (12, 5), 5, (5, 8), 8, (8, 9), 9, (9, 7), 7, (7, 4), 4 in the original network in Figure 10.15, from 11 to 4.

THEOREM 10.9 *Let $B = (\mathcal{N}_B, \mathcal{A}_B)$ be a blossom in* G wrt *the matching* $\mathbf{M}$, *with t as its apex node. If B satisfies (10.16), either t is unmatched in* $\mathbf{M}$, *or t is a matched node and there exists an alternating path* wrt $\mathbf{M}$ *from t beginning with the matching edge incident at* *it, to an unmatched node such that none of the edges on this path are from* $\mathcal{A}_B$.

Proof If the stem of the simple blossom corresponding to B is empty, t is the unmatched root node, otherwise t is a matched node and the required path is obtained by expanding this stem. ∎

THEOREM 10.10 *(i) If there exists an augmenting path in the current network* G^1 wrt *the current matching* $\mathbf{M}^1$, *then there exists an augmenting path in the original network* G wrt *the matching* $\mathbf{M}$ *in it. (ii) If all the pseudonodes in* G^1 *correspond to blossoms which satisfy (10.16), and if an augmenting path exists in* G wrt $\mathbf{M}$, *then there exists an augmenting path in* G^1 wrt $\mathbf{M}^1$.

Proof The current network and the current matching in it change after each blossom shrinking. The result here follows by applying the results in Theorems 10.2 and 10.9 after each successive blossom shrinking. ∎

Once an alternating tree is rooted at an unmatched node r, another unmatched node can only join it as an inner labeled node, and if that happens the tree is said to have become an *augmenting tree* . If the unmatched node labeled is i, its predecessor path $\mathcal{P}^1$ in the current network is an augmenting path. The corresponding augmenting path $\mathcal{P}$ in G between the apex of i and r is obtained by expanding the pseudonodes on $\mathcal{P}^1$, always expanding the highest level pseudonode, one at a time. Hence pseudonodes are expanded in reverse order to the one in which they are formed.

By Theorem 10.6, if there exists an augmenting path in G wrt the matching $\mathbf{M}$, containing the root node r, then the alternating tree grown will become an augmenting tree after at most $(n/2)$ blossom shrinking operations. If an augmenting path is discovered, a rematching operation is carried out. If there are still some unmatched nodes, the whole procedure can be repeated with the new matching.

As an example, consider the alternating tree rooted at the unmatched node 1 and grown in the network G in Figure 10.15 with the wavy matching $\mathbf{M}$. We adopt the rule of selecting the node with

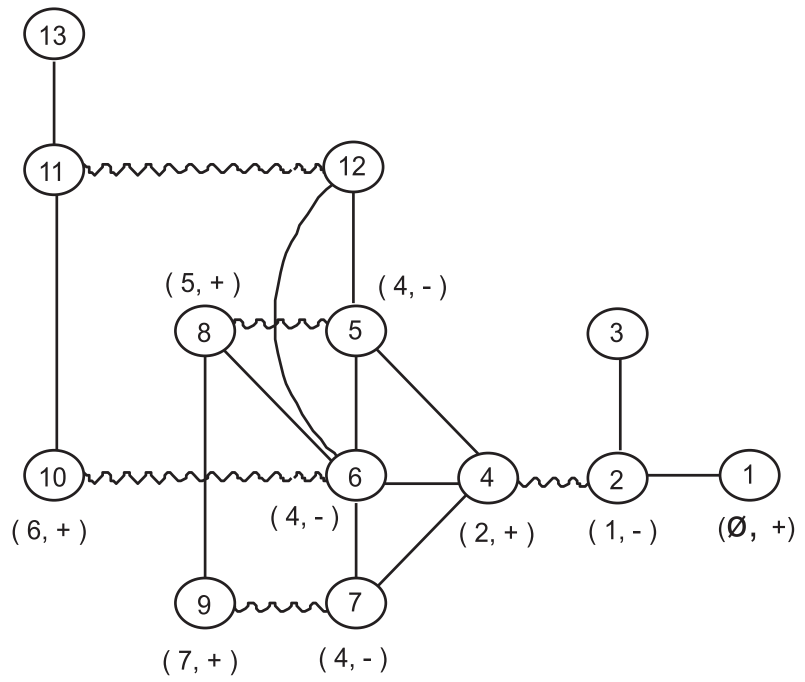

Figure 10.15: Network G. **M** is the wavy matching. Labels are recorded by the side of the nodes.

the least serial number among those in the list for scanning. Nodes 8, 9 are the identifying pair for the simple blossom defined by the subset of nodes {9, 8, 5, 4, 7}, which is shrunk into the pseudonode B_1 resulting in the current network in Figure 10.16. Labeling is continued after shrinking B_1.

Nodes 11, 12 are the identifying pair of nodes for the simple blossom defined by the subset of nodes {11, 12, B_1, 6, 10} which is shrunk into the pseudonode B_2 resulting in the current network in Figure 10.17. When tree growth is resumed, the unmatched node 13 is labeled, and the alternating tree in Figure 10.17 has become an augmenting tree with the horizontal path between nodes 1 and 13 as an augmenting path. We derive the corresponding augmenting path in the original network by expanding the pseudonode on it, and rematch. This leads

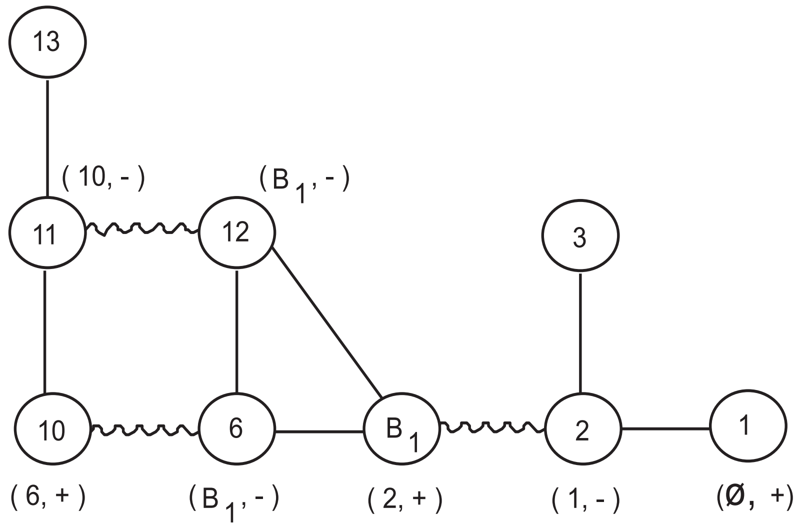

Figure 10.16: Labeling is continued after shrinking B_1.

to the new matching marked with thick edges in Figure 10.18. A new alternating tree is rooted at the unmatched node 3 and grown. After nodes 3, 2, 1 are labeled, we are unable to label any more, even though the tree has not become an augmenting tree, and has not blossomed. We discuss the implications of this next (in this example, the original network has 13 nodes, and the thick matching in Figure 10.18 has 6 edges, so clearly this matching is a maximum cardinality matching).

Hungarian Trees and Their Properties

At some stage it may happen that the current alternating tree has not become an augmenting tree, it has no blossoms to be shrunk, and it cannot grow any further (i.e., labeling cannot be continued any further). At this stage we say that the alternating tree has become **hungarian**, and the tree itself is called a **Hungarian tree**. By the above results, this can only happen if there exists no augmenting path wrt the present matching **M**, beginning with the unmatched root node r in G. Clearly the only unmatched node in a Hungarian tree is its root node.

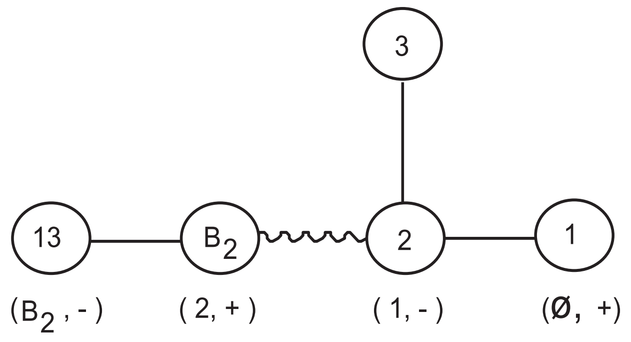

Figure 10.17: Labeling continued. Unmatched node 13 is labeled.

An example of a Hungarian tree is the tree consisting of nodes 3, 2, 1 rooted at the unmatched node in Figure 10.18.

THEOREM 10.11 *A Hungarian tree contains an odd number of current nodes, all but the root node in which are matched nodes contained on in-tree current matching edges. The total number of original nodes contained within the current nodes in a Hungarian tree is odd, and all but the apex of the root node among these is a matched node.*

Proof Since a Hungarian tree has not become an augmenting tree, all nonroot nodes in it must be matched nodes lying on current in-tree matching edges. This also implies that the number of current nodes in the Hungarian tree is one plus twice the number of current matching edges in it, which is an odd number. Since every current node contains an odd number of original nodes inside it, the total number of original nodes inside current nodes in this Hungarian tree is also an odd number. As all the current nodes in the Hungarian tree other than the root node are matched, all the original nodes inside current nodes in it are matched except the apex of the root node. ∎

Let $\mathcal{N}_H$ be the set of original nodes contained within nodes on a Hungarian tree H in the current network. Let $G_H = (\mathcal{N}_H, \mathcal{A}_H)$ be the partial network of the original network determined by $\mathcal{N}_H$, and $\mathbf{M}_H$ the set of edges in the present matching $\mathbf{M}$ in G contained within G_H. Theorem 10.11 implies that all the nodes in $\mathcal{N}_H$ excepting the apex of

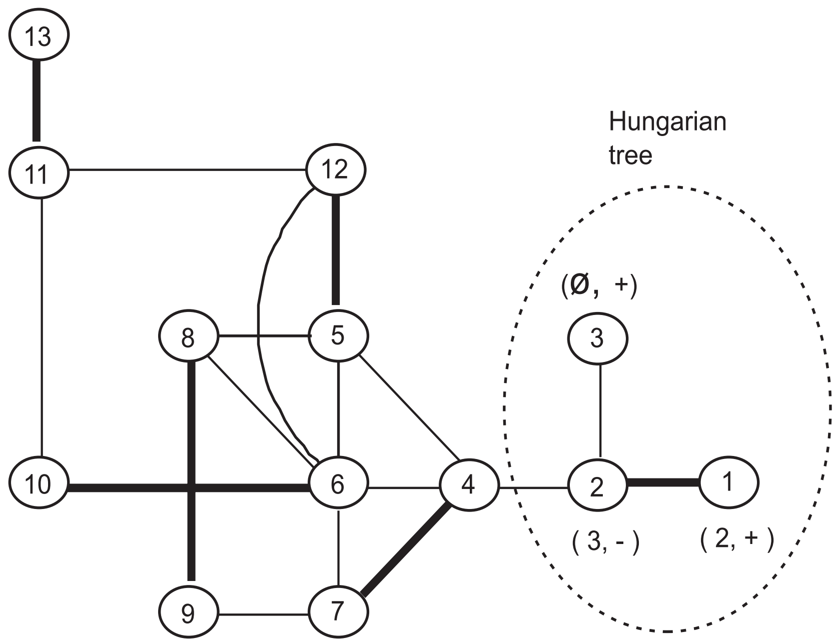

Figure 10.18: New matching edges are thick. New alternating tree rooted at 3, and grown.

the root node in the Hungarian tree H, are matched by the matching edges in $\mathbf{M}_H$. So, $|\mathcal{N}_H|$ is odd, and $\mathbf{M}_H$ is a maximum cardinality matching in G_H.

THEOREM 10.12 *All leaf nodes in a Hungarian tree must be outer nodes.*

Proof Let $j \neq$ root be a leaf node in a Hungarian tree. If j is an inner node, and it is unmatched, the tree is an augmenting tree contrary to the hypothesis that it is hungarian. If j is matched, let its mate be p. If p is unlabeled, it can be labeled, a contradiction. If p is already labeled, it is not the root since it is matched, and it cannot be an outer node because for every nonroot outer node its predecessor must be its mate. So, if p is an outer node, j must be its predecessor, contradiction to the fact that j is a leaf node. Hence p must be an inner node. But since $(j;p)$ is a current matching edge, if j, p are both inner nodes, they are the identifying pair for a blossom that can be shrunk, contradiction to the hungarianness of the tree. Hence j must be an outer node. ∎

THEOREM 10.13 *If $(i;j)$ is a current edge with i in a Hungarian tree and j not in it, then i must be an inner node.*

Proof On the contrary, suppose i is an outer node. If $(i;j)$ is a nonmatching edge, then j could have been labeled, contradicting the fact that the tree has become hungarian. If $(i;j)$ is a matching edge, then j must be the predecessor of i, contradicting the fact that j is not an in-tree node. So, i must be an inner node. ∎

COROLLARY 10.1 *Suppose the current network contains a Hungarian tree. If an edge in this network is incident at node i which is an outer node in the Hungarian tree, then that edge must contain an inner node in this Hungarian tree at its other end.*

Proof Follows directly from Theorem 10.13. ∎

THEOREM 10.14 $\mathrm{G} = (\mathcal{N}, \mathcal{A})$ *is the original network with the matching* $\mathbf{M}$ *in it. An alternating tree is rooted at an unmatched node r and grown. It becomes the Hungarian tree* H *at the stage when* $\mathrm{G}^1 = (\mathcal{N}^1, \mathcal{A}^1)$ *is the current network.* $\mathcal{N}_H$ *is the set of original nodes contained within current nodes on* H*, and* $\mathcal{N}_{\overline{H}} = \mathcal{N} \backslash \mathcal{N}_H$*.* $\mathrm{G}_H = (\mathcal{N}_H, \mathcal{A}_H)$, $\mathrm{G}_{\overline{H}} = (\mathcal{N}_{\overline{H}}, \mathcal{A}_{\overline{H}})$ *are the partial networks of* G *determined by* $\mathcal{N}_H, \mathcal{N}_{\overline{H}}$ *respectively.* $\mathbf{M}_H = \mathbf{M} \cap \mathcal{A}_H$*, and* $\overline{\mathbf{M}}_{\overline{H}}$ *is a maximum cardinality matching in* $\mathrm{G}_{\overline{H}}$*. Then* $\mathbf{M}_H \cup \overline{\mathbf{M}}_{\overline{H}}$ *is a maximum cardinality matching in* G.

Proof Let $\mathbf{M}^1$ be the current matching in G^1. Let $\mathbf{A} = \{(i;j) : i \in \mathcal{N}_H, j \in \mathcal{N}_{\overline{H}}, (i;j) \in \mathcal{A}\}$. Since all nodes in $\mathcal{N}_H$ other than r are matched by matching edges within $\mathcal{A}_H$, all edges in $\mathbf{A}$ are nonmatching edges, so $\mathbf{A} \cap \mathbf{M} = \emptyset$. Let $\hat{\mathcal{A}} = \{(i;j) : (i;j) \in \mathcal{A}$, at least one of i or j is in $\mathcal{N}_H\} = \mathcal{A}_H \cup \mathbf{A}$. So, $\hat{\mathbf{M}} = \mathbf{M} \cap \hat{\mathcal{A}} = \mathbf{M} \cap \mathcal{A}_H = \mathbf{M}_H$. Let $\hat{\mathcal{N}}$ be the set of nodes on edges in $\hat{\mathcal{A}}$, and $\hat{\mathrm{G}} = (\hat{\mathcal{N}}, \hat{\mathcal{A}})$. We will now prove that $\mathbf{M}_H$ is a maximum cardinality matching in $\hat{\mathrm{G}}$. Define

$$\begin{aligned}
\mathcal{N}_H^1 &= \text{ Set of all current nodes in H} \\
\hat{\mathcal{N}}^1 &= \mathcal{N}_H^1 \cup (\hat{\mathcal{N}} \backslash \mathcal{N}_H) = \text{ Set of all current nodes} \\
&\ \text{ corresponding to original nodes in } \hat{\mathcal{N}} \\
\hat{\mathcal{A}}^1 &= \text{ Set of current edges with at least one node in } \mathcal{N}_H^1 \\
\hat{\mathrm{G}}^1 &= (\hat{\mathcal{N}}^1, \hat{\mathcal{A}}^1) = \text{ Current network corresponding to } \hat{\mathrm{G}} \\
\hat{\mathbf{M}}^1 &= \mathbf{M}^1 \cap \hat{\mathcal{A}}^1 \\
\mathbf{A}^1 &= \{(p;q) : p \in \mathcal{N}_H^1, q \in \hat{\mathcal{N}} \backslash \mathcal{N}_H, (p;q) \in \mathcal{A}^1\}
\end{aligned}$$

By previous arguments, $\mathbf{A}^1 \cap \hat{\mathbf{M}}^1 = \emptyset$ and the only nodes in $\hat{\mathrm{G}}^1$ which are unmatched in $\mathbf{M}^1$ are those in $\hat{\mathcal{N}} \backslash \mathcal{N}_H$, and the root node of H. So any augmenting path wrt $\mathbf{M}^1$ in $\hat{\mathrm{G}}^1$ must contain at least one node in $\hat{\mathcal{N}} \backslash \mathcal{N}_H$ as a terminal node.

Suppose $\mathcal{P}$ is an augmenting path in $\hat{\mathrm{G}}^1$ wrt $\hat{\mathbf{M}}^1$ beginning with an unmatched node $p \in \hat{\mathcal{N}} \backslash \mathcal{N}_H$, to another unmatched node $q \in \hat{\mathcal{N}}^1$. Give the nodes on this path alternately *outer and inner designations* (the reader is cautioned not to confuse these outer and inner designations with the outer and inner labels already existing on nodes in H) beginning with an outer designation for p. If $\mathcal{P}$ contains any nodes from $\mathcal{N}_H^1$, the fact that all the edges in $\mathbf{A}^1$ are nonmatching edges, and Theorem 10.13 together imply that any node from $\mathcal{N}_H^1$ in $\mathcal{P}$ bears an inner or outer designation iff it is an inner or outer labeled node respectively in H. Since $\mathcal{P}$ is an augmenting path, its terminal node q must bear an inner designation, and since all nodes labeled as inner nodes in H are matched nodes, q cannot be a node in H, and hence q must also be a node in $\hat{\mathcal{N}} \backslash \mathcal{N}_H$.

Hence an alternating path in $\hat{\mathrm{G}}^1$, starting from an unmatched node in $\hat{\mathcal{N}}^1$ outside the Hungarian tree H, can only be an augmenting path if it terminates at another node outside H. But an augmenting path always consists of an odd number of edges. So, if the first edge in an augmenting path in $\hat{\mathrm{G}}^1$ is a current edge joining a node in $\hat{\mathcal{N}} \backslash \mathcal{N}_H$ to a current inner labeled node in H, then the last edge on this path must be a current edge joining an outer labeled node in H with an unmatched node. This is impossible by Theorem 10.13. Hence there exists no augmenting path wrt the current matching $\hat{\mathbf{M}}^1$ in $\hat{\mathrm{G}}^1$. By Theorem 10.10, this implies that there exists no augmenting path in $\hat{\mathrm{G}}$ wrt $\hat{\mathbf{M}}$. Hence $\hat{\mathbf{M}} = \mathbf{M}_H$ is a maximum cardinality matching in $\hat{\mathrm{G}}$.

Now let $\mathbf{N}$ be any other matching in G. Let $\hat{\mathbf{N}} = \mathbf{N} \cap \hat{\mathcal{A}}$, and $\mathbf{N}_{\overline{H}} = \{(i;j) : (i;j) \in \mathbf{N}$, and i and j are both in $\mathcal{N}_{\overline{H}}\}$. Then $\mathbf{N} = \hat{\mathbf{N}} \cup \mathbf{N}_{\overline{H}}$. By definition, $|\overline{\mathbf{M}}_{\overline{H}}| \geqq |\mathbf{N}_{\overline{H}}|$. Since $\hat{\mathbf{M}} = \mathbf{M}_H$ is a maximum cardinality matching in $\hat{\mathrm{G}}$, we have $|\mathbf{M}_H| \geqq |\hat{\mathbf{N}}|$. These facts imply that $|\mathbf{M}_H \cup \overline{\mathbf{M}}_{\overline{H}}| \geqq |\mathbf{N}|$, so $\mathbf{M}_H \cup \overline{\mathbf{M}}_{\overline{H}}$ is a maximum cardinality matching in G. ∎

So, to find a maximum cardinality matching in G, start with an arbitrary matching, root an alternating tree at an unmatched node and grow it. If the tree blossoms, shrink the blossom. If the tree becomes augmenting, trace the augmenting path, rematch using it, and repeat the procedure with the new matching. If the tree becomes hungarian, identify the set $\mathcal{N}_H$ of all the original nodes contained within that Hungarian tree. Let $\mathbf{M}_H$ be the set of all matching edges in the present matching, both of whose incident nodes are in $\mathcal{N}_H$. Delete all the nodes in $\mathcal{N}_H$ and all the edges which contain at least one node from $\mathcal{N}_H$, let the remaining network be $\mathrm{G}_{\overline{H}}$. The union of $\mathbf{M}_H$ and a maximum cardinality matching in $\mathrm{G}_{\overline{H}}$, is a maximum cardinality matching in G. The problem of finding a maximum cardinality matching in the smaller network $\mathrm{G}_{\overline{H}}$ remains. For that, root an alternating tree at an unmatched node in $\mathrm{G}_{\overline{H}}$ and repeat this process with it.

A different procedure is to root an alternating tree at each unmatched node in G and grow them all simultaneously. This is called a **planted forest**. In general the forest will consist of disjoint alternating trees each rooted at a separate unmatched node. This forest growth procedure leads to a more efficient algorithm than the procedure of growing only one alternating tree at a time. When growing a forest, it is necessary to maintain the identity of the rooted tree to which each labeled node belongs. The current network $\mathrm{G}^1 = (\mathcal{N}^1, \mathcal{A}^1)$ changes each time a simple blossom is shrunk in this forest growth process. Edges in the present G^1 can be generated as they are needed, and we outline methods for doing it efficiently here.

The original network is stored by storing $\mathcal{A}$. Each new pseudonode created is given a distinct identification number when it is formed; and we store the numbers of the current nodes on the simple blossom corresponding to it, together with the present labels on them; its identifying pair of nodes, base and apex nodes; and the set of original nodes contained inside it. We store the present matching $\mathbf{M}$, and update it whenever it changes in augmentation steps. We store $\mathcal{N}^1$, the set of numbers of all the current nodes, and update it whenever it changes.

If i is an original node which is a current node, the set of current edges incident at i is $\{(i;p) : p \in \mathcal{N}^1$ and p contains a j inside it such that $(i;j) \in \mathcal{A}\}$. This set can be easily generated from the stored data whenever needed. If i is a matched node, let its mate be j, the current matching edge incident at i is $(i;p)$ where p is the unique current node containing j inside it.

If p is a pseudonode which is a current node, the set of current edges incident at p is $\{(p;q) : q$ is a current node such that there exists an original node i inside p, and an original node j inside q, with $(i;j) \in \mathcal{A}\}$. There can exist at most one original node i inside p such that it is joined to an original node j outside p by a matching edge. If no such node i exists, p is unmatched. If such a node i exists, find its mate j, and the current node q containing j inside it, then $(p;q)$ is the current matching edge incident at p. All this can be generated efficiently from the stored information whenever needed.

In practice, it seems to be convenient to also maintain and update the set of current matching edges, but generate all other data about the current network from the stored information as needed. We also maintain a record of all the pseudonodes in which each node lies, in the order of outermost first, this information is needed for expanding the pseudonodes on a path containing the node.

10.1.1 Blossom Algorithm for the Maximum Cardinality Matching Problem

Based on the ideas discussed so far, we provide an augmenting path method called the **blossom algorithm** for finding a maximum cardinality matching in an undirected network G $= (\mathcal{N}, \mathcal{A})$. It grows an **al-**

ternating forest with an alternating tree planted at each unmatched node. In this algorithm, an augmenting path is identified when a pair of trees become augmenting trees together, the augmenting path will then be the alternating path between the root nodes of these trees. **M**, $G^1 = (\mathcal{N}^1, \mathcal{A}^1)$, $\mathbf{M}^1$ always denote the present matching in G, the present current network, and the present current matching respectively. In the algorithm, node labels have 3 entries, the predecessor index, the symbol + (for outer nodes) or − (for inner nodes), and the root index, in that order. List always refers to the present set of labeled and unscanned nodes.

BLOSSOM ALGORITHM FOR THE
MAXIMUM CARDINALITY MATCHING PROBLEM

Step 1 Initialization Choose an initial matching (it could be $\emptyset$) in G.

Step 2 Rooting an Alternating Forest If there are no unmatched nodes, go to Step 7. Otherwise root an alternating tree at each unmatched node i, by labeling it with $(\emptyset, +, i)$. List now consists of all these root nodes.

Step 3 Select A Node to be Scanned If list $= \emptyset$, go to Step 7. Otherwise select one node from the list to scan and delete it from the list.

Step 4 Scanning Let the node to be scanned be the current node i with label $(\mathrm{P}(i), \pm, r)$.

Scanning an Outer Node If i is an outer node, for each $j \neq \mathrm{P}(i)$ such that $(i;j)$ is a current edge (all these will be nonmatching edges) do the following.

If j is an already labeled outer node associated with a root $\neq r$, an augmenting path has been found, go to Step 5.

If j is an already labeled outer node with the same root r, the alternating tree containing i and j has blossomed, go to Step 6.

If j is an already labeled inner node, continue.

If j is unlabeled, label it with $(i, -, r)$ and include it in the list.

Scanning an Inner Node If i is an inner node, it cannot be unmatched since all unmatched nodes are outer root nodes, let $(i; j)$ be the current matching edge incident at i.

If j is already labeled, it must be an inner node too. If the root indices of i and j are the same, the tree containing them has blossomed, go to Step 6. If the root indices of i, j are different, an augmenting path has been found, go to Step 5.

If j is unlabeled, label it with $(i, +, r)$ and include it in the list.

Go back to Step 3.

Step 5 Augmentation We come to this step when scanning has revealed a pair of adjacent current nodes i, j associated with different root nodes $r(i) \neq r(j)$, such that either i, j are both outer nodes and $(i; j) \notin \mathbf{M}^1$, or i, j are both inner nodes and $(i; j) \in \mathbf{M}^1$. Combine the predecessor paths of i, j together with edge $(i; j)$, leading to the path $\mathcal{P}^1$. $\mathcal{P}^1$ is an augmenting path between $r(i)$ and $r(j)$. Let t_1, t_2 be the apex nodes of $r(i), r(j)$. Find the corresponding augmenting path $\mathcal{P}$ between t_1 and t_2 in G by expanding all the pseudonodes on $\mathcal{P}^1$. Erase all the labels on the nodes in the two trees containing i, j, and throw away all the blossoms in them (this operation is called **dismembering the two trees**). Rematch using $\mathcal{P}$, and revise the current matching accordingly. Nodes t_1, t_2 are matched in the new matching. If there are no trees left in the forest go to Step 7, otherwise put all the outer current nodes in the list and go back to Step 3.

Step 6 Blossom Shrinking We come to this step when scanning has revealed a pair of adjacent current nodes i, j associated with the same root node, such that either i, j are both outer nodes and $(i; j) \notin \mathbf{M}^1$, or i, j are both inner nodes and $(i; j) \in \mathbf{M}^1$. i, j are the identifying pair of nodes for a simple blossom in the current network, whose base node t is the first common node on their predecessor paths. The nodes on this simple blossom are the base node, and those remaining in these paths after eliminating the common nodes on them; store these nodes together with the predecessor index and the $+$ or $-$ indicator for outer, inner status, in the labels on them. Shrink this simple blossom into a pseudonode, say B_p. Change the predecessor index of all the nodes outside this simple blossom which are immediate successors of nodes on this simple blossom, into B_p. Give B_p the same label as on the base node t. If t is unmatched, B_p becomes the new root node of the tree containing it after the shrinking, change the root index of all the nodes on the tree to B_p. Include B_p in the list and go back to Step 3.

Step 7 Termination When we come to this step either we have a perfect matching in G, or all the planted trees have become Hungarian trees. In the latter case we say that the labeling has become hungarian, and the forest has become a *Hungarian forest.* The present matching in G is a maximum cardinality matching, terminate.

Discussion

Each augmentation step increases the cardinality of the matching by 1. So, Step 5 is carried at most $n/2$ times in the algorithm. Between any two consecutive occurrences of Step 5, Step 6 can be carried out at most $n/2$ times by Theorem 10.6, and Step 3 will be carried out at most $O(n)$ times. Each execution of Step 3 requires at most $O(n)$ effort. Each execution of Steps 5 or 6 requires tracing the predecessor paths, which requires at most $O(n)$ effort. Thus the overall work between two consecutive occurrences of Step 5 requires at most $O(n^{\mathbf{2}})$ effort. So, the overall computational effort in the algorithm is bounded above by $O(n^{\mathbf{3}})$.

Comment 10.1 The pioneering work on the blossom algorithm is due to Edmonds [1965a, b], and the original version of this algorithm is due to him.

10.1.2 The Minimum Cost Perfect Matching Problem

We consider the problem of finding a minimum cost perfect matching in G $= (\mathcal{N}, \mathcal{A}, c = (c_{ij}))$ with $|\mathcal{N}| = n, |\mathcal{A}| = m$ and c as the vector of edge cost coefficients. We assume that n is even, otherwise there is no perfect matching in G. The problem is to find $x = (x_{ij} : (i;j) \in \mathcal{A})$ that

$$\begin{array}{rrcl} \text{Minimizes} & z(x) & = & \sum(c_{ij}x_{ij} : \text{ over } (i;j) \in \mathcal{A}) \\ \text{subject to} & x(i) & = & 1, \text{ for all } i \in \mathcal{N} \\ & x_{ij} & \geqq & 0 \text{ for all } (i;j) \in \mathcal{A} \end{array} \qquad (10.17)$$

$$x_{ij} \quad \text{integer for all } (i;j) \qquad (10.18)$$

where $x(i)$ is defined in (10.1). It is not necessary to include the constraint $x_{ij} \leqq 1$ in this problem, as the first constraint in (10.17) implies it automatically. Let $\mathbf{Y} \subset \mathcal{N}$ with $|\mathbf{Y}|$ odd and $\geqq 3$. Define $\mathbf{Y}^-(x)$ as below. If x is a matching vector, (10.20) must hold.

$$\mathbf{Y}^-(x) = \sum(x_{ij} : \text{ over } i,j \text{ both} \in \mathbf{Y} \text{ and } (i;j) \in \mathcal{A}) \qquad (10.19)$$

$$\mathbf{Y}^-(x) \leqq (|\mathbf{Y}| - 1)/2 \qquad (10.20)$$

The constraint (10.20) is known as the **matching blossom inequality** *or* **matching blossom constraint corresponding to Y** for (10.17), (10.18). Each subset of $\mathcal{N}$ of odd cardinality $\geqq 3$ leads to a blossom inequality, and every matching vector satisfies all matching blossom inequalities. Let $\{\mathbf{Y}_1, \ldots, \mathbf{Y}_L\}$ be the set of all distinct subsets of $\mathcal{N}$ of odd cardinality $\geqq 3$. Now consider the following LP.

$$\begin{array}{rrcl} \text{Minimize} & z(x) & = & \sum(c_{ij}x_{ij} : \text{ over } (i;j) \in \mathcal{A}) \\ \text{subject to} & x(i) & = & 1, \text{ for all } i \in \mathcal{N} \\ & \mathbf{Y}_\sigma^-(x) & \leqq & (|\mathbf{Y}_\sigma| - 1)/2, \sigma = 1 \text{ to } L \\ & x_{ij} & \geqq & 0 \text{ for all } (i;j) \in \mathcal{A} \end{array} \qquad (10.21)$$

Every perfect matching vector in G is feasible to (10.21) and every integer feasible vector for (10.21) is a perfect matching vector in G. So, if an optimum solution of (10.21) is an integer vector, it is a minimum cost perfect matching vector in G. We will discuss a primal-dual algorithm for solving (10.21) known as the **blossom algorithm for the minimum cost perfect matching problem** and show that if a perfect matching exists in G, then this algorithm terminates with an optimum solution of (10.21) which is an integer vector, and this integer vector is therefore a minimum cost perfect matching vector in G.

To write the dual of (10.21), associate a dual variable π_i with the constraint corresponding to node i in (10.21), and a dual variable μ_σ with the blossom inequality corresponding to the odd subset $\mathbf{Y}_\sigma, \sigma = 1$ to L. The dual variables π_i are only associated with original nodes in G, and hence are also called **original node prices**. The dual variables μ_σ are known as **pseudonode prices**, the reason for this name will become clear later. Let $\pi = (\pi_i), \mu = (\mu_\sigma)$. Since the number of odd subsets of nodes, L, grows exponentially with n, the vector μ has a lot of entries. Given the dual solution (π, μ), define for each $(i; j) \in \mathcal{A}$

$$\mu^-(i;j) = \sum(\mu_\sigma : \text{over } \sigma \text{ s.t. both } i, j \in \mathbf{Y}_\sigma) \qquad (10.22)$$

$$d_{ij}(\pi, \mu) = \pi_i + \pi_j - \mu^-(i;j) \qquad (10.23)$$

The dual of (10.21) is

$$\begin{array}{rrcl} \text{Maximize} & W(\pi,\mu) & = & \sum_{i \in \mathcal{N}} \pi_i - \sum_{\sigma=1}^{L} (|\mathbf{Y}_\sigma| - 1)(\mu_\sigma)/2 \\ \text{subject to} & d_{ij}(\pi,\mu) & \leqq & c_{ij}, \text{ for each } (i;j) \in \mathcal{A} \qquad (10.24) \\ & \mu & \geqq & 0 \end{array}$$

Given a dual feasible solution (π, μ), define $\mathcal{A}_*(\pi, \mu)$ as in (10.25). Since (π, μ) satisfies the first constraint in (10.24) as an equation for each edge in $\mathcal{A}_*(\pi, \mu)$, they are called **equality edges** wrt (π, μ), and the subnetwork $\mathrm{G}_*(\pi, \mu) = (\mathcal{N}, \mathcal{A}_*(\pi, \mu))$ is known as the **equality subnetwork** wrt (π, μ) for (10.21). The complementary slackness conditions for optimality in the primal, dual pair (10.21), (10.24) are (10.26), (10.27) given below.

$$\mathcal{A}_*(\pi,\mu) \;=\; \{(i;j) : (i;j) \in \mathcal{A}, \text{ and } d_{ij}(\pi,\mu) = c_{ij}\} \quad (10.25)$$

$$x_{ij} \;>\; 0 \Rightarrow d_{ij}(\pi,\mu) = c_{ij}, \text{ for each } (i;j) \in \mathcal{A} \quad (10.26)$$
$$\mu_\sigma \;>\; 0 \Rightarrow \mathbf{Y}^-_\sigma(x) = (|\mathbf{Y}_\sigma| - 1)/2, \sigma = 1 \text{ to} L \quad (10.27)$$

The blossom algorithm is initiated with an initial dual feasible solution (π^0, μ^0) in which $\mu^0 = 0$. If we can find a perfect matching in the equality subnetwork $\mathrm{G}_*(\pi^0, \mu^0)$, it is a minimum cost perfect matching in G, since the corresponding perfect matching vector x satisfies the complementary slackness optimality conditions (10.26), (10.27) together with (π^0, μ^0). For this we apply the maximum cardinality matching algorithm of Section 10.1.1 on $\mathrm{G}_*(\pi^0, \mu^0)$ beginning with the empty matching. Suppose this leads to the matching $\mathbf{M}_1$ in $\mathrm{G}_*(\pi^0, \mu^0)$, corresponding to the matching vector x^1. So, all matching edges in $\mathbf{M}_1$ are equality edges in $\mathcal{A}_*(\pi^0, \mu^0)$ and (10.26), (10.27) hold for $x^1, (\pi^0, \mu^0)$. In the process of applying this algorithm blossoms may have been discovered and shrunk into pseudonodes, leading to **current equality subnetworks** $\mathrm{G}^1_*(\pi^0, \mu^0)$ of the form $(\mathcal{N}^1, \mathcal{A}^1_*(\pi^0, \mu^0))$ where $\mathcal{N}^1$ is the set of current nodes, and $\mathcal{A}^1_*(\pi^0, \mu^0)$ is the set of current equality edges.

If $\mathbf{M}_1$ is not a perfect matching, we are left with a Hungarian forest in the equality subnetwork $\mathrm{G}^1_*(\pi^0, \mu^0)$, with a Hungarian tree rooted at each unmatched current node in it. Each pseudonode at this stage corresponds to a shrunken blossom wrt the present matching $\mathbf{M}_1$, the set of original nodes inside it has odd cardinality $\geqq 3$, and hence there is a blossom inequality and a dual variable μ_σ associated with it. By the second property in (10.15) of blossoms, we verify that the present matching vector x^1 satisfies the blossom inequality corresponding to every existing pseudonode (whether it is an outermost pseudonode or not) as an equation.

To get things moving, the approach now goes to a dual solution change step. The purpose of this is to obtain a new dual feasible solution, the equality subnetwork corresponding to which allows a matching of higher cardinality than $\mathbf{M}_1$. This dual solution change step is designed to satisfy the following properties.

(i) All present matching edges (which are equality edges now) remain equality edges after the change, so that the present matching vector x^1 continues to satisfy (10.26) together with the new dual solution.

(ii) The equality, nonequality status of all original edges contained within any existing pseudonode remains unchanged, and all in-tree current equality edges remain equality edges after the dual solution change. So, all the existing alternating trees in the current equality subnetwork, are also contained in the new current equality subnetwork after the dual solution change.

(iii) In the new current equality subnetwork obtained after the dual solution change, the existing forest is not hungarian, i.e., at least one of the trees can grow, or there is at least one augmenting path, etc. This makes it possible to repeat the application of the maximum cardinality matching algorithm in the new current equality subnetwork, beginning with the existing forest, and make some movement in the algorithm.

(iv) In the new dual solution, some of the dual variables μ_σ may be given positive values, but if a μ_σ is positive, the corresponding subset of original nodes $\mathbf{Y}_\sigma$ will always be the set of original nodes inside an existing pseudonode (outermost or not). We have already seen that the present matching vector satisfies the blossom inequality corresponding to every existing pseudonode as an equation, so this guarantees that the complementary slackness condition (10.27) continues to hold.

The application of the maximum cardinality matching algorithm is continued with the present alternating forest in the new current equality subnetwork, and this process is repeated. The following summarizes the properties maintained by the algorithm.

> It maintains a matching vector x, and (π, μ) always feasible to (10.24). It alternates between changing the matching vector x, keeping (π, μ) constant, using the maximum cardinality matching algorithm in the equality subnetwork $\mathrm{G}_*(\pi, \mu)$; or changing (π, μ) keeping x constant.
>
> $x_{ij} = 1$ implies that $(i; j) \in \mathcal{A}_*(\pi, \mu)$, so, (10.26) holds always. (10.28)
>
> $\mu_\sigma > 0$ always implies that the associated $\mathbf{Y}_\sigma$ is the set of original nodes inside an existing pseudonode, so (10.27) holds always. This also guarantees that even though the dual vector $\mu = (\mu_\sigma)$ has a large number of entries, all but at most $(n/2)$ of them will be 0 at every stage. Hence, it is only necessary to store values of μ_σ associated with each pseudonode at that stage of the algorithm. That's why μ_σ are known as **pseudonode prices.**

Changes in Blossoms after an Augmentation Step

Let $x, (\pi, \mu)$ be the solution pair at some stage during the matching change phase of the algorithm. If an augmenting path $\mathcal{P}^1$ is discovered, we would augment. In the maximum cardinality matching algorithm we then discard all the existing blossoms on the two trees containing nodes along $\mathcal{P}^1$. However, some of these blossoms may correspond to pseudonodes associated with a positive μ_σ in the present dual solution (π, μ) and discarding these blossoms will violate the third property in (10.28). So, in this algorithm, all the blossoms along the augmenting path are retained after augmentation, but changes have to be made in the stored data on the simple blossoms corresponding to them, to reflect the change in the matching caused by the augmentation step. This is called the operation of **revising all the blossoms along $\mathcal{P}^1$** .

We describe this operation now for a pseudonode B either on $\mathcal{P}^1$ or

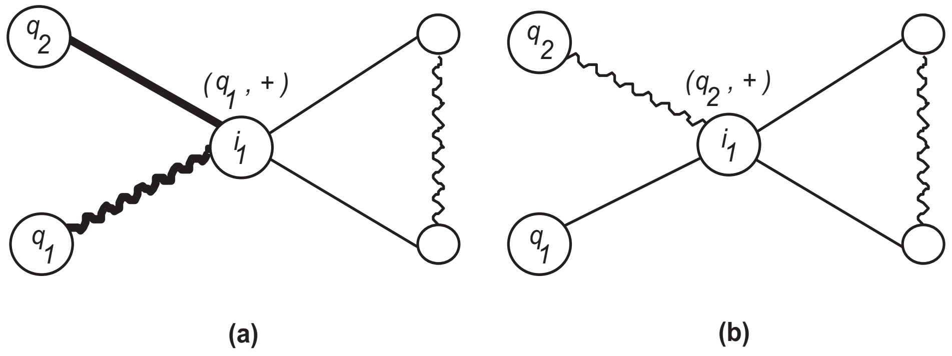

Figure 10.19: Before augmentation (a), and after (b). Matching edges are wavy. In Figure 10.19 (a), $\mathcal{P}^0$ is thick. q_1, q_2 are not on the simple blossom. Stored label on i_1 is entered by its side.

inside some node on $\mathcal{P}^1$. Let $\mathcal{P}$ be the augmenting path in $\mathrm{G}_*(\pi, \mu)$ corresponding to $\mathcal{P}^1$. Let i_1 be the base node (which may itself be another pseudonode) of B. When all the pseudonodes along $\mathcal{P}^1$ containing B within them are expanded, we will get the portion of the augmenting path passing through the simple blossom corresponding to B. We will denote this portion of the path by $\mathcal{P}^0$. We consider two cases.

Case 1 Apex of Pseudonode B Is an Intermediate Node On $\mathcal{P}$
In this case the augmenting path $\mathcal{P}$ passes through the matching edge incident at the apex node of B, and leaves through either the apex node or some nonapex node within B. We consider two subcases.

Subcase 1 $\mathcal{P}$ Passes Only Through the Apex Node of B
In this subcase $\mathcal{P}^0$ appears as in Figure 10.19 (a). After the augmentation is carried out, $(q_1; i_1)$ becomes a nonmatching edge and $(i_1; q_2)$ becomes a matching edge. The only change needed in the stored data on this simple blossom, is to change the label on its base node i_1 as in Figure 10.19 (b).

Subcase 2 $\mathcal{P}$ Contains Some Nonapex Nodes Contained in B
In this subcase $\mathcal{P}^0$ appears as in Figure 10.20. The i-nodes

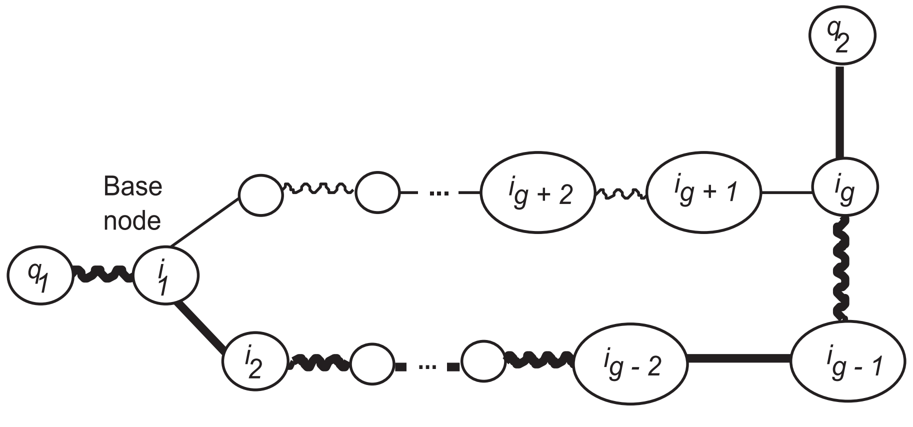

Figure 10.20: Matching edges are wavy. $\mathcal{P}^0$, the thick path, contains more than one node on the simple blossom corresponding to B. See Figure 10.21 for position after augmentation.

are nodes on the simple blossom corresponding to B, and q_1, q_2 are nodes outside B. The odd cycle in this simple blossom remains the same, but augmentation changes the matching edges along $\mathcal{P}^0$ into nonmatching edges and vice versa. After augmentation (see Figure 10.21) $(i_1; i_2)$ is a matching edge, and i_1 is no longer the base node, i_g becomes the new base node of this simple blossom, and its apex will be the revised apex of i_g (i.e., after the corresponding change due to augmentation is carried out for i_g). Change the label on i_g in the stored data to $(q_2, +)$. Change the labels on both the neighbor nodes of i_g on the odd cycle,

i_{g+1} and i_{g-1}, to $(i_g, -)$. Change the labels on i_{g+2}, i_{g-2} to $(i_{g+1}, +)$ and $(i_{g-2}, +)$ respectively. Keep on changing the labels along the odd cycle this way until all the node labels are changed. The last pair of nodes to be relabeled in this manner are the new identifying pair of nodes for this simple blossom. It is clear that the odd alternating cycle in this simple blossom, the new matching and nonmatching edges in it, can all be retrieved by the procedures discussed earlier

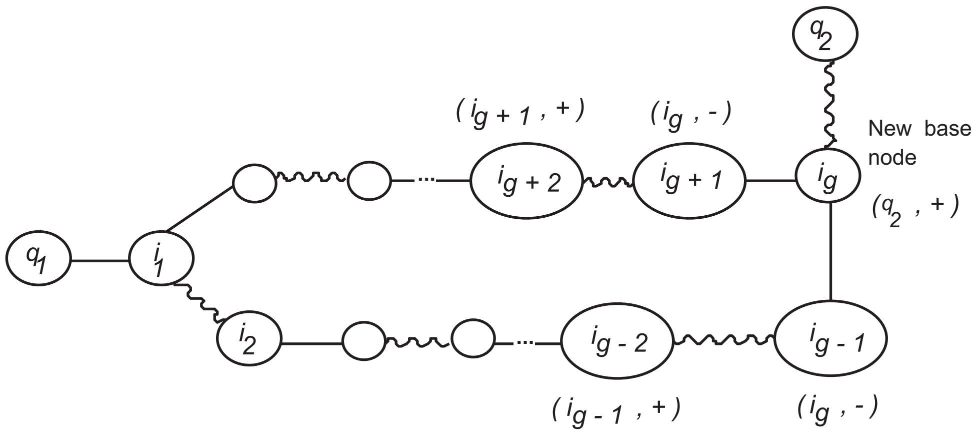

Figure 10.21: Position after augmentation in the simple blossom in Figure 10.20. New base node and new node labels are indicated.

using the revised labels on the nodes.

Case 2 Apex of Pseudonode B Is a Terminal Node of $\mathcal{P}$ In this case i_1, the base node of B is an unmatched node. We again consider two subcases.

Subcase 1 $\mathcal{P}$ Contains Only the Apex Node of B In this subcase $\mathcal{P}^0$ appears as in Figure 10.22 (a). After the augmentation $(q_1; i_1)$ becomes a matched edge, change the stored label on i_1 as in Figure 10.22 (b).

Subcase 2 $\mathcal{P}$ Contains Some Nonapex Nodes Contained in B In this subcase $\mathcal{P}^0$ appears as in Figure 10.23. In this subcase i_g becomes the new base node after augmentation, and the revision is carried out as in Subcase 2 of Case 1. See Figure 10.24.

After the augmentation, the existing two alternating trees containing nodes on the path used are no longer rooted at unmatched nodes, in fact all the nodes on these two trees are matched. So, we erase these two trees by erasing the present labels on the current nodes in them, and leaving all these current nodes as unlabeled

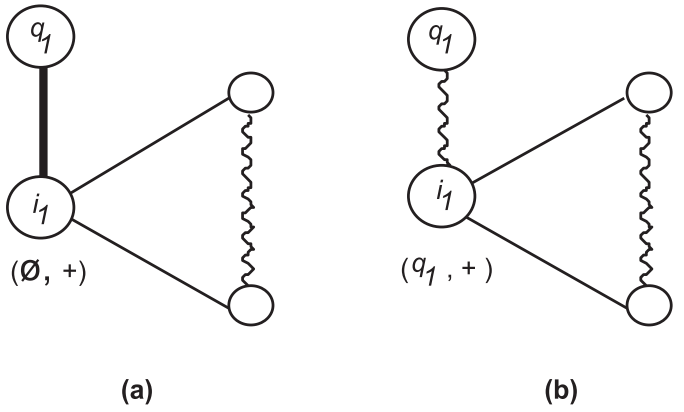

Figure 10.22: Before augmentation (a), and after (b). $\mathcal{P}^0$ is thick in Figure (a), and contains only unmatched base node i_1. q_1 is not on the simple blossom of B. Stored label on i_1 is entered by its side.

nodes. This operation is called **chopping down the trees**. Notice the difference between the operation of dismembering the trees (in Step 5 of the algorithm in Section 10.1.1, all the blossoms on the trees are thrown away), and that of chopping down the trees discussed here (only the tree structures are eliminated, but all the current nodes on them are left as unlabeled nodes in the current network).

Each of the out-of-tree (i.e., unlabeled) current nodes which are pseudonodes, are matched nodes, and the blossoms corresponding to them satisfy the conditions (10.15) wrt the present matching, but they may not satisfy (10.16). This is another difference between the algorithm in Section 10.1.1, and the one to be discussed here.

When the growth of the remaining alternating trees in the present $G^1_*(\pi, \mu)$ is resumed, it is possible that some of the unlabeled pseudonodes get labeled. In this process they may get labeled either as outer or inner nodes. In Section 10.1.1, pseudonodes were always outer nodes, and carried the same label as their base nodes. Here these labels may be quite different. Also, pseudonodes may be labeled as inner nodes.

In the algorithm to be discussed in this section a freshly shrunk pseudonode always gets labeled as an outer node, its label at that time will be the same as that on the base node of the corresponding simple

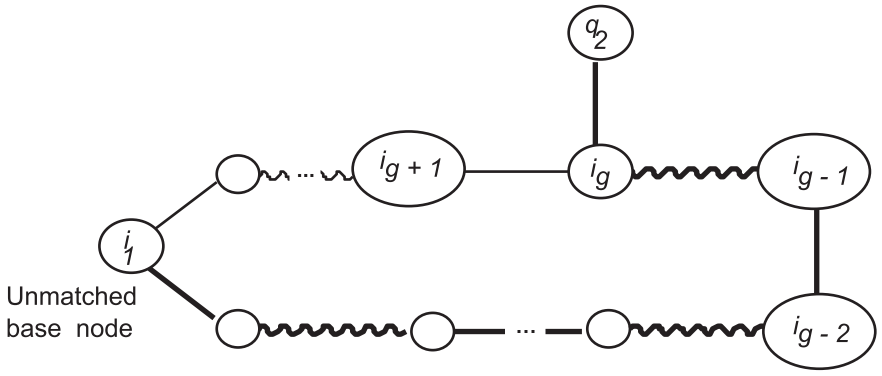

Figure 10.23: Matching edges are wavy. $\mathcal{P}^0$ is thick, it contains more than one node on the simple blossom of B. See Figure 10.24 for position after augmentation.

blossom before it was shrunk. So, any inner labeled pseudonode in the current network must have received that label after remaining as an unlabeled node for some time.

THEOREM 10.15 *Let pseudonode p be an outer labeled current node, then the blossom corresponding to p satisfies (10.16).*

Proof If p is unmatched, it is the root of its alternating tree and (10.16) is satisfied trivially. Suppose p is matched. The root of its alternating tree is unmatched. In this case, the alternating path required in condition (10.16) can be obtained from the predecessor path of p. ∎

Hence in the algorithm to be discussed in this section, the only pseudonodes whose blossoms may not satisfy (10.16) are inner labeled pseudonodes. When there are pseudonodes in $\mathrm{G}^1_*(\pi, \mu)$ violating (10.16), there may exist augmenting paths wrt the present matching in $\mathrm{G}_*(\pi, \mu)$, and yet there may not exist any augmenting path in $\mathrm{G}^1_*(\pi, \mu)$ wrt the current matching. Hence these pseudonodes may prevent us from discovering augmenting paths in $\mathrm{G}_*(\pi, \mu)$ through operations on $\mathrm{G}^1_*(\pi, \mu)$. The only reason for keeping such pseudonodes is to satisfy the third property in (10.28), if the dual variables μ_σ corresponding to them

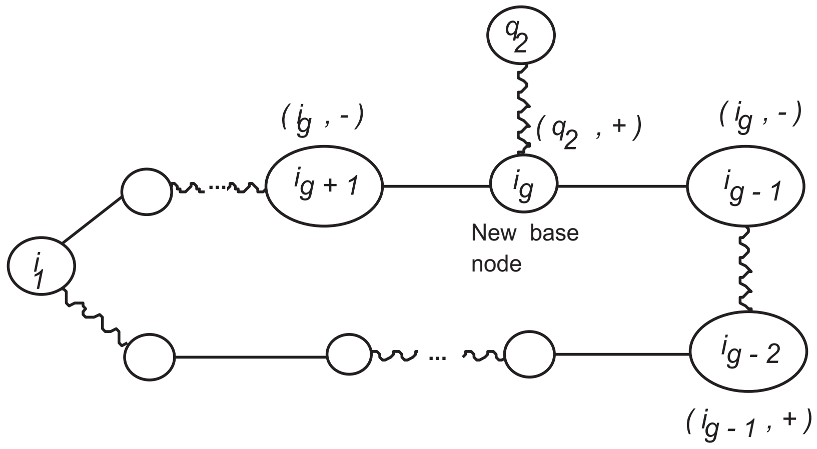

Figure 10.24: New base node, new labels after augmentation are indicated.

are strictly positive in the present dual solution. Therefore, whenever there is a pseudonode which is an inner current node associated with $\mu_\sigma = 0$ in the present dual solution, we unshrink that pseudonode into the simple blossom corresponding to it. This **unshrinking operation** is discussed next.

Unshrinking an Inner Labeled Pseudonode Associated with $\mu_\sigma = 0$

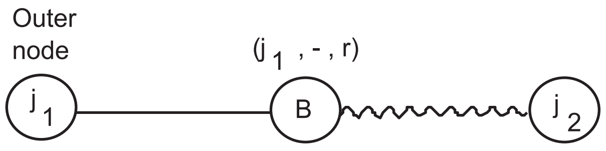

Figure 10.25: Wavy edge is a current matching edge. Inner labeled pseudonode B associated with $\mu_\sigma = 0$ to be unshrunk. Label of B is by its side.

This operation is carried out only on pseudonodes which are inner labeled current nodes associated with $\mu_\sigma = 0$. Let B with label

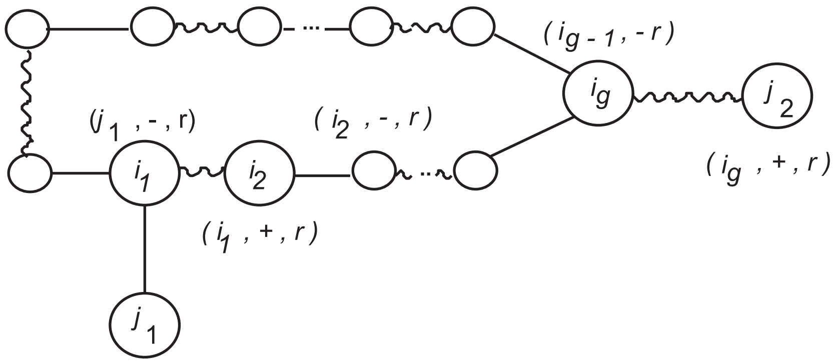

Figure 10.26: Connecting the tree through the alternating path in the simple blossom, after a pseudonode is unshrunk.

$(j_1, -, r)$ be such a pseudonode. So, j_1, is an outer node and $(j_1; B)$ is a current nonmatching edge. By earlier discussion such a pseudonode will always be matched, let $(B; j_2)$ be the current matching edge incident at it. See Figure 10.25. Let $\overline{\mathrm{G}}_B = (\overline{\mathcal{N}}_B, \overline{\mathcal{A}}_B)$ be the simple blossom corresponding to B. Since $(j_1; B)$ is a current in-tree nonmatching edge, there must exist an $i_1 \in \overline{\mathcal{N}}_B$ such that $(j_1; i_1)$ is a current equality edge at the stage that pseudonode B was formed. Either i_1 is the base node of B, or there exists an AP in $\overline{\mathrm{G}}_B$ from i_1 beginning with the matching edge incident at i_1, to its base node. Let this path be $i_1, (i_1, i_2), i_2, \ldots, (i_{g-1}, i_g), i_g$, with i_g being the base node of $\overline{\mathrm{G}}_B$ (if i_1 is itself the base node, $g = 1$, and this path is the empty path containing no edges). So, $(i_g; j_2)$ is a current matching edge at the stage that pseudonode B was formed. See Figure 10.26.

Unshrinking of B replaces it with the simple blossom $\overline{\mathrm{G}}_B$ with all its nodes and edges on it. Each of the current edges of the form $(p; B)$ are replaced by edges of the form $(p; i)$ for $i \in \overline{\mathcal{N}}_B$. After the unshrinking, all the nodes in $\overline{\mathcal{N}}_B$ become current nodes. Update the current matching by replacing the single current matching edge $(B; j_2)$ from it by $(i_g; j_2)$ and all the matching edges within the simple blossom $\overline{\mathrm{G}}_B$. The alternating tree that contained B before unshrinking, is connected again by labeling the newly introduced nodes along the path

$j_1, (j_1, i_1), i_1, \ldots, i_g, (i_g, j_2), j_2$ alternately as inner and outer nodes as indicated in Figure 10.26.

In the algorithm discussed below, the dual solution change routine has the effect of reducing the value of μ_σ corresponding to pseudonodes which are inner labeled current nodes. So, after each dual solution change step, we always check for possibilities of carrying out unshrinking.

The operation of unshrinking is not needed in the blossom algorithm for the maximum cardinality matching problem, it is needed in the blossom algorithms for minimum cost matching problems. The reader is cautioned not to confuse the two operations of expanding pseudonodes along an alternating path, and unshrinking pseudonodes.

A **dormant period** for a pseudonode B begins when it is just absorbed inside another newly formed pseudonode (i.e., when it just became dormant), and it ends when the pseudonode B becomes a current node again due to all the pseudonodes containing it inside becoming unshrunk.

In the algorithm described below, the matching change phase stops when we reach a current equality subnetwork containing a Hungarian forest. This state is characterized by the following conditions.

> No alternating tree can grow any further (i.e., the list of labeled and unscanned nodes is empty). There are no augmenting paths in the current equality subnetwork wrt the current matching. There are no blossoms which can be shrunk. There are no inner labeled pseudonodes associated with pseudonode price $\mu_\sigma = 0$.

These are the **Hungarian forest conditions** for the blossom algorithm for minimum cost matching problems, and when they are satisfied we say that **the labeling has become hungarian**. The algorithm then goes to the dual change phase, which obtains a new dual feasible solution satisfying the following properties: All the present matching edges, and the present alternating trees lie in the new equality subnetwork, but the Hungarian forest conditions are not satisfied in it anymore. So, in the new equality subnetwork, at least one of tree growth, augmentation, blossom shrinking, or pseudonode unshrinking steps can be carried out when labeling is resumed. And the method continues.

BLOSSOM ALGORITHM FOR THE MINIMUM COST PERFECT MATCHING PROBLEM

Step 1 Initialization An initial dual feasible solution is $(\pi^0 = (\pi_i^0), \mu^0)$ where $\mu^0 = 0$ and $\pi_i^0 = (1/2)(\text{min. } \{c_{ij} : (i; j) \in \mathcal{A}\})$ for each $i \in \mathcal{N}$. Choose an initial matching (could be empty) in the equality subnetwork $\mathrm{G}_*(\pi^0, \mu^0)$.

Step 2 Rooting an Alternating Forest If there are no unmatched nodes, go to Step 10. Otherwise root an alternating tree at each unmatched node i, by labeling it with $(\emptyset, +, i)$. List now consists of all these root nodes.

Step 3 Select a Node to be Scanned If list $= \emptyset$, go to Step 8. Otherwise select one node from the list to scan and delete it from the list.

Step 4 Scanning Let the node to be scanned be the current node i with label $(\mathrm{P}(i), \pm, r)$.

Scanning an Outer Node If i is an outer node, for each $j \neq \mathrm{P}(i)$ such that $(i; j)$ is a current equality edge (all these will be nonmatching edges) do the following.

If j is an already labeled outer node associated with a root $\neq r$, an augmenting path has been found, go to Step 5.

If j is an already labeled outer node with the same root r, the alternating tree containing i and j has blossomed, go to Step 6.

If j is an already labeled inner node, continue.

If j is unlabeled, label it with $(i, -, r)$ and include it in the list.

Scanning an Inner Node If i is an inner node, it cannot be unmatched since all unmatched nodes are outer root nodes, let $(i; j)$ be the current matching edge incident at i.

If j is already labeled, it must be an inner node too. If the root indices of i and j are the same, the tree containing them has blossomed, go to Step 6. If the root indices of i, j are different, an augmenting path has been found, go to Step 5.

If j is unlabeled, label it with $(i, +, r)$ and include it in the list.

Go back to Step 3.

Step 5 Augmentation We come to this step when scanning has revealed a pair of adjacent current nodes i, j associated with different root nodes $r(i) \neq r(j)$, contained on an augmenting path. Combine the predecessor paths of i, j together with edge $(i; j)$, leading to the path $\mathcal{P}^1$. $\mathcal{P}^1$ is an augmenting path between $r(i)$ and $r(j)$ in the current equality subnetwork. Let t_1, t_2 be the apex nodes of $r(i), r(j)$. Find the corresponding augmenting path $\mathcal{P}$ between t_1 and t_2 in the present equality subnetwork in G by expanding all the pseudonodes on $\mathcal{P}^1$. Rematch using $\mathcal{P}$, and revise all the blossoms along $\mathcal{P}^1$ as discussed above. Chop down the two trees containing nodes on $\mathcal{P}^1$. If there are no unmatched nodes, go to Step 10, otherwise put all the outer current nodes in the list and go back to Step 3.

Step 6 Blossom Shrinking We come to this step when scanning has revealed a pair of adjacent current nodes i, j associated with the same root node, which are the identifying pair for a simple blossom in the current equality subnetwork. Identify this simple blossom, store all the necessary data on it, and shrink it into a new pseudonode B_p, label B_p, revise the root index of all the current nodes in the tree, revise the labels on the immediate successors of nodes in this simple blossom, revise $\mathcal{N}^1$ and $\mathbf{M}^1$, exactly as in Step 6 of the algorithm in Section 10.1.1. Include B_p in the list and go back to Step 3

Step 7 Pseudonode Unshrinking Unshrink all pseudonodes that are current inner nodes with the associated pseudonode price $\mu_\sigma = 0$ in the present dual solution. Revise the set of current matching edges, and the set of current nodes accordingly. Include in the list all the new outer labeled nodes in the simple blossoms corresponding to the unshrunk pseudonodes. Repeat this procedure again if necessary, until there are no pseudonodes that are inner labeled current nodes associated with $\mu_\sigma = 0$. Go back to Step 3.

Step 8 Dual Solution Change We reach this step if we do not yet have a perfect matching, and the Hungarian forest conditions are satisfied. This implies that the present matching is a maximum cardinality matching in the current equality subnetwork. Let (π, μ) be the present dual feasible solution. Compute the following using the convention that the minimum in the empty set is $+\infty$.

$$\begin{aligned}
\delta_1 &= \text{Min. } \{c_{ij} - d_{ij}(\pi, \mu) : (i; j) \in \mathcal{A}, i[j] \text{ is inside an outer [an unlabeled] current node}\} \\
\delta_2 &= \text{Min. } \{\tfrac{1}{2}(c_{ij} - d_{ij}(\pi, \mu)) : (i; j) \in \mathcal{A}, i \text{ and } j \text{ are inside distinct outer current nodes}\} \\
\delta_3 &= \text{Min. } \{\tfrac{1}{2}\mu_\sigma : \sigma \text{ s. t. } \mathbf{Y}_\sigma \text{ is the set of original nodes inside a current inner labeled pseudonode}\} \\
\delta &= \text{Min. } \{\delta_1, \delta_2, \delta_3\}
\end{aligned}$$

If $\delta = +\infty$, go to Step 9. If δ is finite, it will be positive (this is proved below), define the new dual solution to be $\hat{\pi} = (\hat{\pi}_i), \hat{\mu} = (\hat{\mu}_\sigma)$ where

$$\hat{\pi}_i = \begin{cases} \pi_i + \delta & \text{for all } i \text{ inside outer current nodes} \\ \pi_i - \delta & \text{for all } i \text{ inside inner current nodes} \\ \pi_i & \text{for all } i \text{ inside unlabeled nodes} \end{cases}$$

$$\hat{\mu}_\sigma = \begin{cases} \mu_\sigma + 2\delta & \text{if } \mathbf{Y}_\sigma \text{ is the set of original nodes in a current outer labeled pseudonode} \\ \mu_\sigma - 2\delta & \text{if } \mathbf{Y}_\sigma \text{ is the set of original nodes in a current inner labeled pseudonode} \\ \mu_\sigma & \text{otherwise} \end{cases}$$

Find $G_*(\hat{\pi}, \hat{\mu})$. Include all outer current nodes in the list. If $\delta = \delta_3$ go to Step 7. If $\delta < \delta_3$ go to Step 3.

Step 9 Infeasibility We come to this step if $\delta = +\infty$ in a dual solution change step. In this case there exists no perfect matching in G (this is proved below).

Step 10 Optimality We come to this step if the matching in the present $G_*(\pi, \mu)$ is a perfect matching, it is a minimum cost perfect matching in G, and the corresponding perfect matching vector is an optimum solution of (10.17), (10.18), or (10.21). Terminate.

Validity of the Algorithm and Its Computational Complexity

THEOREM 10.16 *If the present (π, μ) is dual feasible, in Step 8, δ will either be finite and > 0, or $+\infty$.*

Proof We execute Step 8 only when the Hungarian forest conditions are satisfied. This and the dual feasibility of (π, μ) implies that each of the sets of which $\delta_1, \delta_2, \delta_3$ are minima, is either empty or consists only of strictly positive entries. So, each of $\delta_1, \delta_2, \delta_3$ is either finite and > 0, or $+\infty$, hence the same thing holds for δ. ∎

THEOREM 10.17 *If (π, μ) is dual feasible just before executing Step 8, the vector $(\hat{\pi}, \hat{\mu})$ obtained after completing the execution of Step 8 is also dual feasible.*

Proof By Theorem 10.16, $\delta > 0$, and since $\delta \leqq \delta_3$ and $\mu \geqq 0$, we have $\hat{\mu} \geqq 0$.

Let $(i; j) \in \mathcal{A}$ with i and j both contained within a current node in the present $G^1_*(\pi, \mu)$ which is a pseudonode. Whether this pseudonode is unlabeled or labeled, it can be verified that in this case $d_{ij}(\hat{\pi}, \hat{\mu}) = d_{ij}(\pi, \mu) \leqq c_{ij}$, the last inequality because (π, μ) is dual feasible. So, $(\hat{\pi}, \hat{\mu})$ satisfies the dual constraint corresponding to this edge $(i; j)$. This also points out that if $d_{ij}(\pi, \mu) = c_{ij}$, then $d_{ij}(\hat{\pi}, \hat{\mu}) = c_{ij}$, so the equality, nonequality status of any edge contained within a pseudonode is unaffected by a dual solution change step.

Now consider an edge $(i; j) \in \mathcal{A}$ with nodes i, j contained within distinct current nodes, i in p, and j in q. Hence, in this case there exists no pseudonode which contains both i and j. If p is outer labeled and q is inner labeled, $d_{ij}(\hat{\pi}, \hat{\mu}) = \hat{\pi}_i + \hat{\pi}_j = \pi_i + \pi_j = d_{ij}(\pi, \mu) \leqq c_{ij}$. If both p, q are outer labeled, $d_{ij}(\hat{\pi}, \hat{\mu}) = \hat{\pi}_i + \hat{\pi}_j = \pi_i + \pi_j + 2\delta = d_{ij}(\pi, \mu) + 2\delta \leqq c_{ij}$ since $\delta \leqq \delta_2$ and (π, μ) is dual feasible. If p is outer labeled and q is unlabeled, $d_{ij}(\hat{\pi}, \hat{\mu}) = \hat{\pi}_i + \hat{\pi}_j = \pi_i + \pi_j + \delta = d_{ij}(\pi, \mu) + \delta \leqq c_{ij}$ since $\delta \leqq \delta_1$ and (π, μ) is dual feasible. If p, q are either inner labeled or unlabeled each, $d_{ij}(\hat{\pi}, \hat{\mu}) = \hat{\pi}_i + \hat{\pi}_j$, and this is either $\pi_i + \pi_j$ or $\pi_i + \pi_j - \delta$ or $\pi_i + \pi_j - 2\delta$, and $\pi_i + \pi_j = d_{ij}(\pi, \mu) \leqq c_{ij}$, this and $\delta > 0$ implies that $d_{ij}(\hat{\pi}, \hat{\mu}) \leqq c_{ij}$. Hence $(\hat{\pi}, \hat{\mu})$ satisfies the dual constraint corresponding to this edge $(i; j)$ in all these cases. This completes the proof that $(\hat{\pi}, \hat{\mu})$ is dual feasible.

Since $d_{ij}(\hat{\pi}, \hat{\mu}) = d_{ij}(\pi, \mu)$ for every $(i; j) \in \mathcal{A}$ with i inside an outer labeled node, and j inside an inner labeled node, we see that the equality, nonequality status of such edges is unaffected by a dual solution change step. ∎

The initial price vector (π^0, μ^0) is clearly dual feasible. By repeated application of the result in Theorem 10.17 after each dual solution change step in the algorithm, we conclude that all price vectors obtained in this algorithm are dual feasible.

THEOREM 10.18 *In this algorithm, all matching edges are always equality edges. Also, after each dual solution change step, in-tree edges remain equality edges.*

Proof The initial matching $\mathbf{M}_0 = \emptyset$, so the property that all matching edges are equality edges is trivially satisfied initially. Every augmenting path discovered in the algorithm is a path in the equality subnetwork at that stage, and new matching edges are only created during the augmentation step (Step 5), so they are all equality edges when they are created. If $(i; j)$ is a matching edge at the beginning of execution of Step 8, we must have, either (i) both i and j are contained inside a current node which is a pseudonode (labeled or unlabeled), or (ii) one of i, j is inside an outer labeled node and the other is within an inner labeled node in the same tree, or (iii) both i, j are inside distinct unlabeled nodes. The arguments in the proof of Theorem 10.17 imply that any such equality edge remains an equality edge wrt the new dual feasible solution obtained at the end of this Step 8. By repeated application of these results we conclude that all matching edges are always equality edges.

Each new in-tree edge created during scanning (Step 4) is always an equality edge when it joins the tree. Each in-tree edge always joins an outer to an inner node, and by the argument in the proof of Theorem 10.17, the equality, nonequality status of such edges remains unaffected during a dual solution change step. By repeated application of this result, we conclude that all in-tree edges are always equality edges. ∎

Thus, after each dual solution change step, the existing alternating forest is contained in the new current equality subnetwork, and its growth can be resumed. If δ was equal to δ_1 in that step, the edges which produced the value for δ_1 lead to new current equality edges, using which the trees containing the labeled nodes on them can be grown further, or a discovery of an augmenting path made. If δ was equal to δ_2 in that step, the edges which produced the value for δ_2 lead to new current equality edges, using which the trees containing the labeled nodes on them can be grown further, or a discovery of a new blossom made. If δ was equal to δ_3 in that step, all the current inner labeled pseudonodes which produced the value for δ_3 will now have zero pseudonode price, and these can now be unshrunk. Thus after an execution of Step 8, the existing alternating forest is one that is no longer hungarian in the new current equality subnetwork, and scanning can be resumed, and at least one tree growth step or Steps 5, or 6, or 7 can be carried out.

Initially $\mu^0 = 0$. A μ_σ can become positive only during Step 8, and there μ_σ is only made positive if the associated set $\mathbf{Y}_\sigma$ is the subset of original nodes corresponding to an outermost pseudonode. In Step 7 pseudonodes are unshrunk, but only if the corresponding pseudonode price μ_σ is 0. Hence the third property in (10.28) holds throughout the algorithm. The first property in (10.28) holds by the nature of the algorithm, and the second holds by the results in Theorem 10.17. Hence all the properties in (10.28) hold throughout the algorithm.

THEOREM 10.19 *Let* $\mathbf{M}$, (π, μ), $\mathrm{G}^1_*(\pi, \mu)$, *be the present matching in* G, *present dual feasible solution, present current equality subnetwork* *respectively, when the algorithm arrives at Step 8 at some stage. If* $\delta = +\infty$ *in that Step 8, there exists no perfect matching in* $\mathbf{G}$*, in fact the present* $\mathbf{M}$ *is a maximum cardinality matching in* G*.*

Proof Suppose $\delta = +\infty$ in that Step 8. So, all of $\delta_1, \delta_2, \delta_3$ are $+\infty$. $\delta_3 = +\infty$ implies that there are no current nodes which are inner labeled pseudonodes. So, all the inner labeled current nodes are original nodes at this stage, and all pseudonodes that are current nodes are either unlabeled or outer labeled. Define $(\pi(\lambda) = (\pi_i(\lambda)), \mu(\lambda) = (\mu_\sigma(\lambda)))$ by

$$\pi_i(\lambda) = \begin{cases} \pi_i + \lambda & \text{for all } i \text{ inside outer current nodes} \\ \pi_i - \lambda & \text{for all } i \text{ inside inner current nodes} \\ \pi_i & \text{for all } i \text{ inside unlabeled nodes} \end{cases}$$

$$\mu_\sigma(\lambda) = \begin{cases} \mu_\sigma + 2\lambda & \text{if } \mathbf{Y}_\sigma \text{ is the set of original nodes in} \\ & \text{a current outer labeled pseudonode} \\ \mu_\sigma & \text{otherwise} \end{cases}$$

The dual objective function $W(\pi, \mu)$ in (10.24) at this stage can be written as $\sum(f_p(\pi, \mu)$: over current nodes $p)$, where

$$\begin{aligned} f_p(\pi, \mu) = & \sum(\pi_i : \text{ over } i \text{ inside } p) - \sum((|\mathbf{Y}_\sigma| - 1)\mu_\sigma/2 : \text{ over } \sigma \\ & \text{s. t. } \mathbf{Y}_\sigma \text{ is set of original nodes inside a pseudonode} \\ & \text{within } p) \end{aligned}$$

If p is an unlabeled node it can be verified that $f_p(\pi(\lambda), \mu(\lambda)) = f_p(\pi, \mu)$ for all λ. Also, if p is an inner labeled node, it must be an original node as mentioned above, and hence $f_p(\pi(\lambda), \mu(\lambda)) = f_p(\pi, \mu) - \lambda$.

Suppose p is an outer labeled current node. If p is an original node, then $f_p(\pi(\lambda), \mu(\lambda)) = f_p(\pi, \mu) + \lambda$, because in this case $\pi_p(\lambda) = \pi_p + \lambda$. If p is a pseudonode, let $\mathbf{Y}_{\sigma_1}$ be the set of original nodes inside it. Let $\mathbf{S} = \{\sigma : \mathbf{Y}_\sigma \neq \mathbf{Y}_{\sigma_1}$ is the set of original nodes inside a pseudonode strictly within p $\}$. Then $\mu_\sigma(\lambda) = \mu_\sigma$ for all $\sigma \in \mathbf{S}$ because the associated pseudonode is not a current node. So

$$\begin{array}{rcl} f_p(\pi(\lambda), \mu(\lambda)) & = & \sum(\pi_i(\lambda) : \text{ over } i \text{ inside } p) \\ -(|\mathbf{Y}_{\sigma_1}| & - & 1)\mu_{\sigma_1}(\lambda)/2 - \sum((|\mathbf{Y}_\sigma| - 1)\mu_\sigma/2 : \text{over } \sigma \in \mathbf{S}) \\ & = & \sum(\pi_i : \text{ over } i \text{ inside } p) + |\mathbf{Y}_{\sigma_1}|\lambda \\ -(|\mathbf{Y}_{\sigma_1}| & - & 1)\mu_{\sigma_1} + 2\lambda)/2 - \sum((|\mathbf{Y}_\sigma| - 1)\mu_\sigma/2 : \text{over } \sigma \in \mathbf{S}) \\ & = & f_p(\pi, \mu) + \lambda \end{array}$$

So, if p is an outer labeled current node, whether p is an original or pseudonode, $f_p(\pi(\lambda), \mu(\lambda)) = f_p(\pi, \mu) + \lambda$.

All the nodes in a Hungarian tree in $\mathrm{G}^1_*(\pi, \mu)$ are matched nodes with the exception of the root node which is an outer labeled current unmatched node. Each current matching edge $(p; q)$ in a Hungarian tree contains one inner labeled node and one outer labeled node, and hence from the above facts we have $f_p(\pi(\lambda), \mu(\lambda)) + f_q(\pi(\lambda), \mu(\lambda)) = f_p(\pi, \mu) + f_q(\pi, \mu)$. If r is the root node of a Hungarian tree, since it is an outer labeled node we have $f_r(\pi(\lambda), \mu(\lambda)) = f_r(\pi, \mu) + \lambda$. Since matching edges are node disjoint, all the nonroot nodes in a Hungarian tree can be partitioned into pairs, each pair being the two nodes on a current matching edge in that tree. So, from the above, $\sum(f_p(\pi(\lambda), \mu(\lambda))$: over current nodes p in a Hungarian tree) $= \sum(f_p(\pi, \mu)$: over current nodes p in that Hungarian tree) $+ \lambda$. Summing over all the Hungarian trees in $\mathrm{G}^1_*(\pi, \mu)$ and over all the unlabeled current nodes, we have

$$W(\pi(\lambda), \mu(\lambda)) = W(\pi, \mu) + \lambda l$$

where l is the number of distinct Hungarian trees (same as the number of unmatched nodes at this stage). Since $\delta = +\infty$, $(\pi(\lambda), \mu(\lambda))$ remains dual feasible for all $\lambda \geqq 0$ by Theorem 10.17, and $W(\pi(\lambda), \mu(\lambda)) = W(\pi, \mu) + l\lambda \to +\infty$ as $\lambda \to +\infty$. Thus the dual problem (10.24) is feasible and the dual objective function is unbounded above on it. By the duality theorem of LP the primal problem (10.21) is infeasible in this case. We have discussed earlier that every perfect matching vector is feasible to (10.21), however, since (10.21) is infeasible in this case, there exists no perfect matching in G.

We will now provide an alternate proof of this theorem which does not need the duality theorem of LP. Let G^1 denote the current network consisting of all of the equality and nonequality edges. Suppose $\delta = +\infty$ in Step 8 at some stage. So, all of $\delta_1, \delta_2, \delta_3$ are $+\infty$. $\delta_3 = +\infty$ implies that there are no inner labeled current nodes in G^1 that are pseudonodes. So, at this stage all the current nodes that are pseudonodes are either outer labeled in-tree nodes or matched unlabeled nodes. Throw away (or dismember as discussed in Section 10.1.1) all the unlabeled pseudonodes and replace that part of the network with the corresponding original network, leaving the matching, nonmatching status of each edge unchanged. This changes the current network G^1 into $\overline{G}^1$ say. $\overline{G}^1$ contains all the alternating trees in G^1, but all the nodes in $\overline{G}^1$ outside the alternating trees are matched original nodes. The fact that δ_1, δ_2 are $+\infty$ implies that the present forest is a Hungarian forest in $\overline{G}^1$. Since $\overline{G}^1$ is a current network corresponding to G containing a Hungarian forest, this implies that the present matching (which is not a perfect matching) is a maximum cardinality matching in G. So, there exists no perfect matching in G in this case. ∎

Let $(\pi, \mu), (\hat{\pi}, \hat{\mu})$ be the dual feasible solutions at the beginning and end of a Step 8 in which δ is finite, and l the number of alternating trees in the Hungarian forest (or the number of unmatched nodes in G) at that stage. If $\mathbf{M}$ is the matching at that stage, $l = n - 2|\mathbf{M}|$. From the arguments in the proof of Theorem 10.19, we have $W(\hat{\pi}, \hat{\mu}) = W(\pi, \mu) + l\delta$. So the dual objective value strictly increases (by δ times the number of unmatched nodes at that stage) each time Step 8 is carried out.

Suppose the algorithm terminates in Step 10. Let $x, (\pi, \mu)$ be the perfect matching vector, dual feasible solution at that stage. $x, (\pi, \mu)$ together satisfy primal feasibility (10.21), dual feasibility (10.24), and the complementary slackness conditions for optimality (10.26), (10.27). So, by duality theory of LP, x is optimal to (10.21), and since it is a perfect matching vector, it is a minimum cost perfect matching vector in G.

We will now analyze the worst case computational complexity of this algorithm. Define an *iteration* in this algorithm to begin just after an augmentation step has been completed (the *first iteration* of course begins at the start), and end when the next augmentation is completed (the final iteration may also end with the infeasibility conclusion). Thus augmentation is carried out exactly once in each iteration, at the end of that iteration (if the problem is infeasible, no augmentation occurs in the final iteration). The algorithm goes through at most $(n/2)$ iterations.

Consider the ρth iteration in this algorithm. A newly formed blossom in this iteration is shrunk into a pseudonode, that is labeled as an outer node when it is formed. This pseudonode can become an unlabeled node only at the end of the iteration, only if it lies on the augmenting path discovered in this iteration. Until an augmentation step occurs after it is formed, this pseudonode either remains an outer current node, or gets absorbed inside another pseudonode which is an outer current node. So, by Theorem 10.6, the blossom shrinking step (Step 6) can occur at most $(n/2)$ times during this iteration.

Only pseudonodes which are inner current nodes are unshrunk during the algorithm. This implies that any pseudonode which is unshrunk in this ρth iteration must be a pseudonode formed in earlier iterations, which is an existing pseudonode at the beginning of this iteration. Again by Theorem 10.6, this implies that the pseudonode unshrinking step (Step 7) can occur at most $(n/2)$ times during this iteration.

Let $\mathbf{\Gamma}$ = set of all original nodes contained within outer labeled current nodes, $\mathbf{I}$ = set of inner labeled current nodes. So, if $i \in \mathbf{\Gamma}$, then i is not contained within any current node in $\mathbf{I}$. So, $|\mathbf{\Gamma}| + |\mathbf{I}| \leqq n$.

After each execution of Step 8 we either terminate (if $\delta = +\infty$), or δ is finite and equal to δ_1, δ_2, or δ_3.

If δ is finite and $= \delta_2$, at least one new simple blossom will be shrunk into a new outer labeled pseudonode when labeling is resumed. All the pseudonodes which were inner labeled nodes on this simple blossom before its shrinking, will be lost from the set $\mathbf{I}$ because of this shrinking, but each original node contained within any such pseudonode will get included in the set $\mathbf{\Gamma}$ after this blossom shrinking. So, $|\mathbf{\Gamma}| + |\mathbf{I}|$ either stays the same or increases in this shrinking step. Also, the maximum number of times that this blossom shrinking step can occur in this iteration is $(n/2)$. Hence, the maximum number of dual solution change steps in this iteration in which δ turns out to be finite and $= \delta_2$ is $(n/2)$.

If δ is finite and $= \delta_3$, at least one pseudonode which is an inner current node will be unshrunk when labeling is resumed. After the unshrinking this pseudonode will be lost from the set $\mathbf{I}$. However, at least 3 nodes on the simple blossom replacing this pseudonode will get labeled, at least 2 of them as inner nodes, and at least one as an outer node. Hence, after the unshrinking, $|\mathbf{I}|$ increases by at least 1, and $|\mathbf{\Gamma}|$ increases by at least 1, i.e., $|\mathbf{\Gamma}| + |\mathbf{I}|$ increases by at least 2.

If δ is finite and $= \delta_1$, at least one unlabeled current node gets labeled as an inner node immediately after this dual solution change step, as a result $|\mathbf{I}|$ increases by at least one, while $|\mathbf{\Gamma}|$ either stays the same (if no more nodes are labeled) or increases (if some more nodes are labeled). Hence, each dual solution change step in this iteration in which δ is finite and equal to δ_1 or δ_3 has the effect of increasing $|\mathbf{\Gamma}| + |\mathbf{I}|$ by at least 1, and hence the total number of such steps cannot exceed n in this iteration.

Hence the total number of dual solution change steps in this iteration cannot exceed $(3n/2)$.

The computational effort in a dual solution change step, a pseudonode unshrinking step, a blossom shrinking step, an augmentation step, or a tree growth step, is clearly bounded above by $\mathrm{O}(m)$. Hence the total computational effort in one iteration of this algorithm is bounded above by $\mathrm{O}(nm)$. Hence the overall computational effort in this algorithm is bounded above by $\mathrm{O}(n^{\mathbf{2}}m)$.

In a straightforward implementation of this algorithm, the values of $d_{ij}(\pi, \mu)$ will be revised for each $(i; j) \in \mathcal{A}$ after each dual solution change step, and this itself involves $\mathrm{O}(m)$ effort right away in this step, and makes the overall complexity of this algorithm $\mathrm{O}(n^{\mathbf{2}}m)$. As it was done for the Hungarian method for the bipartite matching problem in Section 3.1, it is possible to implement this algorithm so that for any $(i; j) \in \mathcal{A}$, $d_{ij}(\pi, \mu)$ is computed at most twice per iteration. Because of this, the improved implementation has an overall worst case computational complexity of $\mathrm{O}(n^{\mathbf{3}})$ (Lawler [1976 of Chapter 1]). We discuss this efficient implementation next.

An $\mathrm{O}(n^3)$ Implementation on the Blossom Algorithm for the Minimum Cost Perfect Matching Problem

As before, we define an *iteration* in this blossom algorithm to begin either at initialization, or just after an augmentation step (Step 5) is carried out, and to finish when the next augmentation step is carried out. So there are at most $(n/2)$ iterations in the algorithm. In this implementation, we will use a quantity denoted by $\alpha_t(\pi, \mu)$, and subsets of original edges $\mathbf{\Gamma}_t(\pi, \mu), \mathbf{\Delta}_{tv}(\pi, \mu)$, defined for each current node t and current edge $(t; v)$ wrt the present dual solution (π, μ) as below.

$$\begin{aligned}
\alpha_t(\pi, \mu) &= \text{Min.}\{c_{ij} - d_{ij}(\pi, \mu) : (i; j) \in \mathcal{A}, i \text{ inside } t, \\
&\qquad j \text{ inside an outer current node } \neq t\} \\
\mathbf{\Gamma}_t(\pi, \mu) &= \{(i; j) : (i; j) \in \mathcal{A} \text{ attains the min.} \\
&\qquad \text{in the definition of } \alpha_t(\pi, \mu)\} \\
\mathbf{\Delta}_{tv}(\pi, \mu) &= \{(i; j) : (i; j) \in \mathcal{A} \text{ attains the min.} \\
&\qquad \text{in min. } \{c_{pq} - d_{pq}(\pi, \mu) : (p; q) \in \mathcal{A}, p \in t, q \in v\}\}
\end{aligned}$$

We will now describe the work in one iteration and how it is executed. Let $\mathrm{G}^1 = (\mathcal{N}^1, \mathcal{A}^1), (\overline{\pi}, \overline{\mu})$ be the current network, and the dual feasible solution at the beginning of the iteration. Compute $d_{ij}(\overline{\pi}, \overline{\mu})$ for every $(i; j) \in \mathcal{A}$, and $\alpha_t(\overline{\pi}, \overline{\mu}), \mathbf{\Gamma}_t(\overline{\pi}, \overline{\mu})$ for all $t \in \mathcal{N}^1$, and $\mathbf{\Delta}_{tv}(\overline{\pi}, \overline{\mu})$ for all $(t; v) \in \mathcal{A}^1$. In subsequent steps in this iteration, we compute $d_{ij}(\pi, \mu)$ only for those original edges $(i; j)$ needed for updating $\alpha_t(\pi, \mu), \mathbf{\Gamma}_t(\pi, \mu), \mathbf{\Delta}_{tv}(\pi, \mu)$ at that stage; and we show that this needs to be done at most once for any edge.

At some stage during this iteration, let $\mathrm{G}^1 = (\mathcal{N}^1, \mathcal{A}^1), (\pi, \mu)$ be the current network, and dual feasible solution. We will now describe how the various tasks in the algorithm are carried out at this stage.

TO SCAN AN OUTER NODE For scanning an outer node v, look for unlabeled current nodes t such that $\alpha_t(\pi, \mu) = 0$ and $\mathbf{\Gamma}_t(\pi, \mu)$ contains an edge incident to an original node inside v; these are all the unlabeled nodes which can be labeled from v. If $\alpha_v(\pi, \mu) > 0$, labeling the above nodes is all you can do when scanning v. If, on the other hand, $\alpha_v(\pi, \mu) = 0$, look for current outer labeled nodes t containing inside it an original node j incident to an edge in $\mathbf{\Gamma}_v(\pi, \mu)$. If one of these outer nodes t has a root index different from that of v, the predecessor paths of t and v together with the current edge $(t; v)$ is an augmenting path, go to the augmentation step (Step 5) and after this step terminate this iteration and go to the next iteration. If no augmenting path is found, using one of the above nodes t with the same root index as v, a simple blossom is identified in the current equality subnetwork, go to the blossom shrinking step (Step 6).

UPDATING OF $\alpha_t(\pi, \mu), \mathbf{\Gamma}_t(\pi, \mu), \mathbf{\Delta}_{tv}(\pi, \mu)$ WHEN AN UNLABELED NODE IS LABELED AS AN INNER NODE If an unlabeled current node gets labeled as an inner node, there is no change in any of these quantities or sets. Continue.

UPDATING WHEN AN UNLABELED NODE IS LABELED AS AN OUTER NODE Suppose an unlabeled current node u becomes labeled as an outer node. For every other current node w such that $(u; w) \in \mathcal{A}^1$ do the following: Find $\beta_w = c_{ij} - d_{ij}(\pi, \mu)$ for some $(i; j) \in \mathbf{\Delta}_{uw}(\pi, \mu)$. If

$$\begin{array}{rcl}\alpha_w(\pi, \mu) & < & \beta_w, \text{ no change in } \alpha_w(\pi, \mu) \text{ or } \mathbf{\Gamma}_w(\pi, \mu) \\ \alpha_w(\pi, \mu) & = & \beta_w, \text{ replace } \mathbf{\Gamma}_w(\pi, \mu) \text{ by } \mathbf{\Gamma}_w(\pi, \mu) \cup \mathbf{\Delta}_{uw}(\pi, \mu) \\ \alpha_w(\pi, \mu) & > & \beta_w, \text{ change value of } \alpha_w(\pi, \mu) \text{ to } \beta_w \\ & & \quad \text{and replace } \mathbf{\Gamma}_w(\pi, \mu) \text{ by } \mathbf{\Delta}_{uw}(\pi, \mu)\end{array}$$

No change in the other quantities or sets.

UPDATING WHEN A BLOSSOM IS SHRUNK Suppose a new pseudonode, t_0 has just been formed. So, at this stage t_0 is an outer labeled node. Find out all current nodes t such that $(t; t_0)$ is a current edge. For each such t, do the following: let $\mathbf{D}$ be the set of nodes t' on the simple blossom corresponding to t_0 such that $(t; t')$ was a current edge before t_0 was formed. For each $t' \in \mathbf{D}$ let $\nu_{tt'} = c_{ij} - d_{ij}(\pi, \mu)$ for any $(i; j) \in \mathbf{\Delta}_{tt'}(\pi, \mu)$. Let $\beta_{tt_0} = \text{min. } \{\nu_{tt'} : t' \in \mathbf{D}\}$. Let $\mathbf{X}_t =$ union of $\mathbf{\Delta}_{tt'}(\pi, \mu)$ over $t' \in \mathbf{D}$ satisfying $\nu_{tt'} = \beta_{tt_0}$. If

$$\begin{array}{rcl}\alpha_t(\pi, \mu) & < & \beta_{tt_0}, \text{ no change in } \alpha_t(\pi, \mu) \text{ or } \mathbf{\Gamma}_t(\pi, \mu) \\ \alpha_t(\pi, \mu) & = & \beta_{tt_0}, \text{ replace } \mathbf{\Gamma}_t(\pi, \mu) \text{ by } \mathbf{\Gamma}_t(\pi, \mu) \cup \mathbf{X}_t \\ \alpha_t(\pi, \mu) & > & \beta_{tt_0}, \text{ change value of } \alpha_t(\pi, \mu) \text{ to } \beta_{tt_0} \\ & & \quad \text{and replace } \mathbf{\Gamma}_t(\pi, \mu) \text{ by } \mathbf{X}_t\end{array}$$

Also define

$$\begin{array}{rcl}\mathbf{\Delta}_{tt_0}(\pi,\mu) & = & \mathbf{X}_t \\ \alpha_{t_0}(\pi,\mu) & = & \text{min. } \{\beta_{tt_0} : \text{over outer current nodes} \\ & & t \text{ such that } (t;t_0) \text{ is a current edge } \} \\ \mathbf{\Gamma}_{t_0}(\pi,\mu) & = & \text{union of } \mathbf{X}_t \text{ over } t \text{ attaining the minimum} \\ & & \text{in the definition of } \alpha_{t_0}(\pi,\mu)\end{array}$$

This completes the updating in this case.

UPDATING WHEN A PSEUDONODE IS UNSHRUNK Suppose a pseudonode v has just been unshrunk. For each node t' on the simple blossom corresponding to v, and current edge $(t;t')$ incident at it after the unshrinking: compute $\alpha_{t'}(\pi,\mu), \mathbf{\Gamma}_{t'}(\pi,\mu), \mathbf{\Delta}_{tt'}(\pi,\mu)$ using their definitions. Let $\nu_{tt'} = c_{ij} - d_{ij}(\pi,\mu)$ for any $(i;j) \in \mathbf{\Delta}_{tt'}(\pi,\mu)$.

For each node t such that $(t;t')$ is a current edge after the unshrinking for some t' on the simple blossom corresponding to v do the following: Define β_t = min. $\{\nu_{tt'} : t'$ an outer labeled node on the simple blossom corresponding to v, and $(t;t')$ is a current edge after the unshrinking$\}$, $\mathbf{X}_t$ = union of $\mathbf{\Delta}_{tt'}(\pi,\mu)$ over all t' attaining the minimum in the definition of β_t. If

$$\begin{array}{rcl}\alpha_t(\pi,\mu) & < & \beta_t, \text{ no change in } \alpha_t(\pi,\mu) \text{ or } \mathbf{\Gamma}_t(\pi,\mu) \\ \alpha_t(\pi,\mu) & = & \beta_t, \text{ replace } \mathbf{\Gamma}_t(\pi,\mu) \text{ by } \mathbf{\Gamma}_t(\pi,\mu) \cup \mathbf{X}_t \\ \alpha_t(\pi,\mu) & > & \beta_t, \text{ change value of } \alpha_t(\pi,\mu) \text{ to } \beta_t \\ & & \qquad \text{and replace } \mathbf{\Gamma}_t(\pi,\mu) \text{ by } \mathbf{X}_t\end{array}$$

TO CARRY OUT A DUAL SOLUTION CHANGE STEP Let (π,μ) denote the present dual feasible solution. Clearly, in this step, δ_1 = min. $\{\alpha_t(\pi,\mu) : t$ an unlabeled current node at this stage$\}$, δ_2 = min. $\{\frac{1}{2}\alpha_t(\pi,\mu) : t$ an outer labeled current node at this stage$\}$, and δ_3, δ are the same as defined in Step 8 of the algorithm. If the new dual feasible solution is $(\hat{\pi},\hat{\mu})$, then

$$\alpha_t(\hat{\pi},\hat{\mu}) = \begin{cases} \alpha_t(\pi,\mu) - \delta & \text{if } t \text{ is an unlabeled current node} \\ \alpha_t(\pi,\mu) - 2\delta & \text{if } t \text{ is an outer labeled current node} \\ \alpha_t(\pi,\mu) & \text{if } t \text{ is an inner labeled current node} \end{cases}$$

and the sets $\mathbf{\Gamma}_t(\hat{\pi},\hat{\mu}) = \mathbf{\Gamma}_t(\pi,\mu), \mathbf{\Delta}_{tv}(\hat{\pi},\hat{\mu}) = \mathbf{\Delta}_{tv}(\pi,\mu)$.

If carried out this way, it can be verified that for each $(i;j) \in \mathcal{A}$, $d_{ij}(\pi,\mu)$ is computed at most twice during the entire iteration. We will now analyze the computational complexity of this implementation, and show that the effort per iteration is $\mathrm{O}(n^{\mathbf{2}})$. First, consider the computational effort involved in updating after blossom shrinkings during an iteration. Suppose there are s blossom shrinkings in this iteration. None of the resulting pseudonodes are unshrunk during this iteration. Let $g_1,\ldots,g_s$ be the number of nodes on the simple blossoms corresponding to blossoms, in the order in which they are discovered. When the first pseudonode, t_0 say, was formed in this iteration, the updating operations take at most $(n-g_1)g_1$ effort, since there are at most $(n-g_1)g_1$ edges of the form $(t;t')$ discussed above. After this first pseudonode was formed, the number of current nodes is at most $n-g_1+1$. So, the effort involved in updating when the second pseudonode is unshrunk in this iteration is at most $g_2(n-g_1+1-g_2)$. After this the number of current nodes is at most $(n-g_1+1-g_2+1) = (n-g_1-g_2+2)$. Continuing this argument, we see that the total effort in updating due to blossom shrinking operations in this iteration is at most

$$g_1(n-g_1) + g_2(n-g_1-g_2+1) + \ldots + g_s(n-g_1-\ldots-g_s+s-1)$$
$$\leqq n(g_1+g_2+\ldots+g_s) \leqq n^{\mathbf{2}}$$

Now consider the computational effort involved in updating after pseudonode unshrinking operations in this iteration. Let u be the number of pseudonodes unshrunk in this iteration, let $a_1,\ldots,a_u$ be the number of nodes on the simple blossoms corresponding to these pseudonodes in the order in which they are unshrunk. Let b_1 be the number of current nodes just before the first pseudonode is unshrunk in this iteration.

Updating after the first unshrinking operation clearly requires at most b_1a_1 effort. After this the number of current nodes is a_1+b_1-1. So, the number of current nodes just before the second unshrinking in this iteration is $\leqq a_1+b_1-1$, and by the same argument, the effort involved in updating after it is at most $a_2(a_1+b_1-1)$. Similarly, the number of current nodes just before the third unshrinking in this iteration is $\leqq (a_1 + a_2 + b_1 - 2)$. Continuing the argument in this way, we see that the total effort in updating due to pseudonode unshrinking operations in this iteration is at most

$$a_1 b_1 + a_2(a_1 + b_1 - 1) + a_3(a_1 + a_2 + b_1 - 2) + \ldots +$$
$$a_u(a_1 + \ldots + a_{u-1} + b_1 - (u-1))$$

But $b_1 + a_1 + \ldots + a_u - (u-1) \leqq n$. So, the above sum in reverse order is $\leqq (n - a_u)a_u + (n - a_u - a_{u-1} + 1)a_{u-1} + \ldots \leqq \mathrm{O}(n^{\mathbf{2}})$, as before.

The computation of $d_{ij}(\pi, \mu)$ for all $(i; j) \in \mathcal{A}$ at the beginning of this iteration takes $\mathrm{O}(m)$ effort. The total effort for tree growth, augmentation, etc., in this iteration is clearly $\leqq \mathrm{O}(n^{\mathbf{2}})$. Summing up, we see that the overall effort in this iteration is $\mathrm{O}(n^{\mathbf{2}})$. Since there are at most $(n/2)$ iterations, the overall effort in this implementation is at most $\mathrm{O}(n^{\mathbf{3}})$.

10.1.3 The Minimum Cost Matching Problem

Here we consider the problem of finding a minimum cost matching in the undirected network $\mathrm{G} = (\mathcal{N}, \mathcal{A}, c = (c_{ij}))$ with c as the vector of edge cost coefficients, $|\mathcal{N}| = n$, and $|\mathcal{A}| = m$, without any constraint on the matchings cardinality. This problem is the same as in (10.17), (10.18) except that the constraint $x(i) = 1$ is to be replaced by $x(i) \leqq 1$ for all $i \in \mathcal{N}$; exactly the same change has to be made in (10.21) to get the corresponding LP formulation replacing the integrality requirements on the variables by the same blossom constraints. This is the LP

$$\begin{array}{rrcl} \text{Minimize} & z(x) & = & \sum(c_{ij}x_{ij} : \text{ over } (i;j) \in \mathcal{A}) \\ \text{subject to} & x(i) & \leqq & 1, \text{ for all } i \in \mathcal{N} \\ & \mathbf{Y}_\sigma^-(x) & \leqq & (|\mathbf{Y}_\sigma| - 1)/2, \sigma = 1 \text{ to } L \\ & x_{ij} & \geqq & 0 \text{ for all } (i;j) \in \mathcal{A} \end{array} \qquad (10.29)$$

where $\mathbf{Y}_\sigma, \sigma = 1$ to L are all the subsets of $\mathcal{N}$ of odd cardinality $\geqq 3$. Define the dual solution $\pi = (\pi_i : i \in \mathcal{N}), \mu = (\mu_\sigma : \sigma = 1 \text{ to } L)$, and the quantity $\mu^-(i;j)$ for each $(i;j) \in \mathcal{A}$ as in (10.22), exactly as in Section 10.1.2. The dual problem is

$$\begin{array}{rrcl} \text{Maximize} & W(\pi,\mu) & = & -\sum_{i\in\mathcal{N}} \pi_i - \sum_{\sigma=1}^{L}(|\mathbf{Y}_\sigma|-1)(\mu_\sigma)/2 \\ \text{subject to} & d_{ij}(\pi,\mu) & \leqq & c_{ij}, \text{ for each } (i;j)\in\mathcal{A} \\ & \pi,\mu & \geqq & 0 \end{array} \qquad (10.30)$$

where

$$d_{ij}(\pi,\mu) = -\pi_i - \pi_j - \mu^-(i;j) \qquad (10.31)$$

This formula for $d_{ij}(\pi,\mu)$ is different from that in (10.23) of Section 10.1.2, because of the difference in the degree constraints here. Also, here π is restricted to be $\geqq 0$ for the same reason. Given a dual feasible solution (π,μ), the set of equality edges wrt it is $\mathcal{A}_*(\pi,\mu)$ defined below in (10.32), and $\mathrm{G}_*(\pi,\mu) = (\mathcal{N},\mathcal{A}_*(\pi,\mu))$ is the equality subnetwork wrt (π,μ). The complementary slackness conditions for optimality in this primal, dual pair of problems are (10.33), (10.34), (10.35) given below.

$$\mathcal{A}_*(\pi,\mu) = \{(i;j) : (i;j)\in\mathcal{A}, \text{ and } d_{ij}(\pi,\mu) = c_{ij}\} \qquad (10.32)$$

$$\begin{array}{rcll} x_{ij} & > & 0 \Rightarrow d_{ij}(\pi,\mu) = c_{ij}, \text{ for each } (i;j)\in\mathcal{A} & (10.33) \\ \mu_\sigma & > & 0 \Rightarrow \mathbf{Y}^-_\sigma(x) = (|\mathbf{Y}_\sigma|-1)/2, \ \sigma = 1\ldots L & (10.34) \\ \pi_i & > & 0 \Rightarrow x(i) = 1 & (10.35) \end{array}$$

The blossom algorithm of this section maintains x, (π,μ) satisfying dual feasibility (10.30), and (10.33), (10.34) throughout. In this problem, an unmatched node i does not violate primal feasibility, but if $\pi_i > 0$ for an unmatched node i, then the complementary slackness condition (10.35) is violated. For this reason, in this problem we classify unmatched nodes into two classes: an unmatched node i is said to be an

exposed node	if $\pi_i > 0$ in the present dual solution
nonexposed node	otherwise

Since nonexposed nodes do not violate any of the feasibility or optimality (complementary slackness) conditions, the algorithm tries to reduce the number of exposed nodes. This is done by alternating between two phases, just as in the blossom algorithm of Section 10.1.2.

In the **matching change phase**, the dual solution (π, μ) is held constant and the algorithm tries to find alternating paths wrt the present matching, with an exposed node at at least one end of it. This is done by growing an alternating forest with one alternating tree rooted at each exposed node in the equality subnetwork $\mathrm{G}_*(\pi, \mu)$. If an augmenting path, i.e., an alternating path between two exposed nodes, is found, we rematch using it, the trees containing nodes on that path are chopped down and the growth of the remaining trees, if any, is resumed. Each augmentation step reduces the number of exposed nodes by 2.

Even if an alternating path joining a matched node t associated with $\pi_t = 0$ beginning with the matching edge incident at it, to an exposed node i is found, we rematch using it. This operation does not change the cardinality of the matching, but converts i into a matched node and t into an unmatched but nonexposed node (since $\pi_t = 0$) and thus reduces the number of exposed nodes by 1.

When such matching changes are not possible any more and a maximum cardinality matching in $\mathrm{G}_*(\pi, \mu)$ is obtained, and there are still exposed nodes left, the algorithm moves to a **dual solution change step**. After this step, the process is repeated in the new equality subnetwork.

In all dual feasible solutions obtained during the algorithm, the values of all the original node prices π_i associated with exposed nodes i at that stage will be the same.

If $c \geqq 0$ the empty matching is clearly a minimum cost matching. So, we assume that $c_{ij} < 0$ for at least one $(i; j) \in \mathcal{A}$.

Suppose $x, (\pi, \mu)$ are the present solutions. The matching change phase of the algorithm stops now if the present current equality subnetwork $\mathrm{G}^1_*(\pi, \mu)$ contains a Hungarian forest with a Hungarian tree rooted at each exposed node. This state is characterized by the following conditions.

> No alternating tree can grow any further (i.e., the list of labeled and unscanned nodes is empty). There are no augmenting paths in the current equality subnetwork, no blos-

soms which can be shrunk, and no inner labeled pseudonodes associated with pseudonode price of $\mu_\sigma = 0$. And $\pi_i > 0$ for all original nodes i contained within outer labeled nodes (this guarantees that there are no alternating paths between an exposed node i and a matched node t with $\pi_t = 0$).

When all these conditions are satisfied, we say that the **labeling has become hungarian** , and the alternating forest has become a **Hungarian forest** . Then the algorithm moves to the dual solution change phase.

BLOSSOM ALGORITHM FOR THE MINIMUM COST MATCHING PROBLEM

Step 1 Initialization Define the initial dual feasible solution to be $(\pi^0 = (\pi^0_i), \mu^0 = (\mu^0_\sigma))$ where $\mu^0 = 0$ and $\pi^0_i = (1/2)\max.\{0, -c_{pq}$, for all $(p;q) \in \mathcal{A}\}$, for all $i \in \mathcal{N}$. Choose $x^0 = 0$ and the corresponding initial matching $\mathbf{M}_0 = \emptyset$.

Step 2 Rooting an Alternating Forest If there are no exposed nodes, go to Step 9. Otherwise root an alternating tree at each exposed node i, by labeling it with $(\emptyset, +, i)$. List now consists of all these root nodes.

Step 3 Same as Step 3 in the algorithm of Section 10.1.2

Step 4 Scanning Let the node to be scanned be the current node i with label (P(i), $\pm$, r).

Scanning an Outer Node If i is an outer node, for each $j \neq$ P(i) such that $(i;j)$ is a current equality edge do the following.

If j is an already labeled outer node associated with a root $\neq r$, an augmenting path joining two exposed nodes has been found, go to Step 5.

If j is an already labeled outer node with the same root r, and there is an original node t inside t such that $\pi_t = 0$, an alternating path joining t to the apex node of r, beginning with the matching edge incident at t has been found, go to Step 5.3.

If j is an already labeled outer node with the same root r, and $\pi_t > 0$ for all original nodes t inside j, the alternating tree containing i and j has blossomed, go to Step 6.

If j is an already labeled inner node, continue.

If j is unlabeled, label it with $(i, -, r)$ and include it in the list.

Scanning an Inner Node If i is an inner node, and it is an unmatched nonexposed node, go to Step 5.2 (Augmentation 2 step).

If i is a matched inner node, let $(i; j)$ be the current matching edge incident at it. If j is already labeled, it must be an inner node too. If the root indices of i and j are the same, the tree containing them has blossomed, go to Step 6. If the root indices of i, j are different, an augmenting path has been found, go to Step 5.

If j is unlabeled, label it with $(i, +, r)$ and include it in the list.

Go back to Step 3.

Step 5 Same as Step 5 in the algorithm of Section 10.1.2.

Step 5.2 Augmentation 2 Step We come to this step when scanning has revealed an inner labeled node i which is an unmatched nonexposed node. Let $\mathcal{P}^1$ be the predecessor path of i, it is an augmenting path containing one exposed and one nonexposed unmatched nodes. Find the corresponding augmenting path $\mathcal{P}$ in G between i and the apex of the root node, by expanding the pseudonodes on $\mathcal{P}^1$. Rematch using $\mathcal{P}$. Revise all the blossoms along $\mathcal{P}^1$, as discussed in Section 10.1.2. Chop down the tree containing i. If there are no exposed nodes, go to Step 9. Otherwise include all the outer labeled current nodes in the list and go to Step 3.

Step 5.3 Rematching an Alternating Path between an Exposed and a Matched Node with 0 Node Price We come to this step when scanning has revealed an outer node i joined to another outer node in the same tree by a current equality edge, such that j is matched and contains inside it an original node t with $\pi_t = 0$. Let the root of this tree be r. Let $\mathcal{P}^1$ be the path consisting of the edge $(j;i)$ and the predecessor path of i. By expanding the pseudonodes on $\mathcal{P}^1$, find the corresponding alternating path $\mathcal{P}$ in G beginning with the matching edge incident at t to the apex node of r. Rematch using $\mathcal{P}$. This makes t unmatched, but since $\pi_t = 0$, it remains nonexposed. The apex of r is now matched. Revise all the blossoms along $\mathcal{P}^1$ as discussed in Section 10.1.2. Chop down the tree containing r. If there are no exposed nodes, go to Step 9. Otherwise, include all the outer nodes in the list and go back to Step 3.

Steps 6, 7 Same as Steps 6, 7 in the algorithm of Section 10.1.2.

Step 8 Dual Solution Change Step We reach this step only if we still have exposed nodes in the present equality subnetwork and the Hungarian forest conditions are satisfied. This implies that the present matching is a maximum cardinality matching in the current equality subnetwork. Let (π, μ) be the present dual feasible solution. Compute the following using the convention that the minimum in the empty set is $+\infty$.

$$\begin{aligned}
\delta_1 &= \text{Min. } \{c_{ij} - d_{ij}(\pi,\mu) : (i;j) \in \mathcal{A}, i[j] \text{ is inside an outer [an unlabeled] current node}\} \\
\delta_2 &= \text{Min. } \{\tfrac{1}{2}(c_{ij} - d_{ij}(\pi,\mu)) : (i;j) \in \mathcal{A}, i \text{ and } j \text{ are inside distinct outer current nodes}\} \\
\delta_3 &= \text{Min. } \{\tfrac{1}{2}\mu_\sigma : \sigma \text{ s. t. } \mathbf{Y}_\sigma \text{ is the set of original nodes inside a current inner labeled pseudonode}\} \\
\delta_4 &= \text{Min. } \{\pi_i : i \text{ is inside an outer current node}\} \\
\delta &= \text{Min. } \{\delta_1, \delta_2, \delta_3, \delta_4\}
\end{aligned}$$

Define the new dual solution to be $\hat{\pi} = (\hat{\pi}_i), \hat{\mu} = (\hat{\mu}_\sigma)$ where

$$\hat{\pi}_i = \begin{cases} \pi_i - \delta & \text{for all } i \text{ inside outer current nodes} \\ \pi_i + \delta & \text{for all } i \text{ inside inner current nodes} \\ \pi_i & \text{for all } i \text{ inside unlabeled nodes} \end{cases}$$

$$\hat{\mu}_\sigma = \begin{cases} \mu_\sigma + 2\delta & \text{if } \mathbf{Y}_\sigma \text{ is the set of original nodes in} \\ & \text{a current outer labeled pseudonode} \\ \mu_\sigma - 2\delta & \text{if } \mathbf{Y}_\sigma \text{ is the set of original nodes in} \\ & \text{a current inner labeled pseudonode} \\ \mu_\sigma & \text{otherwise} \end{cases}$$

Find $\mathrm{G}_*(\hat{\pi}, \hat{\mu})$. If $\delta = \delta_4, \hat{\pi}_i = 0$ for all unmatched nodes i, go to Step 9. If $\delta < \delta_4$ but $= \delta_3$ go to Step 7. If $\delta < \delta_4$ and $< \delta_3$ include all outer current nodes in the list and go to Step 3.

Step 9 Optimality We only come to this step when we have a matching $\mathbf{M}$, the associated matching vector x, and dual feasible solution (π, μ), and there are no exposed nodes. $\mathbf{M}$ is a minimum cost matching in G. Terminate.

Discussion

By the manner in which scanning is carried out, it can be verified that if p is a root node, it cannot contain inside it an original node t with $\pi_t = 0$. So, whenever we come to Step 5.3, the node j will be a matched node (since it is an outer node and not a root node). Also, by this property and the Hungarian forest conditions, when the algorithm comes to Step 8, then δ_4 is finite and strictly positive. It can be verified that δ will always be finite and positive whenever Step 8 is carried out, and that (π, μ) remains dual feasible throughout the algorithm. Also, verify that all matching edges are all equality edges, and in-tree edges remain equality edges after each dual solution change step. The equality, nonequality status of any edge contained within a pseudonode, or any edge $(i; j) \in \mathcal{A}$ with i inside an outer node, and j inside an inner node, remains unchanged during a dual solution change step. The pair $x, (\pi, \mu)$ always satisfy (10.30), (10.33), (10.34).

By arguments similar to those in the proofs in Section 10.1.2, it can be shown that if $(\pi, \mu), (\hat{\pi}, \hat{\mu})$ are the present and new dual solutions in a Step 8, then $W(\hat{\pi}, \hat{\mu}) = W(\pi, \mu) + l\delta$, where l is the number of alternating trees in the current equality subnetwork at that stage (same as the number of exposed nodes at the beginning of this step). Thus each execution of Step 8 increases the dual objective value strictly.

Exposed nodes are always outer nodes. Every time Step 8 is carried out, the same δ is subtracted from the original node price of exposed nodes, while it may be subtracted or added to the original node price of other nodes. The initial node price is the same for all the nodes. Hence, at every stage, the original node prices of all the exposed nodes are the same, and their value is $\leqq$ the original node price of any other node.

Finally, when the algorithm arrives at Step 9, there are no exposed nodes, and the complementary slackness condition (10.35) is also satisfied. Since primal feasibility, dual feasibility, and the complementary slackness conditions for optimality are all satisfied at this stage, and the x-vector is an integer vector, it is a minimum cost matching vector in G.

Steps 5 or 5.2 or 5.3 need to be carried at most n times in the algorithm, since each of them reduces the number of exposed nodes by at least 1. Define an *iteration* in this algorithm to begin just after one of Steps 5, 5.2, or 5.3 is completed (the first iteration of course begins at the start) and end when one of these matching change steps is completed next. So, the algorithm goes through at most n iterations. Using arguments similar to those in Section 10.1.2, it can be verified that the computational effort in one iteration is bounded above by $O(nm)$. Hence, the overall computational effort in the algorithm is bounded above by $O(n^2 m)$, which can be reduced to $O(n^3)$ by implementing it efficiently as discussed in Section 10.1.2.

10.1.4 The Convex Hulls of Matching and Perfect Matching Vectors

Let $\mathbf{K}^{\leqq}, \mathbf{K}^{=}$ denote the convex hulls of matching, perfect matching vectors respectively in $G = (\mathcal{N}, \mathcal{A})$. Irrespective of what the edge cost vector c may be, we have shown in Section 10.1.3, that the LP (10.29) has an optimum solution that is a matching vector in G, and that the blossom algorithm discussed there will find it. This implies that all the extreme points of the set of feasible solutions of (10.29) are matching vectors in G. Also, clearly every matching vector in G is an extreme point of this set. Hence, $\mathbf{K}^{\leqq}$ is the set of feasible solutions of (10.29), i.e., (10.29) provides a linear constraint representation of $\mathbf{K}^{\leqq}$.

Using a similar argument, we conclude that (10.21) provides a linear constraint representation of $\mathbf{K}^{=}$.

Here we proved that systems (10.21), (10.29) provide linear constraint representations of $\mathbf{K}^{=}$, $\mathbf{K}^{\leqq}$ respectively, using the fact that the blossoms algorithms discussed earlier, based on these systems, find minimum cost extreme points of these polytopes, for all cost vectors c. Direct polyhedral proofs of these results without any recourse to algorithms, have been found recently, see Schrijver [1983].

10.1.5 Other Algorithms for Minimum Cost Matching Problems

An algorithm that begins with a perfect matching in G and finds a minimum cost perfect matching by moving through a sequence of perfect matchings of decreasing cost has been developed by Cunningham and Marsh [1978]. This algorithm has all the features of the primal simplex algorithm for solving LPs, and hence can be thought of as a primal algorithm for the minimum cost perfect matching problem. Its worst case computational complexity is also $O(n^2 m)$, but it is not expected to be computationally competitive with versions of the primal-dual blossom algorithms discussed above. Cunningham and Marsh [1978] also developed methods for doing sensitivity analysis in the minimum cost perfect matching problem using their primal algorithm. Another primal approach for the minimum cost perfect matching problem has been developed based on negative cost alternating cycles. This approach is also initiated with any perfect matching (with artificial edges, if necessary) and it improves this matching successively over negative cost alternating cycles, see Derigs [1986]. Another approach for the minimum cost perfect matching problem is to start with any matching ($\mathbf{M} = \emptyset$ for example) satisfying the complementary slackness optimality conditions with a dual feasible solution, and augment it successively using shortest augmenting paths until a perfect matching is obtained, see Derigs [1981, 1991].

10.1.6 Integer Valued Optimum Dual Solutions

Here we will prove that (10.30), the dual of the minimum cost matching problem in $\mathrm{G} = (\mathcal{N}, \mathcal{A}, c)$ has an integer optimum solution (π, μ) whenever c is an integer vector.

THEOREM 10.20 *If c is an integer vector, and $\tilde{x}, (\tilde{\pi}, \tilde{\mu})$ are the minimum cost matching vector, and optimum dual solution, obtained by the blossom algorithm of Section 10.1.3 when it is applied on* G*, then $\tilde{\mu}$ and $2\tilde{\pi}$ are integer vectors.*

Proof Assume that c is an integer vector. Let (π, μ) be a dual feasible solution obtained during the course of applying the blossom algorithm of Section 10.1.3 on G. We will actually show that (π, μ) satisfy the following properties.

1. μ and 2π are integer vectors.

2. Let $\mathbf{T}$ = set of all original nodes inside a current in-tree node. Then $\pi_i - \lfloor \pi_i \rfloor$ = the fractional part in π_i has the same value for all $i \in \mathbf{T}$, and this value is either 0 or 1/2.

By the definition of the initial dual feasible solution (π^0, μ^0) in Step 1, it is clear that both properties 1 and 2 hold for it.

Suppose properties 1 and 2 hold for a dual feasible solution (π, μ) obtained at some stage in the algorithm. Since μ is an integer vector, if $(i; j) \in \mathcal{A}$ is an equality edge at this stage such that i is inside a labeled node, and j is inside an unlabeled node, then $d_{ij}(\pi, \mu)$ can only be equal to c_{ij}, an integer, if $\pi_i - \lfloor \pi_i \rfloor = \pi_j - \lfloor \pi_j \rfloor$. This implies that property 2 continues to hold after tree growth steps occur. Properties 1 and 2 imply that in the ensuing dual solution change step, the fractional part in δ_2, if any, is 1/2, and that 2δ is integer. From the dual solution updating formula, these facts imply that properties 1, 2 continue to hold in the new dual feasible solution after a dual solution change step.

Using this repeatedly, we verify that properties 1, 2 hold throughout the algorithm. The theorem follows directly. ∎

THEOREM 10.21 *If $c = (c_{ij})$ is a vector in which all the entries are 0 or -1, (10.30) has an optimum solution $(\pi, \mu = 0)$ which is integer.*

Proof Let $c_{ij} = 0$ or -1 for all $(i;j) \in \mathcal{A}$. Let $\mathcal{A}_2 = \{(i;j) : c_{ij} = -1\}$. Finding a minimum cost matching in $\mathrm{G} = (\mathcal{N}, \mathcal{A}, c)$ in this case is the same problem as finding a maximum cardinality matching in $\mathrm{G}_2 = (\mathcal{N}, \mathcal{A}_2)$. Let $\mathbf{M}_2$ be a maximum cardinality matching in G_2, and let x^2 be the matching vector in G corresponding to the matching $\mathbf{M}_2$. Define the 0-1 vector $\pi^2 = (\pi^2_i)$ with π^2_i equal to 1 for exactly one node on each edge in $\mathbf{M}_2$, and 0 at all other nodes. Verify that $(\pi^2, \mu^2 = 0)$ is feasible to (10.30) in this case, and that $x^2, (\pi^2, \mu^2)$ satisfy the complementary slackness optimality conditions (10.33), (10.34), (10.35) for the primal, dual pair (10.29), (10.30) in this case. So, (π^2, μ^2) is an optimum solution for (10.30) in this case, and it is integer, proving the theorem. ∎

THEOREM 10.22 *Suppose c is an integer vector. Then (10.30) has an integer optimum solution (π, μ).*

Proof Let $(\tilde{\pi} = (\tilde{\pi}_i), \tilde{\mu})$ be the optimum dual solution obtained when the minimum cost matching problem in G is solved by the blossom algorithm of Section 10.1.3. By Theorem 10.20, $2\tilde{\pi}$ is an integer vector. If $\tilde{\pi}$ is an integer vector, we are done. So, assume that $\tilde{\pi}$ is not an integer vector. By Theorem 10.20, the fractional part of $\tilde{\pi}_i$ is either 0 or 1/2 for each $i \in \mathcal{N}$. Let $\mathbf{T} = \{i$: the fractional part of $\tilde{\pi}_i$ is $1/2\}$. For $(i;j) \in \mathcal{A}$ define $c'_{ij} = -1$ if both i and j are in $\mathbf{T}$, 0 otherwise. Let $(\pi', \mu' = 0)$ be an integer dual optimum solution for the minimum cost matching problem in $(\mathcal{N}, \mathcal{A}, c')$ constructed as in the proof of Theorem 10.21. Define $(\pi^* = (\pi^*_i = \lfloor \tilde{\pi}_i \rfloor + \pi'_i), \mu^* = \tilde{\mu})$. Clearly (π^*, μ^*) is an integer feasible solution of (10.30). Also

$$W(\pi^*, \mu^*) \geqq -\frac{1}{2}|\mathbf{T}| + (-\sum_{i \in \mathcal{N}} \lfloor \tilde{\pi}_i \rfloor - \sum_{\sigma=1}^{L} (|\mathbf{Y}_\sigma| - 1)\tilde{\mu}_\sigma / 2)$$

$$= -\sum_{i\in\mathcal{N}} \tilde{\pi}_i - \sum_{\sigma=1}^{L}(|\mathbf{Y}_\sigma| - 1)\tilde{\mu}_\sigma/2)$$
$$= W(\tilde{\pi}, \tilde{\mu})$$

Since, (π^*, μ^*) is feasible to (10.30) and $(\tilde{\pi}, \tilde{\mu})$ is optimal to it, $W(\pi^*, \mu^*) \geqq W(\tilde{\pi}, \tilde{\mu})$ implies that (π^*, μ^*) is an alternate optimum solution for (10.30). Also, (π^*, μ^*) is an integer vector, so it is an integer optimum solution of (10.30). ∎

Exercises

10.1 Prove that (10.24), the dual of the minimum cost perfect matching problem, always has an integer optimum solution if c is an integer vector, and it has an optimum solution. Develop an algorithm for finding such an integer optimum solution to this problem.

10.2 Show that the various matchings obtained during the successive iterations of the blossom algorithm for the minimum cost matching problem of Section 10.1.3, are minimum cost matchings among all matchings in G having the same cardinality as themselves.

10.3 Let ν be a positive integer. Show that the convex hull of all matching vectors in G whose cardinality is ν, is the set of feasible solutions of the system of constraints in (10.29) and the additional constraint $\sum(x_{ij}$: over $(i;j) \in \mathcal{A}) = \nu$.

Comment 10.2 The proof of the integer property in this section has been communicated to me by S. N. Kabadi. For a different proof of this result, see Cunningham and Marsh [1978].

10.2 1-Edge Covering Problems

Let G $= (\mathcal{N}, \mathcal{A})$ be an undirected connected network with $|\mathcal{N}| = n, |\mathcal{A}| = m$. In this section, we discuss algorithms for minimum cardinality and minimum cost edge covers in G. Define a **star** to be a tree

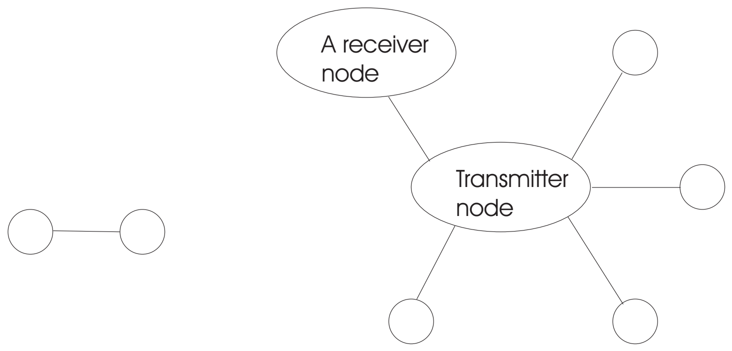

Figure 10.27: Two stars.

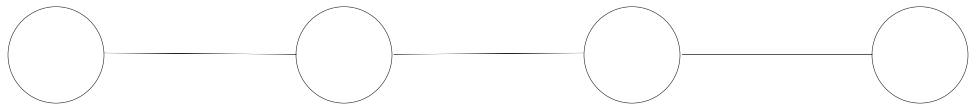

Figure 10.28: This network is not a star, as it has two nodes with degree $\geqq 2$.

spanning at least two nodes, in which at most one node has degree $\geqq 2$. See Figures 10.27, 10.28. In a star, if there is a node of degree $\geqq 2$, it is known as the **transmitter node**; and all other nodes are known as **receiver nodes**.

THEOREM 10.23 *Every minimum cardinality edge cover in* G *is a star forest.*

Proof Let **E** be a minimum cardinality edge cover in G. **E** cannot contain a cycle, because if it did, an edge from that cycle can be deleted, leading to an edge cover of lower cardinality, a contradiction. So **E** is a forest. If **E** is not a star forest, it must contain a path $\mathcal{P}$ consisting of three or more edges. On this path $\mathcal{P}$ there are two adjacent nodes both of which are of degree $\geqq 2$ in **E**. Deleting the edges joining these two nodes in $\mathcal{P}$, leads to an edge cover of lower cardinality than **E**, a contradiction. Hence, **E** is a star forest. ∎

THEOREM 10.24 **(i)** *Let* $\bar{\mathbf{M}}$ *be a maximum cardinality matching in* G*; and* $\bar{\mathbf{E}}$ *a subset containing all the edges in* $\bar{\mathbf{M}}$*, and one* *arbitrary edge incident at each unmatched node. Then* $\bar{\mathbf{E}}$ *is a minimum cardinality edge cover in* G.

(ii) *Let* $\hat{\mathbf{E}}$ *be a minimum cardinality edge cover in* G*; and* $\hat{\mathbf{M}}$ *a subset containing exactly one edge from each connected component of* $(\mathcal{N}, \hat{\mathbf{E}})$. *Then* $\hat{\mathbf{M}}$ *is a maximum cardinality matching in* G.

Proof Let $|\bar{\mathbf{M}}| = r$. So, the number of unmatched nodes in G wrt $\bar{\mathbf{M}}$ is $n - 2r$. So, by construction, $|\bar{\mathbf{E}}| = r + (n - 2r) = n - r = n - |\bar{\mathbf{M}}|$, and $\bar{\mathbf{E}}$ is an edge cover.

By hypothesis, $\hat{\mathbf{E}}$ is a minimum cardinality edge cover. So, by Theorem 10.23, $\hat{\mathbf{E}}$ is a star forest, hence each connected component of $(\mathcal{N}, \hat{\mathbf{E}})$ is a star tree. The subset $\hat{\mathbf{M}}$ defined in (ii) is clearly a matching and $|\hat{\mathbf{M}}| =$ number of connected components in $(\mathcal{N}, \hat{\mathbf{E}})$. Take a connected component of $(\mathcal{N}, \hat{\mathbf{E}}), (\mathbf{V}, \mathbf{T})$ say. So, $(\mathbf{V}, \mathbf{T})$ is a star tree. $\hat{\mathbf{M}}$ contains exactly one edge, e, say from $\mathbf{T}$. So, if $(\mathbf{V}, \mathbf{T})$ has a transmitter node, it must lie on e. So, the number of nodes in $(\mathbf{V}, \mathbf{T})$ not on e is exactly $|\mathbf{V}| - 2$, and $|\mathbf{T} \backslash \{e\}| = |\mathbf{V}| - 2$. Summing up the corresponding result over all the connected components of $(\mathcal{N}, \hat{\mathbf{E}})$ and using the above result leads to $|\hat{\mathbf{E}}| = |\hat{\mathbf{M}}| + (n - 2|\hat{\mathbf{M}}|) = n - |\hat{\mathbf{M}}| \geqq n - |\bar{\mathbf{M}}|$ (since $\bar{\mathbf{M}}$ is a maximum cardinality matching $|\bar{\mathbf{M}}| \geqq |\hat{\mathbf{M}}|) = |\bar{\mathbf{E}}|$, by previous results. Since $\hat{\mathbf{E}}$ is a minimum cardinality edge cover and $|\hat{\mathbf{E}}| \geqq |\bar{\mathbf{E}}|, \bar{\mathbf{E}}$ must also be a minimum cardinality edge cover. Also, from the above, we have $|\hat{\mathbf{M}}| \leqq |\bar{\mathbf{M}}|$, and since $\bar{\mathbf{M}}$ is a maximum cardinality matching, $\hat{\mathbf{M}}$ must also be a maximum cardinality matching in G. ∎

As an example, consider the network in Figure 10.18. $\bar{\mathbf{M}} = \{$(1; 2),(4; 7),(6; 10),(5; 12);(11; 13)$\}$ is a maximum cardinality matching in this network. Node 3 is the only unmatched node wrt $\bar{\mathbf{M}}$. So $\bar{\mathbf{M}} \cup \{(3; 2)\}$ is a minimum cardinality edge cover in this network.

Thus a minimum cardinality edge cover in G can be found by finding a maximum cardinality matching in G first, and then using the procedure described in (i) of Theorem 10.24. The computational effort required by this method is bounded above by $0(n^{\mathbf{3}})$.

A minimum cardinality edge cover in G can also be found directly by starting with some edge cover (for example, $\mathcal{A}$ can be used as the initial edge cover in G), generating **reducing paths** (these are analogous to augmenting paths in the maximum cardinality matching problem) and using them to move to edge covers of lower cardinality, until an edge cover wrt which no reducing path exists, is obtained. See White [1971] and White and Gillenson [1975].

10.2.1 Minimum Cost Edge Covers

In this section we discuss an algorithm for finding a minimum cost edge cover in G $= (\mathcal{N}, \mathcal{A}, c = (c_{ij}))$. Defining an edge covering vector $x = (x_{ij})$ in G as at the beginning of this chapter, this problem is the integer program

$$\begin{aligned} \text{Minimize } \ z(x) &= \sum(c_{ij}x_{ij} : \text{ over } (i,j) \in \mathcal{A}) \\ \text{subject to } \ x(i) &\geqq 1 \text{ for all } i \in \mathcal{N} \\ 0 \leqq x_{ij} &\leqq 1, \text{ for all } (i,j) \in \mathcal{A} \end{aligned} \qquad (10.36)$$

$$x_{ij} \text{ integer} \qquad \text{for all } (i,j) \in \mathcal{A} \qquad (10.37)$$

Let $\mathbf{Y} \subset \mathcal{N}$ with $|\mathbf{Y}|$ odd and $\geqq 3$. Define

$$\mathbf{Y}^+(x) = \sum(x_{ij} \text{ over } (i,j) \in \mathcal{A} \text{ with at least one of } i \text{ or } j \text{ or both in } \mathbf{Y}) \qquad (10.38)$$

Notice the difference in the definition of $\mathbf{Y}^+(x)$ in (10.38) compared to that of $\mathbf{Y}^-(x)$ in (10.19). If x is an edge covering vector in G, and $|\mathbf{Y}|$ is odd and $\geqq 3$, we must have

$$\mathbf{Y}^+(x) \geqq (|\mathbf{Y}| + 1)/2. \qquad (10.39)$$

(10.39) is the **covering blossom inequality** or the **covering blossom constraint** corresponding to $\mathbf{Y}$ for the edge covering problem. Also, observe the difference in the matching blossom constraint defined in (10.20) and the covering blossom constraint. Let $\mathbf{Y}_1, \ldots, \mathbf{Y}_L$ be all the distinct subsets of $\mathcal{N}$ of odd cardinality $\geqq 3$. Consider the following LP

$$\begin{array}{rcll} \text{Minimize} \;\; z(x) & = & \sum(c_{ij}x_{ij} : \text{ over } (i,j) \in \mathcal{A}) & \\ \text{subject to} \;\; x(i) & \geqq & 1 \text{ for all } i \in \mathcal{N} & (10.40) \\ \mathbf{Y}^+_\sigma(x) & \geqq & (|\mathbf{Y}_\sigma| + 1)/2, \;\; \sigma = 1 \text{ to } L & \\ 0 \leqq x_{ij} \leqq 1, & & \text{for all } (i,j) \in \mathcal{A} & \end{array}$$

Every edge covering vector in G is feasible to (10.40) and every integer feasible solution of (10.40) is an edge covering vector in G. So, if an optimum solution of (10.40) is an integer vector, it is a minimum cost edge covering vector in G. Associate the original node price (dual variable) π_i to the constraint corresponding to node i in (10.40), for each $i \in \mathcal{N}$, and the dual variable (which can be interpreted as the pseudonode price just as in Sections 10.1.2, 10.1.3) μ_σ to the blossom inequality corresponding to $\mathbf{Y}_\sigma, \sigma = 1$ to L. Let $\pi = (\pi_i), \mu = (\mu_\sigma)$. Given (π, μ), for each $(i,j) \in \mathcal{A}$, define

$$\begin{array}{rcll} \mu^+(i,j) & = & \sum(\mu_\sigma : \text{ over } \;\sigma \text{ s.t. } \mathbf{Y}_\sigma \text{ contains} & \\ & & \text{either } \; i \text{ or } \; j \text{ or } \; \text{both}) & (10.41) \end{array}$$

$$d_{ij}(\pi, \mu) \;=\; \pi_i + \pi_j + \mu^+(i,j) \qquad (10.42)$$

The dual of (10.40) is

$$\begin{array}{rcll} \text{Maximize} \;\; \sum\limits_{i \in \mathcal{N}} \pi_i & + & \sum\limits_{\sigma=1}^{L} (|\mathbf{Y}_\sigma| + 1)(\mu_\sigma/2) - \sum\limits_{(i,j) \in \mathcal{A}} \xi_{ij} & \\ \text{subject to} \;\; d_{ij}(\pi, \mu) & - & \xi_{ij} + \nu_{ij} = c_{ij}, \text{ for } (i,j) \in \mathcal{A} & (10.43) \\ & & \text{all } \;\; \pi_i, \mu_\sigma, \xi_{ij}, \nu_{ij} \geqq 0. & \end{array}$$

Hence the dual feasibility conditions for $(\pi = (\pi_i), \mu = (\mu_\sigma))$ are

$$\pi \geqq 0, \mu \geqq 0. \qquad (10.44)$$

The complementary slackness conditions for optimality in the bounded variable primal LP (10.40) and its dual (10.43) are: for each $i \in \mathcal{N}$ and $\sigma = 1$ to L:

$$\begin{aligned}
\pi_i > 0 &\Rightarrow x(i) = 1 && (10.45)\\
x(i) > 1 &\Rightarrow \pi_i = 0 && (10.46)\\
d_{ij}(\pi, \mu) > c_{ij} &\Rightarrow x_{ij} = 1 && (10.47)\\
d_{ij}(\pi, \mu) < c_{ij} &\Rightarrow x_{ij} = 0 && (10.48)\\
\mu_\sigma > 0 &\Rightarrow \mathbf{Y}^+_\sigma(x) = (|\mathbf{Y}_\sigma| + 1)/2 && (10.49)\\
\mathbf{Y}^+_\sigma(x) > ((|\mathbf{Y}_\sigma| + 1)/2) &\Rightarrow \mu_\sigma = 0. && (10.50)
\end{aligned}$$

The algorithm for the edge covering problem discussed below, is based on the blossom algorithm for matching problems. At termination, it generates an edge covering vector, $\bar{x}$, and dual feasible solution, $(\bar{\pi}, \bar{\mu})$ which together satisfy (10.45) to (10.50). Hence $\bar{x}$ is optimal to (10.40) and since it is an integer vector, it is a minimum cost edge covering vector.

The algorithm maintains an $\mathbf{E} \subset \mathcal{A}$ called the **solution set of edges**, and the present edge vector x will always be $x^{\mathbf{E}}$. The vector x is also called the *present solution vector.* The solution set of edges $\mathbf{E}$ is partitioned into $\mathbf{M}$ and $\mathbf{A}$. The set $\mathbf{M}$ will always be a matching in G, and $\mathbf{A}$ is called the *set of covering edges.* Throughout the algorithm $\mathbf{M} \cap \mathbf{A}$ will be $\emptyset$; and the set of nodes on edges in $\mathbf{M}$, and the set of nodes on edges in $\mathbf{A}$, will be mutually disjoint.

Original nodes in $\mathcal{N}$ are classified as below during the algorithm. Since $\mathbf{M}$ and $\mathbf{A}$ are always edge and node disjoint, every node is uniquely classifed by this classification.

matched nodes	-	those incident with a matching edge
type 1 nodes	-	those incident with exactly on covering edge
type 2 nodes	-	those incident with two or more covering edges
exposed nodes	-	those incident with no edge from $\mathbf{E} = \mathbf{M} \cup \mathbf{A}$.

Given the dual feasible solution (π, μ), define $d_{ij}(\pi, \mu)$ as in (10. 42) and let

$$\mathcal{A}_*(\pi,\mu) = \{(i,j) : (i,j) \in \mathcal{A}, \text{ and } d_{ij}(\pi,\mu) = c_{ij}\} \tag{10.51}$$

Edges in $\mathcal{A}_*(\pi,\mu)$ are the **original equality edges** in this problem and $\mathrm{G}_*(\pi,\mu) = (\mathcal{N}, \mathcal{A}_*(\pi,\mu))$ is the **equality subnetwork**, wrt the dual feasible solution (π,μ). In the algorithm, the set $\mathbf{M}$ will always be a matching which is a subset of $\mathcal{A}_*(\pi,\mu)$. It tries to obtain matchings of higher cardinality in $G_*(\pi,\mu)$ by growing alternating trees wrt $\mathbf{M}$, rooted at exposed nodes, in it. The alternating trees are grown with the aim of discovering augmenting paths, which are special types of alternating paths that can be used to augment either the matching $\mathbf{M}$, or the set of solution edges $\mathbf{E} = \mathbf{M} \cup \mathbf{A}$.

While growing the alternating trees, simple blossoms may be detected and shrunk into pseudonodes. After some pseudonodes are found, the original network G gets transformed into the current network $\mathrm{G}^1 = (\mathcal{N}^1, \mathcal{A}^1)$. $\mathcal{N}^1$ is the set of current nodes at the present stage, these are either the original nodes not contained in any pseudonode, or outermost pseudonodes. If $\mathbf{F} \subset \mathcal{A}$ is a subset of original edges, the set of current edges $\mathbf{F}^1$ corresponding to it is always defined to be

$$\mathbf{F}^1 = \{(p;q) : p,q \in \mathcal{N}^1, \text{ there exists } i \in p, j \in q, \text{ s.th. } (i;j) \in \mathbf{F}\}. \tag{10.52}$$

The sets of current edges corresponding to $\mathcal{A}_*(\pi,\mu), \mathbf{M}, \mathbf{A}, \mathbf{E} = \mathbf{M} \cup \mathbf{A}$ will always be denoted by $\mathcal{A}^1_*(\pi,\mu), \mathbf{M}^1, \mathbf{A}^1, \mathbf{E}^1 = \mathbf{M}^1 \cup \mathbf{A}^1$. Edges in $\mathcal{A}^1_*(\pi,\mu), \mathbf{M}^1, \mathbf{A}^1, \mathbf{E}^1$ are respectively **current equality edges, current matching edges, current covering edges**, and **current solution edges**.

Simple blossoms in this algorithm are partial networks of the current network satisfying the following properties.

It is a partial network of the current network determined by an odd subset with $\geqq 3$ current nodes.

There exists a unique current node in it which is incident with either exactly 2 or 0 current solution edges within it, this node is called the **base node** of the simple blossom. (10.53)

There exists a simple spanning cycle containing all the current nodes in it, which is an alternating cycle. All solution edges on it other than those incident at the base node, are current matching edges. If a_0 is the base node, and a_1, a_2 are the current nodes adjacent to a_0 on the odd cycle; deleting $(a_0; a_1), (a_0; a_2)$ from the cycle, leaves an AP wrt $\mathbf{M}^1$ (in the sense that edges on it are alternately matching and nonsolution edges). $(a_0; a_1)$ $(a_0; a_2)$ may be nonsolution edges or covering edges. Consequently, every node in the simple blossom other than the base node is incident with exactly one solution edge on the odd alternating cycle in it.

When they are detected, simple blossoms are shrunk into pseudonodes in the algorithm. The **base node** for a pseudonode is the base node on the simple blossom corresponding to it. The **apex node** of a simple blossom or the pseudonode into which it is shrunk, is the original node inside it which is incident exactly to either 2 or 0 solution edges within the simple blossom. For convenience we define the base node and apex node for an original node to be that node itself. Hence the apex node for a pseudonode is the apex node of its base node.

Simple blossoms and the pseudonodes into which they are shrunk are classified in accordance with their configuration. To distinguish between original nodes and pseudonodes we use roman letters to enumerate types of pseudonodes (type A, B, etc.). A pseudonode is:

rooted	-	if the base node is an exposed node (which will be a root node). Correspondingly, the apex of this pseudonode is also an exposed original node.
type A	-	if the base node is a matched node on a matching edge joining the base to a node outside the simple blossom.

type B	-	if the base node is incident to exactly one covering edge joining the base node to a node outside the simple blossom.
type C	-	if the base node is a type 2 original node, incident to exactly two covering edges which are within the simple blossom. The two edges incident at the base node on the odd alternating cycle in the simple blossom, are these covering edges.
type D	-	if the base node is itself a pseudonode which is either a type C or another type D node.

Thus all the edges joining a node in a simple blossom to a node outside are nonsolution edges, with the exception of the unique matching or covering edge incident at the base node in type A or B simple blossoms respectively. See Figure 10.29. The base node of a type A (type B) pseudonode is the unique node on its simple blossom incident to the matching (covering) edge joining it to a node outside the simple blossom. The base node of a type C simple blossom is the unique type 2 original node on the two covering edges inside it. The base node of a type D simple blossom is the unique type C or D pseudonode inside it. The apex node of the various types of pseudonodes are:

type A (type B)	-	it is the unique original node inside it joined by an original matching (covering) edge to a node outside the pseudonode
type C or D	-	it is the unique type 2 original node inside it

A new simple blossom detected during the tree growth process (while scanning some node) will always be either a rooted or a type A simple blossom. Type B, C, D pseudonodes are only created by transformation of an existing rooted or type A pseudonode in a matching and covering augmentation procedure in the algorithm.

If $\mathcal{N}_1$ is the set of original nodes corresponding to an existing pseudonode, since $|\mathcal{N}|$ is odd and $\geqq 3$, there is a covering blossom constraint corresponding to it. If x is the present solution vector, it can be verified that x satisfies the covering blossom constraint corresponding to this pseudonode as an equation if the pseudonode is a type A or

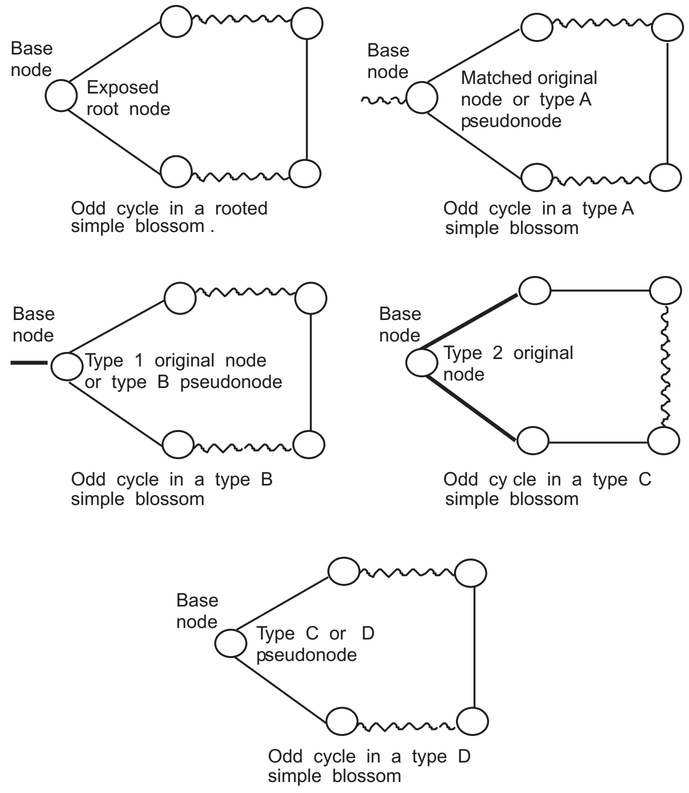
Base node
Exposed root node
Odd cycle in a rooted simple blossom .
Base node
Matched original node or type A pseudonode
Odd cycle in a type A simple blossom
Base node
Type 1 original node or type B pseudonode
Odd cycle in a type B simple blossom
Base node
Type 2 original node
Odd cy cle in a type C simple blossom
Base node
Type C or D pseudonode
Odd cycle in a type D simple blossom

Figure 10.29: Odd cycles in various types of simple blossoms. Wavy (thick) edges are matching (covering) edges.

B or C or D pseudonode, and violates it if it is a rooted pseudonode. Rooted pseudonodes may exist in intermediate stages of the algorithm but they are eliminated before termination. If any blossoms remain at termination of the algorithm, they will be of types A, B, C or D. Let

$$\mathcal{A}^- = \{(i;j) : (i;j) \in \mathcal{A}, c_{ij} \leqq 0\} \tag{10.54}$$

A minimum cost edge cover contains $\mathcal{A}^-$, so the algorithm maintains $\mathcal{A}^- \subset \mathbf{E}$ always. Solution edges $\mathbf{E}$ are classified into matching and covering edges as follows: matching edges ($\mathbf{M}$) are the edges in the components of $(\mathcal{N}, \mathbf{E})$ consisting of single edges which are not in $\mathcal{A}^-$; the remaining edges of $\mathbf{E}$ (i.e., all edges of $\mathcal{A}^-$, and all edges of $\mathbf{E}$ which are adjacent to at least one other edge in $\mathbf{E}$) are the covering edges.

The algorithm plants a rooted alternating forest with roots (each labeled as an outer node) at each exposed node and grows them. The objective of growing the alternating trees is to detect augmenting paths, which are special APs of solution/nonsolution equality edges that allow augmentations. Every time an augmentation is performed, at least one exposed node becomes nonexposed. Once a node becomes nonexposed it remains nonexposed in the sequel. When all nodes become nonexposed, the solution set of edges at that time will be a minimum cost edge cover, and the algorithm terminates. The following additional properties are always satisfied during the algorithm.

Properties Maintained by the Algorithm

π, μ always satisfy dual feasibility conditions (10.44).
$\mathbf{E} \subset \mathcal{A}^- \cup \mathcal{A}_*(\pi, \mu)$, $\mathbf{M} \subset \mathcal{A}_*(\pi, \mu)$, $\mathbf{A} \supset \mathcal{A}^-$.
$\mathcal{A}^- \subset \mathbf{A}^1 \subset \mathcal{A}^1_*(\pi, \mu) \cup \mathcal{A}^-$.
If $(\mathcal{N}, \mathbf{A})$ or $(\mathcal{N}^1, \mathbf{A}^1)$ has a connected component that is a single edge, that edge is in $\mathcal{A}^-$.
$\pi_i = 0$ if i is a type 2 node or a node on an edge in $\mathcal{A}^-$.
$\mu_\sigma > 0$ implies that $\mathbf{Y}_\sigma$ is the set of original nodes inside an existing pseudonode.
$d_{ij}(\pi, \mu) \overset{\leq}{=} c_{ij}$ for all $(i;j) \in \mathcal{A} \backslash \mathcal{A}^-$.
Each in-tree edge is in $\mathcal{A}^1_*(\pi, \mu) \backslash \mathbf{A}^1$.
If node i is an inner labeled current original node, $\pi_i > 0$.
Each in-tree current node that is a pseudonode is either rooted, or a type A labeled outer, or a type A labeled inner associated with pseudonode price $\mu_\sigma > 0$.
Every edge on the odd cycle in the simple blossom corresponding to any pseudonode at any level is an equality edge.
Nodes on edges in $\mathcal{A}^-$ are always current nodes, i.e., these nodes will never be in any pseudonode.

At some stage of the algorithm, if **E** is not yet an edge covering in G, no matching and covering augmentations are possible, no blossom shrinking or pseudonode unshrinking steps are possible, and the list of labeled and unscanned nodes is empty (i.e., no tree growth is possible), the labeling at that stage is said to have **become hungarian**, and the alternating trees at that stage are **Hungarian trees** in the current equality subnetwork. The trees constitute a **Hungarian forest** then, and the following conditions will hold. When these conditions are satisfied, the algorithm moves to the dual solution change step in the algorithm.

Hungarian Forest Conditions

Each inner labeled current node is incident to exactly one matching and one nonmatching current in-tree edge.
No node on an edge in $\mathbf{A}$ is in-tree or contained inside an in-tree pseudonode.
If an original node i is inside an outer labeled pseudonode, $\pi_i > 0$.
If $\mathbf{Y}_\sigma$ is the set of original nodes in an inner labeled pseudonode, $\mu_\sigma > 0$.

In the following algorithm the tree growth step is constructed in such a way that only outer nodes are scanned. The list always refers to labeled and unscanned outer current nodes at that stage.

BLOSSOM ALGORITHM FOR THE MINIMUM COST EDGE COVERING PROBLEM

Step 1 Initialization Let $\mathbf{M} = \emptyset, \mathbf{A} = \mathcal{A}^-, \pi = (\pi_i) = 0, \mu = (\mu_\sigma) = 0$ initially. If there are no exposed nodes, go to Step 13. Otherwise root an alternating tree at each exposed node i by labeling it with $(\emptyset, +, i)$. List = set of all these root nodes now.

Step 2 Select a Node to Scan If list = $\emptyset$ go to Step 12. Otherwise, select a node from it to scan, and delete it from the list.

Step 3 Scanning Let the node to be scanned be p with label $(\mathrm{P}(p), +, r)$. Do the following in the order given.

If there is an already labeled outer node j associated with a root node $s \neq r$ such that $(p; j) \in \mathcal{A}^1_*(\pi, \mu)$, an augmenting path has been found, go to Step 4.

If there is an already labeled outer node j associated with the same root node r such that $(p;j) \in \mathcal{A}^1_*(\pi,\mu)$, the alternating tree containing p, j has blossomed, go to Step 10.

If there is an original current node u such that $(p;u) \in \mathcal{A}^1_*(\pi,\mu)$ and $\pi_u = 0$, go to Step 5.

If there is an original current type 1 node or a type B pseudonode v such that $(p;v) \in \mathcal{A}^1_*(\pi,\mu)$, go to Step 6.

If there is an unlabeled current node w that is a type C or D pseudonode such that $(p;w) \in \mathcal{A}^1_*(\pi,\mu)$, go to Step 7.

Identify all $j \neq \mathrm{P}(p)$ such that $(p;j) \in \mathcal{A}^1_*(\pi,\mu)$. Since we came to this stage, j cannot be type 1, or 2, or B, C, or D, or exposed, or a rooted pseudonode (in that case we would have already gone to some other step). For each such j do the following: If j is unlabeled, let $(j;t)$ be the current matching edge incident at j, label j with $(p,-,r)$. If t is unlabeled and is a pseudonode inside which there is an original node i associated with node price $\pi_i = 0$ go to Step 8. Otherwise label t with $(j,+,r)$ and include it in the list.

Now p is labeled and scanned. Go to Step 11.

Step 4 Matching Augmentation We come to this step when scanning or inspection has revealed a pair of current nodes p and j associated with root indices $r(p) \neq r(j)$, contained on an augmenting path $\mathcal{P}^1$ which is obtained by combining the predecessor paths of p and j with the edge $(p;j)$. Let t_1, t_2 be the apex nodes of $r(p), r(j)$. Find the augmenting path $\mathcal{P}$ between t_1 and t_2 by expanding the pseudonodes on $\mathcal{P}^1$. Rematch using $\mathcal{P}$. Revise all the blossoms along $\mathcal{P}^1$ as in Section 10.1.2. Chop down the two trees rooted at $r(p)$ and $r(j)$. If there are no exposed nodes left, go to Step 13. Otherwise change the list into the set of all outer nodes and go back to inspection in Step 12, or Step 2, depending on whether you arrived here from inspection or Step 3.

Step 5 Matching and Covering Augmentation Procedure 1 We come to this step when scanning or inspection has revealed an outer current node p and an original current node u associated with $\pi_u = 0$, such that $(p; u) \in \mathcal{A}^1_*(\pi, \mu)$. (p, u) must be a non-matching edge. Let the label on p be $(\mathrm{P}(p), +, r)$. Let j be an original node inside p such that $(j; u) \in \mathcal{A}_*(\pi, \mu)$. Let $\mathcal{P}^1$ be the predecessor path of p. Find $\mathcal{P}$, the alternating path from j to the apex node of r, beginning with the matching edge incident at j if j is a matched node ($\mathcal{P}$ may have no edges if for example $p = r = j$) by expanding the pseudonodes along $\mathcal{P}^1$. Rematch using $\mathcal{P}$. Revise all the blossoms along $\mathcal{P}^1$ as discussed in Section 10.1.2. Chop down the tree rooted at r. If u is a type 1 or 2 node, add the edge $(j; u)$ to the set $\mathbf{A}$. If u is a matched node, let $(u; w) \in \mathbf{M}$, delete $(u; w)$ from $\mathbf{M}$, and add both $(j; u)$ and $(u; w)$ to the set $\mathbf{A}$. So, u becomes a type 2 node as a result of these changes. p becomes a type 1 node if it is an original node; or a type B pseudonode with j as its apex node, if it is a pseudonode.

Go to Step 13 if no exposed nodes are left. Otherwise change the list into the set of all outer nodes at this stage and go back to inspection in Step 12, or Step 2, depending on whether you arrived at this step from inspection or step 3.

Step 6 Matching and Covering Augmentation Procedure 2 We come to this step when scanning has revealed an outer current node p and current type 1 or type B node v such that $(p; v) \in \mathcal{A}^1_*(\pi, \mu)$. Let h be the apex node of v. If v is a type 1 node, π_v must be > 0 (otherwise, by the order in which scanning is done we would have gone to Step 5 instead of coming here) and hence v is not on any edge in $\mathcal{A}^-$. If v is a type B pseudonode, since its apex h is inside a pseudonode, h is not on any edge in $\mathcal{A}^-$. Let $(h; j_1)$ be the unique covering edge incident at h. Since p is an in-tree node, $j_1 \neq p$. So, the connected component of $(\mathcal{N}, \mathbf{A})$ containing $(h; j_1)$ must contain some edge other than $(h; j_1)$, this other edge cannot be incident at h since h is a type 1 node, let $(j_1; j_2)$ be one such edge. So, j_1 is a type 2 node.

Let $\mathcal{P}^1$ be the path consisting of the edge $(v; p)$ and then the predecessor path of p. By expanding the pseudonodes along $\mathcal{P}^1$, obtain the alternating path $\mathcal{P}$ from h to the apex of the root node r corresponding to p. If $(j_1; j_2) \in \mathcal{A}^-$ or if the connected component of $(\mathcal{N}, \mathbf{A})$ that contains $(j_1; j_2)$ has $\geqq 3$ edges, delete $(h; j_1)$ from $\mathbf{A}$, and then rematch using $\mathcal{P}$. On the other hand, if both $(h; j_1)$ and $(j_1; j_2) \in \mathbf{A} \backslash \mathcal{A}^-$, and the connected component of $(\mathcal{N}, \mathbf{A})$ containing $(h; j_1)$ contains just the two edges $(h; j_1), (h; j_2)$; delete both $(h; j_1)$ and $(j_1; j_2)$ from $\mathbf{A}$, rematch using $\mathcal{P}$ and then include $(j_1; j_2)$ as a new matching edge. Revise all the blossoms along $\mathcal{P}^1$ as discussed in Section 10.1.2. Chop down the tree rooted at r. If there are no exposed nodes left go to Step 13. Otherwise change the list into the set of all outer nodes and go back to Step 2.

Step 7 Matching and Covering Augmentation Procedure 3 We come to this step when scanning or inspection has revealed an outer labeled current node p and a type C or D pseudonode w such that $(p; w) \in \mathcal{A}^1_*(\pi, \mu)$. Let i_1 be the apex of w, and i_2, i_3 the pair of type 1 nodes inside w. So, $(i_1; i_2), (i_1; i_3) \in \mathbf{A}$ and all nodes inside w other than i_1, i_2, i_3 are matched nodes. Let s, j be original nodes inside p, w respectively such that $(s; j) \in \mathcal{A}_*(\pi, \mu)$.

By expanding the pseudonodes along the predecessor paths of p find the alternating path $\mathcal{P}_1$ in G between s and the apex node of the root associated with p, beginning with the matching edge incident at s, if s and the apex of the root node of p are different. By expanding the pseudonodes on the odd cycle in the simple blossom corresponding to w, find the alternating path $\mathcal{P}_2$ in G between i_2 and i_3 that is contained within w. Delete $(i_1; i_2)$ and $(i_1; i_3)$ from $\mathbf{A}$. When the two edges $(i_1; i_2), (i_1; i_3)$ are added to $\mathcal{P}_2$ it becomes a cycle, call it $\mathbb{C}_2$. First make all the edges in $\mathbb{C}_2$ into nonmatching edges. Then traverse $\mathbb{C}_2$ beginning from j, making the edges traveled alternately into nonmatching and matching edges beginning with the first edge incident at j kept as a nonmatching edge, and returning to j at the end.

Then rematch using $\mathcal{P}_1$, and finally make $(s; j)$ into a matching edge. Revise all the blossoms that have nonempty intersections with $\mathcal{P}_1, \mathbb{C}_2$ as in Section 10.1.2. Chop down the tree containing p. Now w becomes a type A pseudo- node with j as its apex. If there are no exposed nodes left go to Step 13. Otherwise change the list into the set of all outer labeled current nodes, and go back to Step 2, or the inspection in Step 12, depending on whether you arrived at this step from Step 3 or inspection.

Step 8 Matching and Covering Augmentation Procedure 4 We come to this step when scanning or inspection has identified an inner labeled current node j joined by a current matching edge to a pseudonode t that contains an original node i associated with node price $\pi_i = 0$, inside it. Let h be the apex of t. Let $\mathcal{P}^1$ be the path consisting of the matching edge $(t; j)$ and the predecessor path of j. By expanding the pseudonodes along $\mathcal{P}^1$ obtain the alternating path $\mathcal{P}$ from h to the apex of the root associated with j.

In this procedure we convert t into a type C or D pseudonode with i inside it (associated with $\pi_i = 0$) as its apex node. For this, obtain the alternating cycle $\mathbb{C}$ from h (the present apex of t) to h by expanding the pseudonodes on the odd cycle in the simple blossom corresponding to t. Let $(i; i_1), (i; i_2)$ be the edges on $\mathbb{C}$ incident at i. Let $\mathcal{P}_2$ be the path left between i_1 and i_2 when the edges $(i; i_1), (i; i_2)$ and node i are deleted from $\mathbb{C}$. First make all the edges on $\mathbb{C}$ into nonmatching edges. Then add $(i; i_1), (i; i_2)$ to the set $\mathbf{A}$. Now make the edges along $\mathcal{P}_2$ as you travel from i_1 to i_2, alternately nonmatching and matching edges beginning with the edge incident at i_1 as a nonmatching edge. Revise all the blossoms on the odd cycle in the simple blossom corresponding to t.

Rematch using $\mathcal{P}$. Revise all the blossoms along $\mathcal{P}^1$. If there are no exposed nodes left go to Step 13. Otherwise chop down the tree containing j, change the list into the set of all outer current nodes, and go back to inspection in Step 12, or Step 2 depending on whether you arrived at this step from inspection or Step 3.

Step 9 Matching and Covering Augmentation Procedure 5 We come to this step when inspection has identified an outer current node j which is a rooted pseudonode containing within it an original node i with $\pi_i = 0$. Convert j into a type C or D pseudonode with i as its apex node, as discussed in Step 8. If there are no exposed nodes left, go to Step 13. Otherwise chop down the tree containing j, change the list into the set of all outer current nodes and go back to inspection in Step 12.

Step 10 Blossom Shrinking We come to this step when scanning or inspection has revealed two outer current nodes p, j with the same root index, such that $(p; j) \in \mathcal{A}^1_*(\pi, \mu)$. Identify the simple blossom (it will either be rooted or type A) and shrink it into a pseudonode as in Step 6 of the algorithm of Section 10.1.1. Include the new pseudonode (now outer labeled) in the list, and go back to inspection in Step 12 or Step 2 depending on whether you arrived here from inspection or Step 3.

Step 11 Type A Pseudonode Unshrinking Identify all pseudonodes which are current nodes labeled as inner nodes (so they will be type A) associated with pseudonode price $\mu_\sigma = 0$. Unshrink each of them as discussed in Section 10.1.2. Repeat this process as often as possible. Go back to inspection in Step 12, or Step 2, depending on whether you came to this step from inspection or Step 3.

Step 12 Dual Solution Change We come to this step when the Hungarian forest conditions are satisfied. Let (π, μ) be the present dual feasible solution. The new dual feasible solution is $(\hat{\pi} = (\hat{\pi}_i), \hat{\mu} = (\hat{\mu}_\sigma))$ defined below.

$\mathbf{\Delta}_1 =$ $\{i : i \in \mathcal{N}$ is either an inner current node, or contained inside an outer labeled pseudonode$\}$

$\mathbf{\Delta}_2 =$ $\{i : i \in \mathcal{N}$ is inside an unlabeled current node$\}$

$\mathbf{\Delta}_3 =$ $\{i : i \in \mathcal{N}$ is inside an outer current node$\}$

$\mathbf{\Delta}_4 =$ $\{i : i \in \mathcal{N}$ is either an outer current node, or contained inside an inner labeled pseudonode$\}$

$\delta_1 =$ Min. $\{\pi_i : i \in \mathbf{\Delta}_1\}$

$\delta_2 =$ Min. $\{c_{ij} - d_{ij}(\pi, \mu) : i \in \mathbf{\Delta}_2, j \in \mathbf{\Delta}_3$ and $(i;j) \in \mathcal{A}\}$.

$\delta_3 =$ Min. $\{\frac{1}{2}(c_{ij} - d_{ij}(\pi, \mu)) : (i;j) \in \mathcal{A}$, and $i, j \in \mathbf{\Delta}_3$ but in different current nodes$\}$

$\delta_4 =$ Min. $\{\frac{1}{2}\mu_\sigma : \mathbf{Y}_\sigma$ is the set of original nodes inside an inner current node that is a pseudonode$\}$

$\delta =$ Min. $\{\delta_1, \delta_2, \delta_3, \delta_4\}$

$$\hat{\pi}_i = \begin{cases} \pi_i - \delta & \text{for all } i \in \mathbf{\Delta}_1 \\ \pi_i + \delta & \text{for all } i \in \mathbf{\Delta}_4 \\ \pi_i & \text{otherwise} \end{cases}$$

$$\hat{\mu}_\sigma = \begin{cases} \mu_\sigma + 2\delta & \text{if } \mathbf{Y}_\sigma \text{ is the set of original nodes in} \\ & \text{a current outer labeled pseudonode} \\ \mu_\sigma - 2\delta & \text{if } \mathbf{Y}_\sigma \text{ is the set of original nodes in} \\ & \text{a current inner labeled pseudonode} \\ \mu_\sigma & \text{otherwise} \end{cases}$$

Inspection

If $\delta = \delta_1$, look for an original node u which is an inner current node associated with $\hat{\pi}_u = 0$. If p is the predecessor of u, with p, u apply Step 5. Repeat as often as possible. Then look for an original node i with $\hat{\pi}_i = 0$ inside an outer current node j that is a pseudonode. If j is rooted (type A) apply Step 9 (Step 8) with it. Repeat as often as possible.

If $\delta = \delta_2$, look for $(i;j) \in \mathcal{A}_*(\hat{\pi}, \hat{\mu})$ with i inside an unlabeled node, and j inside an outer labeled node. Then include the current node containing j in the list.

If $\delta = \delta_3$, look for $(i;j) \in \mathcal{A}_*(\hat{\pi}, \hat{\mu})$ with i, j contained inside different outer nodes. If the root nodes of the current nodes containing i, j are different (the same) apply Step 4 (Step 10) with them. Repeat as often as possible.

If $\delta = \delta_4$ look for inner nodes that are pseudonodes associated with $\hat{\mu}_\sigma = 0$ and go to Step 11 if there are any.

Go to Step 2.

Step 13 Termination We reach this step when there are no exposed nodes left. Let $\overline{\mathbf{E}}, \overline{x}, (\overline{\pi}, \overline{\mu})$ be the present solution set of edges, its incidence vector, and the dual solution at this stage. $\overline{\mathbf{E}}$ is a minimum cost edge cover in G, $\overline{x}$ is the associated minimum cost edge covering vector, and $(\overline{\pi}, \overline{\mu})$ the optimum dual solution (optimum for (10.43)).

Validity of the Algorithm and Its Computational Complexity

1. We will now show that δ is finite and > 0 whenever Step 12 is carried out in the algorithm; and that the conditions :$\pi \geqq 0, \mu \geqq 0, d_{ij}(\pi, \mu) \leqq c_{ij}$ for all $(i; j) \in \mathcal{A} \backslash \mathcal{A}^-$, hold throughout. Clearly, these conditions hold initially, since $\pi = 0, \mu = 0$ at that stage. When the algorithm arrives at Step 12 at some stage suppose (π, μ) is the dual solution and that these conditions hold. Each of δ_1 to δ_4 is the minimum of a set of numbers, and every number in each of these sets is $\geqq 0$ because of these conditions. Also, if any number in any of these sets is 0, it can be verified that it provides an opportunity to carry out one of Steps 4, 5, 8, 9, 10, or 11, or to label a new unlabeled node, contradicting the Hungarian forest conditions which must hold when the algorithm arrives at Step 12. Hence each of δ_1 to δ_4 is either finite and > 0 or $+\infty$ (which happens when the set in which it is the minimum is $\emptyset$).

   So, δ is finite and > 0, or $+\infty$. Suppose $\delta = +\infty$. So, all δ_1 to δ_4 must be $+\infty$. δ_1, δ_4 are both $+\infty$ implies that there are no inner labeled nodes at this stage. Hence, the only labeled nodes must be the root nodes which are outer labeled. Now, δ_2, δ_3 are both $+\infty$ implies that there are no edges joining a pair of outer labeled nodes, and no edges joining an outer node to an unlabeled node. These facts imply that the root nodes are isolated nodes in G, contradicting the hypothesis that G has no isolated nodes. Hence δ cannot be $+\infty$. So, δ is finite and > 0.

Let $\hat{\pi}, \hat{\mu}$ be the new dual solution obtained in that Step 12. The definition of δ_1 and δ imply that $\hat{\pi} \geqq 0$. The definition of δ_4 and δ imply that $\hat{\mu} \geqq 0$. The definition of $\delta_2, \delta_3, \delta$ imply that $c_{ij} - d_{ij}(\hat{\pi}, \hat{\mu}) \geqq 0$ for all $(i; j) \in \mathcal{A} \backslash \mathcal{A}^-$. Hence if the conditions mentioned above hold at the beginning of a Step 12, they continue to hold after that Step 12 is executed. Repeating the same argument after each occurrence of Step 12, we conclude that these conditions hold always and that $\delta > 0$ and finite in every Step 12 carried out in the algorithm.

2. We will now show that $\mathbf{M} \cup (\mathbf{A} \backslash \mathcal{A}^-) \subset \mathcal{A}_*(\pi, \mu)$ throughout the algorithm, and that all in-tree edges are always in $\mathcal{A}^1_*(\pi, \mu)$ at that stage. Also, all edges in the odd cycles of the various simple blossoms corresponding to the various pseudonodes at any stage, are equality edges at that stage.

   Initially $\mathbf{M} = \emptyset$ and $\mathbf{A} = \mathcal{A}^-$, hence these statements hold then trivially. Any new edge added to any alternating tree is always from the $\mathcal{A}^1_*(\pi, \mu)$ at that stage. Also, from the dual solution change formulas it can be verified that all in-tree edges continue to remain in the new current equality subnetwork after a dual solution change step because each in-tree edge joins an outer to an inner node. A newly created matching edge is always from $\mathcal{A}_*(\pi, \mu)$ at that stage, by the manner in which matching and covering augmentations are carried out. Also, during a dual solution change step, a current matching edge is either in-tree, or out-of-tree with both nodes on it unlabeled. From the dual solution change formulas these facts imply that all matching edges continue to remain in the new $\mathcal{A}_*(\pi, \mu)$ after each dual solution change step. Similarly it can be verified that all edges in $\mathbf{A} \backslash \mathcal{A}^-$ are always equality edges.

   When a new simple blossom is discovered, it is clear that all the edges on its odd cycle are equality edges. From the manner in which the dual solution is changed, we can verify that these edges remain equality edges as long as this simple blossom remains shrunken as a pseudonode.

Hence all the statements made above hold throughout the algorithm.

3. From these facts we verify that all the properties mentioned before the statement of the algorithm are all maintained in it. Let $\overline{\mathbf{E}}, \overline{x}, (\overline{\pi}, \overline{\mu})$ be the solution set of edges, its incidence vector, and the dual solution at termination. Since all the nodes are covered at that stage $\overline{\mathbf{E}}$ is an edge cover in G. It can be verified that $\overline{x}, (\overline{\pi}, \overline{\mu})$ together satisfy all the conditions (10.45) to (10.50). Hence $\overline{x}$ is an optimum solution of (10.40). Since it is also a 0-1 vector, it is an optimum solution of (10.36), (10.37), and hence an optimum edge covering vector for G, and therefore $\overline{\mathbf{E}}$ is an optimum edge cover.

4. We will now analyze the worst case computational complexity of this algorithm. Each time one of Steps 4 to 9 occur, the number of exposed nodes decreases by 1 or 2. So, Steps 4 to 9 can occur at most n times. Define a **stage** in this algorithm to begin either with the initial Step 1 (the **initial stage**) or after one of Steps 4 to 9 has just been completed, and to end when one of Steps 4 to 9 is completed the next time. So, there are at most n stages in the algorithm, and in each stage, exactly one of Steps 4 to 9 occurs.

   A newly shrunken pseudonode always receives an outer label when it is formed, and it either remains outer labeled, or gets absorbed inside another pseudonode till the end of the stage in which it is formed. Thus by the results in Section 10.1, the total number of times that Step 10 (blossom shrinking step) can occur in a stage is at most $(n/2)$.

   Only inner labeled pseudonodes are unshrunk in the algorithm. So, any pseudonode which is unshrunk in a stage must have been formed in earlier stages. Hence, by the results in Section 10.1, Step 11 (pseudonode unshrinking step) can occur at most $(n/2)$ times in a stage.

Define $\mathbf{J} = \{i : i \in \mathcal{N}$, and i is inside an outer node$\}$, $\mathbf{I} = \{p : p$ is an inner labeled current node$\}$, $\theta = |\mathbf{J}| + |\mathbf{I}|$. Whenever a blossom shrinking step (Step 10) is carried out, it can be verified that $|\mathbf{J}|$ strictly increases, and that θ does not decrease. Whenever a pseudonode unshrinking step (Step 11) is carried out, it can be verified that θ strictly increases. Also, θ strictly increases whenever a tree growth step occurs. And $\theta \leqq n$.

Whenever Step 12 occurs with $\delta = \delta_1$, it leads to Step 5 and then the end of the stage. If Step 12 occurs with $\delta = \delta_2$ or δ_4, a tree growth step, or pseudonode unshrinking step occurs, and since θ increases by at least 1 in each of these steps, Step 12 with $\delta = \delta_2$ or δ_4 can occur at most n times in a stage. Step 12 with $\delta = \delta_3$ leads to Step 4 or Step 10, and hence this can occur at most $1+(n/2)$ times in a stage. So, in a stage Step 12 can occur at most $1 + n + (n/2)$ times. The computational effort in one of Steps 4 to 12 can be verified to be bounded above by $O(m)$. So, the total effort in a stage of this algorithm is bounded above by $O(nm)$. Since there are at most n stages, the overall computational effort in this algorithm is bounded above by $O(n^2m)$.

The Convex Hull of Edge Covering Vectors

Let $\mathbf{K}^{\geqq}$ denote the convex hull of edge covering vectors in G. If G has some isolated nodes, it has no edge cover, and hence $\mathbf{K}^{\geqq} = \emptyset$, and it can be verified that the set of feasible solutions of the system (10.40) is also empty in this case. Under the assumption that there are no isolated nodes in G, we will use the same fundamental proof technique as in Section 10.1.4 to show that $\mathbf{K}^{\geqq}$ is the set of feasible solutions of (10.40). We have shown that the LP (10.40) has an optimum solution which is an edge covering vector in G, and that the blossom algorithm discussed above will find it irrespective of what the edge cost vector c may be. This implies that all extreme points of the set of feasible solutions of (10.40) are edge covering vectors in G. Also, every edge covering vector in G is feasible to (10.40), and it is an extreme point of the set of feasible solutions of this system since it is a 0-1 vector, and all the variables in this system are bounded by 0 and 1. Thus the system of constraints in (10.40) provides a linear constraint representation for $\mathbf{K}^{\geqq}$, in all cases.

Comment 10.3 The blossom algorithm for the minimum cost edge covering problem, and the results in this subsection are taken from Murty and Perin [1982].

10.3 Minimum Cost 1-Matching/Edge Coverings

Let G $= (\mathcal{N}, \mathcal{A}, c = (c_{ij}))$ be the original undirected network with c as the edge cost vector and $(\mathcal{N}^{\leqq}, \mathcal{N}^{=}, \mathcal{N}^{\geqq}, \mathcal{N}^{0})$ as the partition of $\mathcal{N}$. Here we discuss a primal-dual blossom algorithm for the minimum cost 1-M/EC problem (10.6), (10.7). We will replace the integrality restrictions (10.7) by blossom constraints, thereby transforming the integer program (10.6), (10.7) into an LP for which we develop a primal-dual blossom algorithm. Let $\mathbf{Y} \subset \mathcal{N} \backslash \mathcal{N}^{0}$ with $|\mathbf{Y}|$ odd and $\geqq 3$. There are 2 different blossom constraints associated with $\mathbf{Y}$; the *matching blossom constraint* $\mathbf{Y}^{-}(x) \leqq (|\mathbf{Y}| - 1)/2$, and the *covering blossom constraint* $\mathbf{Y}^{+}(x) \geqq (|\mathbf{Y}| + 1)/2$; where $\mathbf{Y}^{-}(x)$ and $\mathbf{Y}^{+}(x)$ are defined as in (10.19), (10.38) respectively. From the definitions it is clear that $\mathbf{Y}^{-}(x) \geqq \mathbf{Y}^{+}(x)$ for every edge vector x in G. Every 1-M/EC vector x satisfies the matching blossom constraint (edge covering blossom constraint) corresponding to $\mathbf{Y}$ if $\mathbf{Y} \subset \mathcal{N}^{\leqq}$ ($\mathbf{Y} \subset \mathcal{N}^{\geqq}$) but may or may not satisfy the covering blossom constraint (matching blossom constraint) corresponding to such a $\mathbf{Y}$. If $\mathbf{Y} \subset \mathcal{N}^{=}$, every 1-M/EC vector x satisfies both the matching and edge covering blossom constraints corresponding to $\mathbf{Y}$. Whenever $\mathbf{Y}$ is any subset of $\mathcal{N} \backslash \mathcal{N}^{0}$, every 1-M/EC vector satisfies at least one of the matching or covering blossom constraints corresponding to $\mathbf{Y}$. We will introduce either the matching blossom constraint or the edge covering blossom constraint corresponding to $\mathbf{Y}$, to replace the integer requirements (10.7). We use the symbols

MB, CB to denote the class of all these **Y** for which the matching blossom constraint, edge covering blossom constraint are introduced respectively.

We will include **Y** in **MB** if $\mathbf{Y} \subset \mathcal{N}^{\leqq} \cup \mathcal{N}^{=}$, and in **CB** if $\mathbf{Y} \subset \mathcal{N}^{\geqq} \cup \mathcal{N}^{=}$ and contains at least one node from $\mathcal{N}^{\geqq}$. If $\mathbf{Y} \subset \mathcal{N}^{\leqq} \cup \mathcal{N}^{=} \cup \mathcal{N}^{\geqq}$ and contains at least one node from $\mathcal{N}^{\leqq}$ and from $\mathcal{N}^{\geqq}$, it will be included either in **MB** or in **CB** by a rule specified in the algorithm. This rule is based on identifying a specific node in **Y** called the BCI (*blossom constraint identifier*) node, using the dual solution at that stage. If the BCI node is from $\mathcal{N}^{\leqq}$ ($\mathcal{N}^{\geqq}$) **Y** will be included in **MB** (**CB**). The classes **MB, CB** will be fully known only when the algorithm has terminated. The modified LP corresponding to the blossom constraint specification dictated by the choice of the classes **MB, CB** is : find $x = (x_{ij})$ to

$$
\begin{array}{rcl}
\text{Minimize} \quad \sum(c_{ij}x_{ij} & : & \text{over } (i;j) \in \mathcal{A}) \\
\text{Subject to} \quad x(i) & & \left\{ \begin{array}{l} \leqq 1, \text{ for } i \in \mathcal{N}^{\leqq} \\ = 1, \text{ for } i \in \mathcal{N}^{=} \\ \geqq 1, \text{ for } i \in \mathcal{N}^{\geqq} \end{array} \right. \\
\mathbf{Y}^{-}(x) & \leqq & (|\mathbf{Y}| - 1)/2 \quad \text{for each } \mathbf{Y} \in \mathbf{MB} \\
\mathbf{Y}^{+}(x) & \geqq & (|\mathbf{Y}| + 1)/2 \quad \text{for each } \mathbf{Y} \in \mathbf{CB} \\
 & & \\
0 \leqq x_{ij} & \leqq & 1 \quad \text{for all } (i;j) \in \mathcal{A}
\end{array}
\tag{10.55}
$$

It should be understood that (10.55) is not necessarily equivalent to (10.6), (10.7). However, we develop a blossom algorithm that obtains an optimum solution of (10.6), (10.7) provided it has a feasible solution, using a modified LP of the form (10.55) with the blossom constraint specification classes **MB, CB** which are themselves obtained during the algorithm. Associate the original node price π_i for $i \in \mathcal{N}$ to the first set of constraints (node degree constraints) in (10.55); and the pseudonode price μ_σ to the blossom constraint corresponding to a subset of nodes $\mathbf{Y}_\sigma \in \mathbf{MB} \cup \mathbf{CB}$. Given the classes **MB, CB**, the dual feasibility conditions on $\pi = (\pi_i), \mu = (\mu_\sigma)$ reflecting the structure of the primal problem (10.55), are

$$\begin{array}{ll} \pi_i & \left\{ \begin{array}{l} = 0, \text{ for all } i \in \mathcal{N}^0 \\ \leqq 0, \text{ for all } i \in \mathcal{N}^{\leqq} \\ \geqq 0, \text{ for all } i \in \mathcal{N}^{\geqq} \end{array} \right. \\ \mu_\sigma & \left\{ \begin{array}{l} \leqq 0, \text{ for } \sigma \text{ such that } \mathbf{Y}_\sigma \in \mathbf{MB} \\ \geqq 0, \text{ for } \sigma \text{ such that } \mathbf{Y}_\sigma \in \mathbf{CB} \end{array} \right. \end{array} \tag{10.56}$$

The dual of (10.55) is

$$\begin{array}{rcl} \text{Maximize } \sum(\pi_i & : & \text{over } i \in \mathcal{N}) \\ & + & \sum(\mu_\sigma(|\mathbf{Y}_\sigma| - 1)/2 : \text{ over } \sigma \text{ s. t. } \mathbf{Y}_\sigma \in \mathbf{MB}) \\ & + & \sum(\mu_\sigma(|\mathbf{Y}_\sigma| + 1)/2 : \text{ over } \sigma \text{ s. t. } \mathbf{Y}_\sigma \in \mathbf{CB}) \\ & + & \sum(v_{ij} : \text{ over } (i;j) \in \mathcal{A}) \\ \text{subject to} & & (10.56) \text{ and} \\ d_{ij}(\pi, \mu) + v_{ij} & \leqq & c_{ij}, \text{ for } (i;j) \in \mathcal{A} \\ v_{ij} & \leqq & 0, \text{ for } (i;j) \in \mathcal{A} \end{array} \tag{10.57}$$

where for $(i;j) \in \mathcal{A}$

$$\begin{array}{rcl} \mu(i;j) & = & \sum(\mu_\sigma : \text{ over } \sigma \text{ s. t. } \mathbf{Y}_\sigma \in \mathbf{MB} \text{ contains both } i \text{ and } j\ ) \\ & & + \sum(\mu_\sigma : \text{ over } \sigma \text{ s. t. } \mathbf{Y}_\sigma \in \mathbf{CB} \text{ contains either} \\ & & \qquad i \text{ or } j \text{ or both}) \\ d_{ij}(\pi, \mu) & = & \pi_i + \pi_j + \mu(i;j) \end{array} \tag{10.58}$$

Notice that the formula for $d_{ij}(\pi, \mu)$ in (10.58) is different from those in earlier sections, since the structure of the LP (10.55) here is different from those in earlier sections. From the structure of (10.57), it is clear that in an optimum solution $(\pi, \mu), v$ for it, we will have for each $(i; j) \in \mathcal{A}$, $v_{ij} = \text{min. } \{0, c_{ij} - d_{ij}(\pi, \mu)\}$. Thus without any loss of generality, we can eliminate the variables v_{ij} from the dual objective function in (10.57) and express it purely in terms of (π, μ), leading to $W(\pi, \mu)$ defined below in (10.59).

$$\begin{aligned} W(\pi, \mu) &= \sum(\pi_i : \text{ over } i \in \mathcal{N}) \\ &+ \sum(\mu_\sigma(|\mathbf{Y}_\sigma| - 1)/2 : \text{ over } \sigma \text{ s. t. } \mathbf{Y}_\sigma \in \mathbf{MB}) \\ &+ \sum(\mu_\sigma(|\mathbf{Y}_\sigma| + 1)/2 : \text{ over } \sigma \text{ s. t. } \mathbf{Y}_\sigma \in \mathbf{CB}) \quad (10.59) \\ &+ \sum(\text{min. } \{0, c_{ij} - d_{ij}(\pi, \mu)\}\text{: over } (i; j) \in \mathcal{A}) \end{aligned}$$

The complementary slackness conditions for optimality in the primal, dual pair (10.55). (10.57) are: for all $i \in \mathcal{N}$ and $(i, j) \in \mathcal{A}$

$$\pi_i(x(i) - 1) = 0 \quad (10.60)$$

$$d_{ij}(\pi, \mu) \begin{cases} < c_{ij} \text{ implies } x_{ij} = 0 \\ > c_{ij} \text{ implies } x_{ij} = 1 \end{cases} \quad (10.61)$$

$$\left.\begin{aligned} \mu_\sigma((|\mathbf{Y}| - 1)/2 - \mathbf{Y}^-(x)) = 0, \text{ for } \sigma \text{ s. t. } \mathbf{Y}_\sigma \in \mathbf{MB} \\ \mu_\sigma((|\mathbf{Y}| + 1)/2 - \mathbf{Y}^+(x)) = 0, \text{ for } \sigma \text{ s. t. } \mathbf{Y}_\sigma \in \mathbf{CB} \end{aligned}\right\} \quad (10.62)$$

The algorithm maintains a subset of edges $\mathbf{E} \subset \mathcal{A}$ called the **solution set of edges**, and the corresponding edge vector $x = x^{\mathbf{E}}$. Edges not in $\mathbf{E}$ (i.e., those $(i; j)$ with $x_{ij} = 0$) are called **nonsolution edges**. Let

$$\mathcal{A}^- = \{(i; j) : (i; j) \in \mathcal{A}, i \text{ and } j \text{ both in } \mathcal{N}^{\geqq} \cup \mathcal{N}^0, \text{ and } c_{ij} \leqq 0\} \quad (10.63)$$

It is clear that if (10.6), (10.7) is feasible, then there exists a minimum cost 1-M/EC which contains $\mathcal{A}^-$ as a subset. The algorithm maintains $\mathbf{E} \supset \mathcal{A}^-$ always. Actually the initial solution set of edges $\mathbf{E}$ will be $\mathcal{A}^-$. The algorithm maintains a dual feasible solution (π, μ) such that $x, (\pi, \mu)$ together satisfy the complementary slackness conditions (10.60). (10.61), (10.62) always. And if $\mu_\sigma \neq 0$, $\mathbf{Y}_\sigma$ will be the set of original nodes inside an existing pseudonode. **E** is partitioned into **M, A** where **M** is a matching ; and **A** is called the set of **covering edges** and it always includes $\mathcal{A}^-$, and every solution edge which is adjacent to another solution edge. Every connected component of $(\mathcal{N}, \mathbf{E})$ that consists of a single edge $(i;j)$ satisfying $d_{ij}(\pi, \mu) = c_{ij}$ will be in **M**. Also, if there is a connected component of $(\mathcal{N}, \mathbf{A})$ that consists of a single edge $(i;j)$, it will satisfy $d_{ij}(\pi, \mu) > c_{ij}$. During the algorithm an original node i is said to be a

Matched node	if it is incident with a matching edge
Type 1 node	if it is incident with exactly one edge from **A**
Type 2 node	if $i \in \mathcal{N}^{\geqq} \cup \mathcal{N}^0$, and it is incident with $\geqq 2$ edges from **A**
Type 0 node	if there is no solution edge incident at it; and either $i \in \mathcal{N}^0$, or $i \in \mathcal{N}^{\leqq}$ and $\pi_i = 0$
Exposed node	if there is no solution edge incident at it; and either $i \in \mathcal{N}^{\geqq} \cup \mathcal{N}^=$, or $i \in \mathcal{N}^{\leqq}$ and $\pi_i < 0$

Original nodes of types 1, 2, 0, or matched nodes are called **nonexposed nodes**. Once a node becomes nonexposed, it remains nonexposed in the sequel, but its classification among matched, types 1, 2, 0 might change from step to step. The algorithm terminates either when there are no more exposed nodes in G, or when it becomes clear that it is impossible to convert any more exposed nodes into nonexposed nodes.

If (π, μ) is the present dual feasible solution, define $\mathcal{A}_*(\pi, \mu) = \{(i;j) : (i;j) \in \mathcal{A}$ and $d_{ij}(\pi, \mu) = c_{ij}\ \}$, where $d_{ij}(\pi, \mu)$ is defined as in (10.58). Edges in $\mathcal{A}_*(\pi, \mu)$ are the **original equality edges**, and $\mathrm{G}_*(\pi, \mu) = (\mathcal{N}, \mathcal{A}_*(\pi, \mu))$ is the **equality subnetwork** of G wrt (π, μ). The algorithm plants an alternating tree at each exposed node and grows these trees wrt **M** in $\mathrm{G}_*(\pi, \mu)$. During this process, simple blossoms may be discovered and shrunk into pseudonodes. The nodes in $\mathcal{N}^0$ or any node on an edge in $\mathcal{A}^-$ will never be contained on any simple blossom, and hence will never be inside any pseudonode.

Blossoms and pseudonodes into which they are shrunk are classified by the configuration of the simple blossom corresponding to them. The various types of simple blossoms, blossoms, and pseudonodes are

Types, rooted, A,B,C,D	defined exactly as in Section 10.2.2
Type E	if the base node has no solution edge incident at it, but is either a type 0 node in $\mathcal{N}^{\leqq}$, or another lower level type E pseudonode

Thus the apex node of a type E pseudonode is always the unique type 0 original node in $\mathcal{N}^{\leqq}$ contained inside it, with no solution edge incident at it. If $\mathcal{N}_B$ is the set of original nodes inside a type E pseudonode, there exists no solution edges joining a node in $\mathcal{N}_B$ to one outside it.

A newly formed pseudonode will always be a type A or a rooted pseudonode when it is formed. Types B,C,D,E pseudonodes are obtained through transformation of a type A or rooted pseudonode during an augmentation step. $\mathrm{G}^1 = (\mathcal{N}^1, \mathcal{A}^1)$ denotes the current network. $\mathbf{M}^1, \mathbf{A}^1, \mathbf{E}^1 = \mathbf{M}^1 \cup \mathbf{A}^1$ denote the set of current matching, covering, and solution edges respectively. A type A (B) current pseudo- node in G^1 is incident with a current matching edge (exactly one current covering edge). Type C,D,E and rooted current pseudonodes are incident with no current solution edges. Note that there is no pseudonode incident with 2 or more current solution edges.

The BCI Node of a Blossom

BCI (**blossom constraint identifier**) nodes are only defined for blossoms and pseudonodes containing nodes from both $\mathcal{N}^{\leqq}$ and $\mathcal{N}^{\geqq}$ inside them. Let $\mathrm{G}_B = (\mathcal{N}_B, \mathcal{A}_B)$ be a blossom identified during the algorithm which has been shrunk into the pseudonode B. If $\mathcal{N}_B \cap \mathcal{N}^{\leqq}$ and $\mathcal{N}_B \cap \mathcal{N}^{\geqq}$ are both nonempty we identify an original node j tying for the minimum in (break ties arbitrarily)

$$\text{Min. } \{|\pi_i| : i \in \mathcal{N}_B \cap (\mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq})\} \qquad (10.64)$$

A node j selected among those attaining the minimum in (10.64) is known as the BCI node for the blossom G_B or the pseudonode B. If the BCI node $j \in \mathcal{N}^{\leqq}$ $(j \in \mathcal{N}^{\geqq})$ $\mathcal{N}_B$ is included in the class **MB** (**CB**) and the matching (covering) blossom constraint corresponding to G_B is introduced. Once a BCI node for a pseudonode B is selected, it never changes in the algorithm as long as B remains, also its BCI node continues to tie for the minimum in (10.64) as long as B remains a current node. If the pseudonode B gets absorbed into another pseudonode, its BCI node may not tie for the minimum in (10.64) during the dormant period for B, but it will start being satisfied again as soon as the dormant period ends and B becomes current again.

If p is a pseudonode containing other pseudonodes inside it, it is possible that p has a BCI node different from the BCI nodes of pseudonodes inside it. Also, p may belong in the class **MB** or **CB**, and contain inside it other pseudonodes which belong in either class or both.

Properties Maintained By the Algorithm

Here we summarize the properties maintained by the algorithm. Labeled nodes are always either exposed nodes or matched nodes. Exposed nodes are the roots of alternating trees and labeled as outer nodes, so all inner nodes will always be matched nodes. All in-tree edges will always be current equality edges from the set $\mathcal{A}^1_*(\pi, \mu)\backslash\mathbf{A}^1$ at that stage. The following properties always hold.

(π, μ) is always dual feasible.
$\mathbf{M} \subset \mathcal{A}_*(\pi, \mu), \mathbf{A} \supset \mathcal{A}^-, \mathbf{E} \subset \mathcal{A}^- \cup \mathcal{A}_*(\pi, \mu)$.
If $(i; j) \not\in \mathbf{E}$ then $d_{ij}(\pi, \mu) \leqq c_{ij}$.
$x(i) > 1$ implies $i \in \mathcal{N}^0 \cup \mathcal{N}^{\geqq}$ and $\pi_i = 0$.
$x(i) < 1$ implies that either $i \in \mathcal{N}^0 \cup \mathcal{N}^{\leqq}$ with $\pi_i = 0$, or $i \in \mathcal{N}\backslash\mathcal{N}^0$ and is an exposed node.
$\mu_\sigma < 0$ implies that $\mathbf{Y}^-_\sigma(x) = (|\mathbf{Y}_\sigma| - 1)/2$; $\mu_\sigma > 0$ implies that either $\mathbf{Y}^+_\sigma(x) = (|\mathbf{Y}_\sigma| + 1)/2$, or $\mathbf{Y}^+_\sigma(x) = (|\mathbf{Y}_\sigma| - 1)/2$ with $\mathbf{Y}_\sigma$ being the set of original nodes inside a rooted pseudonode.

Nodes in $\mathcal{N}^0$ and those on edges in $\mathcal{A}^-$ are never contained inside any pseudonode, and $\pi_i = 0$ always for these nodes i.
No type 1,2, or 0 nodes, or type B,C,D, or E pseudonodes will ever be in-tree nodes.

At some stage of the algorithm, if $\mathbf{E}$ is not yet a 1-M/EC; and no augmentation, blossom shrinking, pseudonode unshrinking steps are possible; and the list of labeled and unscanned nodes is empty (i.e., no tree growth is possible), we have a Hungarian forest. The following properties are satisfied at that time, and the algorithm moves to a dual solution change step.

Hungarian Forest Conditions

Each inner node is either matched or type A, incident with a matching edge joining it to an outer node.

There exists no node $i \in \mathcal{N}^{\overset{\leq}{=}}$ which is an outer current node with $\pi_i = 0$.

There exists no inner node $i \in \mathcal{N}^{\overset{\geq}{=}} \cup \mathcal{N}^0$ with $\pi_i = 0$.
If $\mathbf{Y}_\sigma$ is the set of original nodes inside a current inner pseudonode, then $\mu_\sigma \neq 0$.
There exists no current equality edge joining an outer to a non-inner (outer or unlabeled) node.
There exists no outer current pseudonode containing inside it an original node $i \in \mathcal{N}^{\overset{\leq}{=}} \cup \mathcal{N}^{\overset{\geq}{=}}$ with $\pi_i = 0$.

In the blossom algorithm described below, list always refers to the set of labeled and unscanned outer current nodes. The tree growth step in the algorithm is constructed in such a way that only outer nodes are scanned.

BLOSSOM ALGORITHM FOR THE MINIMUM COST 1-M/EC PROBLEM

Step 1 Initialization Initially let $\mathbf{M} = \emptyset, \mathbf{A} = \mathcal{A}^-, \mu = (\mu_\sigma) = 0$; and $\pi = (\pi_i)$ where $\pi_i = 0$ for all $i \in \mathcal{N}^{\overset{\geq}{=}} \cup \mathcal{N}^0$, and = min. $\{0, c_{jg} : (j; g) \in \mathcal{A}\}$ for all $i \in \mathcal{N}^{\overset{\leq}{=}} \cup \mathcal{N}^=$. If there are no exposed nodes, go to Step 13. Otherwise root an alternating tree at each exposed node i by labeling it with $(\emptyset, +, i)$. List = set of all these root nodes now.

Step 2 Select a Node to Scan If list = $\emptyset$ go to Step 12. Otherwise, select a node from it to scan, and delete it from the list.

Step 3 Scanning Let the node to be scanned be p with label (P(p), $+$, r). Do the following in the order given.

If there is an already labeled outer node j associated with a root node $s \neq r$ such that $(p; j) \in \mathcal{A}^1_*(\pi, \mu)$, an augmenting path has been found, go to Step 4.

If there is an already labeled outer node j associated with the same root node r such that $(p; j) \in \mathcal{A}^1_*(\pi, \mu)$, the alternating tree containing p, j has blossomed, go to Step 10.

If there is an original current node $u \in \mathcal{N}^0 \cup \mathcal{N}^{\geqq}$ such that $(p; u) \in \mathcal{A}^1_*(\pi, \mu)$ and $\pi_u = 0$, go to Step 5.

If there is an original current type 1 node or a type B pseudonode v such that $(p; v) \in \mathcal{A}^1_*(\pi, \mu)$, go to Step 6.

If there is an unlabeled current node w that is a type C or D or E pseudonode, or a type 0 original node, such that $(p; w) \in \mathcal{A}^1_*(\pi, \mu)$, go to Step 7.

Identify all $j \neq \mathrm{P}(p)$ such that $(p; j) \in \mathcal{A}^1_*(\pi, \mu)$. Since we came to this stage, j cannot be type 0, 1, or 2, or B, C, D, or E, or exposed, or a rooted pseudonode (in that case we would have already gone to some other step). For each such j do the following. If j is unlabeled, let $(j; t)$ be the current matching edge incident at j, label j with $(p, -, r)$. If t is unlabeled and either $t \in \mathcal{N}^0 \cup \mathcal{N}^{\leqq}$ with $\pi_t = 0$, or t is a pseudonode inside which there is an original node $i \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq}$ associated with node price $\pi_i = 0$ go to Step 8. Otherwise label t with $(j, +, r)$ and include it in the list.

Now p is labeled and scanned. Go to Step 11.

Step 4 Matching Augmentation Same as Step 4 in the algorithm of Section 10.2.1.

Step 5 Matching & Covering Augmentation Procedure 1 Same as Step 5 in the algorithm of Section 10.2.1.

Step 6 Matching & Covering Augmentation Procedure 2 Same as Step 6 in the algorithm of Section 10.2.1.

Step 7 Matching & Covering Augmentation Procedure 3 We

come to this step when scanning or inspection has revealed an outer current node p and a current node w which is either type C,D,E, or 0, and $(p; w) \in \mathcal{A}^1_*(\pi, \mu)$. If w is type C or D, this step is carried out exactly as Step 7 in the algorithm of Section 10.2.1.

Suppose w is either type 0 or E. Let i_1 be the apex node of w. Let $\mathcal{P}^1$ be the path obtained by adding $(p; w)$ to the predecessor path of p. Obtain the augmenting path $\mathcal{P}$ in G connecting i_1 with the apex of the root node associated with p, by expanding the pseudonodes along $\mathcal{P}^1$. Rematch using $\mathcal{P}$. Revise all the blossoms along $\mathcal{P}^1$ as in Section 10.1.2. Chop down the tree containing p. If there are no exposed nodes left, go to Step 13. Otherwise, change the list into the set of all outer labeled current nodes and go back to Step 2 or inspection in Step 12 depending on whether you arrived here from Step 3 or inspection.

Step 8 Matching and Covering Augmentation Procedure 4 We come to this step when scanning or inspection has identified an inner labeled current node j joined by a current matching edge to a current node t, where either $t \in \mathcal{N}^0 \cup \mathcal{N}^{\leqq}$ with $\pi_t = 0$, or t is a pseudonode containing inside it an original node $i \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq}$ with $\pi_i = 0$.

Let $\mathcal{P}^1$ be the predecessor path of j. Remove the matching edge joining the apex nodes of j and t from $\mathbf{M}$ and remove $(j; t)$ from $\mathbf{M}^1$. Rematch using $\mathcal{P}^1$ and revise all the blossoms along it as in Section 10.1.2. If t is a pseudonode and $i \in \mathcal{N}^{\geqq}$ convert t into a type C or D pseudonode with i as its apex, as in Step 8 in the algorithm of Section 10.2.1. If t is a pseudonode and $i \in \mathcal{N}^{\leqq}$, rematch within t to convert it into a type E pseudonode with i as its apex.

t gets converted into type 0, C, D, or E. If there are no exposed nodes left, go to Step 13. Otherwise, chop down the tree containing j, change the list into the set of all outer labeled current nodes, and go back to Step 2 or inspection in Step 12 depending on whether you arrived here from Step 3 or inspection.

Step 9 Matching and Covering Augmentation Procedure 5 We come to this step when inspection or blossom shrinking has identified an outer labeled current node t such that either $t \in \mathcal{N}^0 \cup \mathcal{N}^{\leqq}$ with $\pi_t = 0$, or t is a pseudonode containing inside it an original node $i \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq}$ with $\pi_i = 0$. Rematch the predecessor path of t and revise all the blossoms along it as in Section 10.1.2. If t is an original node, this process converts it into type 0. If t is a pseudonode, either convert it into type C or D with i as its apex if $i \in \mathcal{N}^{\geqq}$, or into type E with i as its apex if $i \in \mathcal{N}^{\leqq}$. If there are no exposed nodes left, go to Step 13. Otherwise chop down the tree containing t, change the list into the set of all outer labeled current nodes and go back to inspection in Step 12, or Step 10, or Step 2.

Step 10 Blossom Shrinking We come to this step when scanning or inspection has revealed two outer labeled current nodes p, j with the same root index, such that $(p; j) \in \mathcal{A}^1_*(\pi, \mu)$. This is an indication that the alternating tree containing p, j has blossomed. The simple blossom will either be rooted or type A. Identify the simple blossom and shrink it into a pseudonode as in Step 6 of the algorithm of Section 10.1.1. Include the new pseudonode, say B, which will now be an outer node, in the list. Let $\mathcal{N}_B$ be the set of original nodes inside this pseudonode. If $\mathcal{N}_B \subset \mathcal{N}^{\leqq} \cup \mathcal{N}^{=}$ ($\mathcal{N}_B \subset \mathcal{N}^{\geqq} \cup \mathcal{N}^{=}$ and $\mathcal{N}_B \cap \mathcal{N}^{\geqq} \neq \emptyset$) include $\mathcal{N}_B$ in the class **MB** (**CB**). In these two cases no BCI node is defined for this pseudonode. If $\mathcal{N}_B \cap \mathcal{N}^{\leqq} \neq \emptyset$ and $\mathcal{N}_B \cap \mathcal{N}^{\geqq} \neq \emptyset$, then define the BCI node of the pseudonode B to be a node j attaining the minimum in (10.64), breaking ties arbitrarily. If $j \in \mathcal{N}^{\geqq}$ ($j \in \mathcal{N}^{\leqq}$) include $\mathcal{N}_B$ in the class **CB** (**MB**). If there is an

$i \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq}$ inside B with $\pi_i = 0$ carry out Step 9. Return to Step 2 or inspection in Step 12, depending on whether you came here from Step 3 or inspection.

Step 11 Type A Pseudonode Unshrinking Same as Step 11 in the algorithm of Section 10.2.1.

Step 12 Dual Solution Change We come to this step when the Hungarian forest conditions are satisfied. Let (π, μ) be the present dual feasible solution. Define (adopt the convention that the minimum in the empty set is $+\infty$)

$\delta_1 =$ Min. $\{-\pi_i : i \in \mathcal{N}^{\leqq}$ is a current outer node$\}$

$\delta_2 =$ Min. $\{\pi_i : i \in \mathcal{N}^{\geqq}$ is a current inner node$\}$

$\delta_3 =$ Min. $\{-\pi_i : i \in \mathcal{N}^{\leqq}$ is inside a current outer labeled pseudonode in the class $\mathbf{MB}\}$

$\delta_4 =$ Min. $\{\pi_i : i \in \mathcal{N}^{\geqq}$ is inside a current outer labeled pseudonode in the class $\mathbf{CB}\}$

$\delta_5 =$ Min. $\{-\frac{1}{2}\mu_\sigma : \mathbf{Y}_\sigma \in \mathbf{MB}$ is the set of original nodes in a current inner labeled pseudonode$\}$

$\delta_6 =$ Min. $\{\frac{1}{2}\mu_\sigma : \mathbf{Y}_\sigma \in \mathbf{CB}$ is the set of original nodes in a current inner labeled pseudonode$\}$

$\delta_7 =$ Min. $\{\frac{1}{2}(c_{ij} - d_{ij}(\pi, \mu)) : (i;j) \in \mathcal{A}, i$ and j are inside distinct outer current nodes$\}$

$\delta_8 =$ Min. $\{(c_{ij} - d_{ij}(\pi, \mu)) : (i;j) \in \mathcal{A}$, one of i, j is inside an outer current node, and the other is inside an unlabeled node$\}$

$\delta =$ Min. $\{\delta_1, \dots, \delta_8\}$

If $\delta = +\infty$, go to Step 14. If δ is finite, it will be > 0 (proved below), define the new dual solution $\hat{\pi} = (\hat{\pi}_i), \hat{\mu} = (\hat{\mu}_\sigma)$ where

$$\hat{\pi}_i = \begin{cases} \pi_i + \delta & \text{if } i \text{ is an outer current node, or is inside} \\ & \text{an outer current pseudonode in } \mathbf{MB}\text{, or is} \\ & \text{inside an inner current pseudonode in } \mathbf{CB} \\ \pi_i - \delta & \text{if } i \text{ is an inner current node, or is inside} \\ & \text{an inner current pseudonode in } \mathbf{MB}\text{, or is} \\ & \text{inside an outer current pseudonode in } \mathbf{CB} \\ \pi_i & \text{if } i \text{ is inside an unlabeled node} \end{cases}$$

$$\hat{\mu}_\sigma = \begin{cases} \mu_\sigma + 2\delta & \text{if } \mathbf{Y}_\sigma \in \mathbf{MB} \text{ forms a current inner labeled} \\ & \text{pseudonode, or if } \mathbf{Y}_\sigma \in \mathbf{CB} \text{ forms a cur-} \\ & \text{rent outer labeled pseudonode} \\ \mu_\sigma - 2\delta & \text{if } \mathbf{Y}_\sigma \in \mathbf{MB} \text{ forms a current outer labeled} \\ & \text{pseudonode, or if } \mathbf{Y}_\sigma \in \mathbf{CB} \text{ forms a cur-} \\ & \text{rent inner labeled pseudonode} \\ \mu_\sigma & \text{otherwise} \end{cases}$$

Inspection

If $\delta = \delta_1$, look for a $t \in \mathcal{N}^{\leq}_{=}$ which is an outer current node associated with $\hat{\pi}_t = 0$, and then apply Step 8.

If $\delta = \delta_2$, look for a $u \in \mathcal{N}^{\geq}_{=}$ which is an inner current node associated with $\hat{\pi}_u = 0$, and then apply Step 5.

If $\delta = \delta_3$, look for an $i \in \mathcal{N}^{\leq}_{=}$ associated with $\hat{\pi}_i = 0$, that is inside an outer current pseudonode t with its predecessor j, and then apply Step 8.

If $\delta = \delta_4$, look for an $i \in \mathcal{N}^{\geq}_{=}$ associated with $\hat{\pi}_i = 0$, that is inside an outer current pseudonode t with its predecessor j, and then apply Step 8.

If $\delta = \delta_5$ or δ_6, look for an inner current pseudonode associated with the set of original nodes $\mathbf{Y}_\sigma$ for which $\hat{\mu}_\sigma = 0$, and unshrink it as in Step 11.

If $\delta = \delta_7$, look for current equality edges $(p; j)$ joining two outer current nodes, and perform Step 4 if p, j have different root indices, or Step 10 if they have the same root index.

If $\delta = \delta_8$, look for outer nodes p and unlabeled nodes j such that $(p; j) \in \mathcal{A}^1_*(\hat{\pi}, \hat{\mu})$, include such nodes p in the list.

Repeat the above checks as many times as possible. Every node or edge whose price or weight led to the value of δ must be checked. Then go to Step 2.

Step 13 Optimality We reach this step when there are no exposed nodes left. Let $\overline{\mathbf{E}}, \overline{x}, (\overline{\pi}, \overline{\mu})$ be the present solution set of edges, its incidence vector, and the dual solution at this stage. $\overline{\mathbf{E}}$ is a minimum cost 1-M/EC in G, $\overline{x}$ is the associated minimum cost 1-M/EC vector, and $(\overline{\pi}, \overline{\mu})$ the optimum dual solution (optimum for (10.57)). Terminate.

Step 14 Infeasibility We come to this step if $\delta = +\infty$ in a dual solution change step. In this case there exists no 1-M/EC in G wrt the given partition of the nodes, i.e., (10.6), (10.7) is infeasible. Terminate.

Validity of the Algorithm and Its Computational Complexity

By verifying for each step, we can check that the algorithm maintains all the properties claimed for it, and that the Hungarian forest conditions hold whenever the algorithm arrives at Step 12. These properties also guarantee that $\delta \geqq 0$ in Step 12. For δ to be 0, at least one among δ_1 to δ_8 must be 0, and if this happens the algorithm would have gone to the appropriate Steps 4 to 11 during scanning and augmented the matching and covering, or shrunk a simple blossom, or unshrunk a type A pseudonode, or labeled an unlabeled node, violating the Hungarian forest conditions at the time of arrival at Step 12. Hence $\delta > 0$ in Step 12 always.

Using exactly the same arguments as in Section 10.2.1, it can be shown that the computational effort required by this algorithm is bounded above by $O(n^{\mathbf{2}}m)$, which can be reduced to $O(n^{\mathbf{3}})$ by implementing it with the approach discussed in Section 10.1.2.

If $\overline{x}, (\overline{\pi}, \overline{\mu})$ are the final solution and dual vectors when the algorithm terminates in Step 13, then clearly $\overline{x}$ is optimal to (10.55), and $(\overline{\pi}, \overline{\mu})$ is optimal to its dual (10.57) for the classes **MB, CB** as at the termination of the algorithm. It should be noted that (10.55) depends on the choice of the classes **MB, CB**, and thus is not necessarily equivalent to (10.6), (10.7). We will now prove that $\overline{x}$ is optimal to (10.6), (10.7) even though (10.55) and (10.6), (10.7) are not necessarily equivalent.

THEOREM 10.25 *Let x be an integer feasible solution of (10.6). Let $\mathbf{Y}_\sigma \subset \mathcal{N}$ with $|\mathbf{Y}_\sigma| \geqq 3$ and odd. Then either $\mathbf{Y}_\sigma^-(x) \leqq (|\mathbf{Y}_\sigma| - 1)/2$, or $\mathbf{Y}_\sigma^+(x) \geqq (|\mathbf{Y}_\sigma| + 1)/2$.*

Proof Suppose there is an integer feasible solution x of (10.6) which violates the hypothesis of the theorem. Then $\mathbf{Y}_\sigma^-(x) > (|\mathbf{Y}_\sigma| - 1)/2$, $\mathbf{Y}_\sigma^+(x) < (|\mathbf{Y}_\sigma| + 1)/2$. But $\mathbf{Y}_\sigma^-(x) \leqq \mathbf{Y}_\sigma^+(x)$. Hence $(|\mathbf{Y}_\sigma| + 1)/2 > \mathbf{Y}_\sigma^+(x) \geqq \mathbf{Y}_\sigma^-(x) > (|\mathbf{Y}_\sigma| - 1)/2$, a contradiction since all these quantities are integers. ∎

THEOREM 10.26 *Let B be a pseudonode formed during the algorithm, associated with the pseudonode price $\mu(B)$. Suppose this pseudonode B itself gets absorbed inside another pseudonode B_1. Let $(\pi', \mu'), (\tilde{\pi}, \tilde{\mu})$ be the dual solutions at the beginning, and at the end respectively, of this dormant period for pseudonode B. Then $\mu'(B) = \tilde{\mu}(B)$, and $\pi'_i = \tilde{\pi}_i$ for all original nodes i inside B.*

Proof During this entire dormant period, the pseudonode price of B does not change at all and so it remains equal to $\mu'(B)$. So, $\mu'(B) = \tilde{\mu}(B)$

The pseudonode price of any newly formed pseudonode is 0 just when it is formed. So $\mu'(B_1) = 0$, at the beginning of this dormant period. As long as B_1 remains current, the change in $\mu(B_1)$ is (-2) times the change in π_i for original nodes i contained inside it, in any dual solution change step. So, for any original node i contained inside B, the net effect on π_i of all the dual solution change steps during the time that B_1 remains a current node, is $(-1/2)$ times the net effect on $\mu(B_1)$ in the same steps. And B_1 will not be unshrunk until its pseudonode price becomes 0 again. So, for $i \in B$, the net effect on π_i

of all the dual solution change steps during the time that B_1 remains a current node from the time it is formed to the time it is unshrunk again, is 0.

It is possible that B_1 itself gets absorbed inside another pseudonode B_2 before it is unshrunk. As long as B_2 remains current, the value of $\mu(B_1)$ remains unchanged and B_1 will not be unshrunk until B_2 is unshrunk at some stage first. Using the same arguments as above, it can be verified that for all original nodes $i \in B$, the net effect on π_i, of all the dual solution change steps during the time that B_2 remains a current node from the time it is formed to the time it is unshrunk again, is 0. It is possible that B_2 itself goes into dormancy and gets absorbed into another pseudonode B_3. The same argument can be applied again.

The dormant period for pseudonode B ends only when every pseudonode containing B inside it, formed since the beginning of this period, is unshrunk and B becomes a current node again. By repeating the above argument for every pseudo- node containing B inside it, formed during this period and unshrunk, we conclude that the net effect of all the dual solution change steps carried out during this dormant period on π_i for $i \in B$ is 0. So, $\pi'_i = \tilde{\pi}_i$ for all $i \in B$ ∎

THEOREM 10.27 *Let $(\pi = (\pi_i), \mu)$ be the dual solution at some stage of the algorithm. Suppose pseudonode B is a current node at this stage, containing some nodes from both the sets $\mathcal{N}^{\leqq}$ and $\mathcal{N}^{\geqq}$. For $j \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq}$ contained inside B, let f_j denote the node price of j at the time that pseudonode B was just formed for the last time. Also, let p be the BCI node of pseudonode B. Let $\mu(B)$ be the pseudonode price of B in the present dual solution. Then for all $j \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq}$ inside B, and for p we have*

$$\begin{array}{rrcl}
\textbf{(i)} & f_j & = & \pi_j + \frac{1}{2}\mu(B) \\
\textbf{(ii)} & |f_p| & = & |\pi_p| + \frac{1}{2}|\mu(B)| \\
\textbf{(iii)} & |f_j| & \geqq & \frac{1}{2}|\mu(B)| \\
\textbf{(iv)} & |f_p| & \geqq & |\pi_p| \\
\textbf{(v)} & |\pi_p| & = & \textit{Min. } \{|\pi_i| : i \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq} \textit{ inside } B\}
\end{array}$$

Proof While B is a current node, whenever Step 12 is carried out, the change in the pseudonode price of B is (-2) times the change in the node price for original nodes inside B. Also, the pseudonode price associated with B is 0 just when it was formed. Hence (i) holds.

Since p is the BCI node of B, p is inside B and $f_p, \pi_p, \mu(B)$ all have the same sign. From these, and by applying (i) to p, we get (ii).

By the definition of the BCI node, we have $|f_j| \geqq |f_p|$ for all nodes $j \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq}$ inside B. Using this and (ii) leads to (iii).

(iv) follows from (ii).

From the definition of the BCI node, we have $|f_p| =$ min. $\{|f_i| : i \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq}$ inside $B\} =$ min. $\{|\pi_i + \frac{1}{2}\mu(B)| : i \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq}$ inside B $\}$ by (i). So, by (ii), and since π_p and $\mu(B)$ have the same sign, $|\pi_p + \frac{1}{2}\mu(B)|$ $=$ min. $\{|\pi_i + \frac{1}{2}\mu(B)| : i \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq}$ inside B $\}$. This implies (v). ∎

THEOREM 10.28 *Let (π, μ) be the dual solution at some stage of the algorithm. Suppose pseudonode B containing some original nodes from both $\mathcal{N}^{\leqq}, \mathcal{N}^{\geqq}$, with BCI node p is dormant at this stage. For $j \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq}$ contained inside B let $f_j[g_j]$ be the node price of j at the time that B was just formed [entered dormancy] for the last time. Let $\mu(B)[\mu'(B)]$ be the pseudonode price associated with B at present [at the beginning of this dormant period]. Then*

$$\begin{array}{rrcl}
\textbf{(i)} & \mu(B) & = & \mu'(B) \\
\textbf{(ii)} & f_j & = & g_j + \frac{1}{2}\mu(B) \textit{ for all original nodes } j \textit{ inside } B \\
\textbf{(iii)} & |f_p| & = & |g_p| + \frac{1}{2}|\mu(B)| \\
\textbf{(iv)} & |f_j| & \geqq & \frac{1}{2}|\mu(B)|, \textit{ for } j \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq} \textit{ inside } B \\
\textbf{(v)} & |g_p| & = & \textit{min. } \{|g_j| : j \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq} \textit{ inside } B\}
\end{array}$$

Proof (i) follows because the pseudonode price of any pseudonode does not change at all during dormancy.

By Theorem 10.26, and the manner in which the dual solution is updated in Step 12, the net effect of all the dual solution change steps on the node price for original nodes i inside B is $(-1/2)$ times the net effect on the pseudonode price of B in the same steps. Also, the pseudonode price of B was 0 when it was just formed. These facts imply (ii).

(iii) follows by applying (ii) for $j = p$, because $f_p, g_p, \mu(B)$ all have the same sign.

From (iii) we have $|f_p| \geqq \frac{1}{2}|\mu(B)|$. Also, by the definition of the BCI node we have $|f_j| \geqq |f_k|$ for all $j \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq}$ inside B. These two together imply (iv).

By the definition of the BCI node we have $|f_p| = \text{min.} \ \{ \ |f_j| : j \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq}$ and inside B $\}$. Using (ii) in this we get $|g_p + \frac{1}{2}\mu(B)| = \text{min.} \ \{ \ |g_j + \frac{1}{2}\mu(B)| : j \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq}$ and inside B $\}$. And from (iv) we have

$$|g_j + \frac{1}{2}\mu(B)| \begin{cases} = |g_j| + \frac{1}{2}|\mu(B)|, & \text{if } g_j \text{ and } \mu(B) \text{ have the same sign} \\ \leqq |g_j| + \frac{1}{2}|\mu(B)|, & \text{if } g_j \text{ and } \mu(B) \text{ have opposite signs} \end{cases}$$

Since g_p and $\mu(B)$ have the same sign, we have $|g_p + \frac{1}{2}\mu(B)| = |g_p| + \frac{1}{2}|\mu(B)|$. These facts together imply (v). ∎

THEOREM 10.29 *Let π, μ be the dual solution at some stage of the algorithm. For each dormant original node $i \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq}$, let y_i be the node price associated with it at the time that it became a noncurrent node for the last time. Then $|y_i| \geqq \frac{1}{2} \sum(|\mu_\sigma|$: over σ such that $i \in \mathbf{Y}_\sigma$).*

Proof Let $B_1, \ldots, B_r$ be the sequence of all the shrunken blossoms containing i inside them, where i is inside B_1, and B_t is inside B_{t+1} for each $t = 1$ to $r - 1$, and B_r is a current node. Let $\mu_1, \ldots, \mu_r$ be the pseudonode prices of $B_1, \ldots, B_r$ at this stage. Let f_i^t be the node price of i when B_t was just shrunk for the last time.

If none of the pseudonodes containing node i inside it, have a BCI node defined for them, the result in this theorem follows directly from the manner in which the dual solution is updated whenever Step 12 is carried out in this algorithm, because the original node price π_i of i, and the pseudonode price μ_σ of any pseudonode containing node i inside it, always have the same sign in this case.

Now consider the case where the lowest level pseudonode containing node i, B_1, itself has a BCI node defined for it. In this case, all the pseudonodes $B_1, \ldots, B_r$ have BCI nodes defined for them. Let j be any original node inside the current pseudonode B_r. The proof in this case is by induction. We now set up an induction hypothesis.

Induction Hypothesis For some u satisfying $1 < u \leqq r$, we have $|f_j^s| \geqq \frac{1}{2}(|\mu_r| + |\mu_{r-1}| + \ldots, |\mu_s|)$ for all $u \leqq s \leqq r$ and for all $j \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq}$ inside B_s.

The induction hypothesis holds for $u = r$ by (iii) of Theorem 10.27. We will now prove that under the induction hypothesis, the statement in it must also hold for $s = u - 1$. Let p be the BCI node of B_{u-1}. By (i) and (iii) of Theorem 10.28, we have $|f_p^{u-1}| = |f_p^u| + \frac{1}{2}|\mu_{u-1}|$. Since p is the BCI node of B_{u-1}, we have for all $j \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq}$ inside B_{u-1}

$$|f_j^{u-1}| \geqq |f_p^{u-1}| \geqq |f_p^u| \geqq \frac{1}{2}(|\mu_r| + \ldots + |\mu_u|)$$

This shows that if the induction hypothesis holds for u where $1 < u \leqq r$, then it also holds for $u - 1$. So, by induction it must hold for $u = 1$ too. But by definition, $y_i = f_i^1$ for all $i \in \mathcal{N}^{\leqq} \cup \mathcal{N}^{\geqq}$ inside B_1, and so the result in the theorem follows in this case by applying the statement in the induction hypothesis for $u = 1$.

Now consider the case where some of the pseudonodes containing node i inside them do not have a BCI node defined for them, and some of the others do. In this case, the proof of the result in the theorem can be accomplished by combining the arguments in the two cases discussed above. ∎

THEOREM 10.30 *Let $\pi' = (\pi_i'), \mu' = (\mu_\sigma')$ be the dual feasible solution at some stage of the algorithm. For every original node i inside a pseudonode (dormant or current) at this stage, let y_i be the original node price of i in the step that node i entered dormancy for the last time in the algorithm. Then $y_i = \pi_i' + \frac{1}{2}\sum(\mu_\sigma'$: over σ such that $i \in \mathbf{Y}_\sigma)$.*

Proof For a dormant original node i, whenever Step 12 occurs, the change in the node price π_i is $(-1/2)$ times the change in the pseudonode price of the outermost pseudonode containing i, and the pseudonode price of any other dormant pseudonodes containing i at that stage does not change at all in that step. The theorem follows from this fact. ∎

THEOREM 10.31 *Let $x', (\pi' = (\pi_i'), \mu' = (\mu_\sigma'))$ be feasible solutions to (10.55), (10.57) respectively when the blossom constraint specification classes are chosen as $\mathbf{MB}', \mathbf{CB}'$. Let B be a pseudonode at* *this stage, either current or dormant, with* $\mathbf{Y}_{\sigma_1}$ *as the set of original nodes inside it, with a BCI node. Suppose we move* $\mathbf{Y}_{\sigma_1}$ *from the class among* $\mathbf{MB}', \mathbf{CB}'$ *in which it is contained, into the other class. Let* $\mathbf{MB}'', \mathbf{CB}''$ *be the resulting blossom constraint specification classes. Define* $\pi'' = (\pi''_i), \mu'' = (\mu''_\sigma)$ *where* $\pi''_i = \pi'_i + \mu'_{\sigma_1}$ *for all* $i \in \mathbf{Y}_{\sigma_1}$*, and* $= \pi'_i$ *for all* $i \notin \mathbf{Y}_{\sigma_1}$*;* $\mu''_\sigma = -\mu'_{\sigma_1}$ *for* $\sigma = \sigma_1$*, and* $= \mu'_\sigma$ *for* $\sigma \neq \sigma_1$*. Then* (π'', μ'') *is feasible to the modified dual (10.57) corresponding to blossom constraint specification classes* $\mathbf{MB}'', \mathbf{CB}''$*, and* $W(\pi', \mu') = W(\pi'', \mu'')$*.*

Proof Clearly μ''_σ satisfies the sign restrictions for dual feasibility for all σ. Original node prices outside $\mathbf{Y}_{\sigma_1}$ keep their values unchanged and hence continue to satisfy the sign restrictions for dual feasibility. It remains to be proved that $\pi''_i \leqq 0$ for $i \in \mathcal{N}^{\leqq} \cap \mathbf{Y}_{\sigma_1}$ and $\pi''_i \geqq 0$ for $i \in \mathcal{N}^{\geqq} \cap \mathbf{Y}_{\sigma_1}$. Define y_i for $i \in \mathbf{Y}_{\sigma_1}$ as in Theorem 10.30. Then by Theorems 10.27, 10.28, 10.30, for $i \in \mathbf{Y}_{\sigma_1}$, $y_i = \pi'_i + \frac{1}{2}\sum(\mu'_\sigma$: over σ such that $i \in \mathbf{Y}_\sigma) = (\pi'_i + \mu'_{\sigma_1}) + \frac{1}{2}(\sum((\mu'_\sigma$: over σ such that $i \in \mathbf{Y}_\sigma)) -2\mu'_{\sigma_1}) = \pi''_i + \frac{1}{2}\sum(\mu''_\sigma$: over σ such that $i \in \mathbf{Y}_\sigma)$. So, $\pi''_i = y_i - \frac{1}{2}\sum(\mu''_\sigma$: over σ such that $i \in \mathbf{Y}_\sigma)$. Since dual feasibility is maintained during the algorithm, we have $y_i \leqq 0$ for $i \in \mathcal{N}^{\leqq} \cap \mathbf{Y}_{\sigma_1}$ and $y_i \geqq 0$ for $i \in \mathcal{N}^{\geqq} \cap \mathbf{Y}_{\sigma_1}$. This together with the result in Theorem 10. 29, and the formula derived above for π''_i implies that $\pi''_i \leqq 0$ for $i \in \mathcal{N}^{\leqq} \cap \mathbf{Y}_{\sigma_1}$ and $\pi''_i \geqq 0$ for $i \in \mathcal{N}^{\geqq} \cap \mathbf{Y}_{\sigma_1}$, establishing that (π'', μ'') satisfies the sign restrictions for dual feasibility.

It can be verified that $d_{ij}(\pi'', \mu'') = d_{ij}(\pi', \mu')$ for all $(i; j) \in \mathcal{A}$. From this and the above result, it follows that (π'', μ'') is feasible to the modified dual corresponding to the blossom constraint specification classes $\mathbf{MB}'', \mathbf{CB}''$.

From the fact that $d_{ij}(\pi', \mu') = d_{ij}(\pi'', \mu'')$ for all $(i; j) \in \mathcal{A}$, and the definition of π'', μ'', it easily follows that $W(\pi'', \mu'') = W(\pi', \mu')$. ∎

THEOREM 10.32 *If the algorithm discussed above terminates in Step 13,* $\overline{x}$*, the edge vector at termination, is a minimum cost 1-M/EC vector in* G*.*

Proof Let $(\overline{\pi}, \overline{\mu})$ be the dual solution at termination. Let $\sigma = 1$ to L_1 correspond to all the blossoms at all levels that are in existence at termination. Of those, let $\sigma = 1$ to L_2 refer to subsets of $\mathcal{N}^{\leqq} \cup \mathcal{N}^{=}$ or $\mathcal{N}^{\geqq} \cup \mathcal{N}^{=}$ for which the type of blossom constraint (matching or covering) is known, and let $\sigma = L_2 + 1$ to L_1 correspond to subsets of nodes containing at least one node from each of $\mathcal{N}^{\leqq}$ and $\mathcal{N}^{\geqq}$.

Let $\mathbf{Y}_\sigma$, $\sigma = L_1 + 1$ to L be all the other odd subsets of $\mathcal{N}\backslash\mathcal{N}^0$ of cardinality $\geqq 3$, the blossom constraint corresponding to which do not appear at the termination of the algorithm. As discussed earlier, every 1-M/EC vector has to satisfy either the matching or the covering blossom constraint, or possibly both, corresponding to $\mathbf{Y}_\sigma$ for every σ $= 1$ to L. Because of this, in formulating the modified problem (10.55), if we eliminate all the blossom constraints (both matching and covering type) corresponding to $\mathbf{Y}_\sigma$ for $\sigma = L_1+1$ to L, and yet obtain an integer feasible solution for the resulting modified problem, then that integer solution must be an optimum solution of the original problem (10.6), (10.7).

Let $\overline{\mathbf{MB}}, \overline{\mathbf{CB}}$ be the blossom constraint specification classes at termination. With this specification, $\overline{x}$ is an optimum solution to the modified problem (10.55) as discussed earlier, and it is an integer vector. So, $\overline{x}$ is a 1-M/EC vector and its cost $z(\overline{x}) = W(\overline{\pi}, \overline{\mu})$ by the duality theorem of LP. Also, by applying Theorem 10.31 repeatedly, we see that if the blossom constraint specifications are given by any sets **MB, CB** obtained from $\overline{\mathbf{MB}}, \overline{\mathbf{CB}}$ by moving some of the $\mathbf{Y}_\sigma$ for σ between L_2+1 to L_1 that are in $\overline{\mathbf{MB}}$ into the set $\overline{\mathbf{CB}}$ and vice versa, then the dual of the corresponding modified problem, has a dual feasible solution (π, μ) satisfying $W(\pi, \mu) = W(\overline{\pi}, \overline{\mu})$. This, by the weak duality theorem of LP implies that the optimum objective value in the corresponding modified problem for that blossom constraint specification, is $\geqq W(\overline{\pi}, \overline{\mu}) = z(\overline{x})$. Hence $\overline{x}$ gives the minimum value for $z(x)$ among all the integer feasible solutions of modified problems given by such blossom constraint specifications. This and the earlier arguments clearly imply that $\overline{x}$ is the optimum solution of (10.6), (10.7), i.e., a minimum cost 1-M/EC vector in G. ∎

The Infeasibility Conclusion

THEOREM 10.33 *At each execution of Step 12 in the blossom al-*

gorithm discussed in this section, the dual objective value $W(\pi, \mu)$ increases by δ times the number of exposed nodes at that stage.

Proof Let (π, μ) be the dual feasible solution when the algorithm arrives at an occurrence of Step 12, and $(\hat{\pi} = (\hat{\pi}_i), \hat{\mu} = (\hat{\mu}_\sigma))$ the new dual feasible solution at the end of this step. Let $\mathbf{Y}_\sigma$ be the set of original nodes inside a current pseudonode at this stage. Let $a_\sigma = (|\mathbf{Y}_\sigma| - 1)/2$ if $\mathbf{Y}_\sigma \in \mathbf{MB}$, or $(|\mathbf{Y}_\sigma| + 1)/2$ if $\mathbf{Y}_\sigma \in \mathbf{CB}$. Then

$$\begin{aligned}
&(a_\sigma \hat{\mu}_\sigma + \textstyle\sum(\hat{\pi}_i : \text{ over } i \in \mathbf{Y}_\sigma)) - (a_\sigma \mu_\sigma + \textstyle\sum(\pi_i : \text{ over } i \in \mathbf{Y}_\sigma)) \\
&= a_\sigma(\hat{\mu}_\sigma - \mu_\sigma) + \textstyle\sum(\hat{\pi}_i - \pi_i : \text{ over } i \in \mathbf{Y}_\sigma) \\
&= \begin{cases} \delta, & \text{if the pseudonode is outer} \\ -\delta, & \text{if the pseudonode is inner} \\ 0, & \text{if the pseudonode is unlabeled} \end{cases}
\end{aligned}$$

Since $d_{ij}(\pi, \mu) \overset{\leq}{=} c_{ij}$ for all $(i; j) \notin \mathcal{A} \backslash \mathcal{A}^-$, min. $\{0, c_{ij} - d_{ij}(\pi, \mu)\} \neq 0$ only for edges $(i; j) \in \mathcal{A}^-$, and these edges are always current edges, with $\pi_p = \hat{\pi}_p = 0$ for all nodes p on these edges. Hence the term $\sum$(min. $\{0, c_{ij} - d_{ij}(\pi, \mu)\}$: over $(i; j) \in \mathcal{A}$) remains unchanged during any execution of Step 12.

Also $\hat{\mu}_\sigma = \mu_\sigma$ if $\mathbf{Y}_\sigma$ is the set of original nodes in a dormant pseudonode at this stage. The number of outer current nodes at this stage is clearly = the number of exposed nodes + the number of inner current nodes, because of the Hungarian forest conditions holding at the beginning of Step 12. These facts imply the result in this theorem. ∎

THEOREM 10.34 *If the value of δ turns out to be $+\infty$ in a dual solution change step during this algorithm, there exists no 1-M/EC in* G *with the given partition of $\mathcal{N}$.*

Proof Let (π, μ) be the dual solution at the beginning of that dual solution change step, and **MB, CB** the classes giving the blossom constraint specifications at that stage. Let $\sigma = 1$ to L_1 correspond to all the blossoms at all levels that are in existence at this time. Of these, let $\sigma = 1$ to L_2 refer to subsets of $\mathcal{N}^{\overset{\leq}{=}} \cup \mathcal{N}^{=}$ or $\mathcal{N}^{\overset{\geq}{=}} \cup \mathcal{N}^{=}$ for which the type of blossom constraint (matching or covering) is known, and let $\sigma = L_2 + 1$ to L_1 correspond to subsets of nodes containing at least one node from each of $\mathcal{N}^{\leqq}$ and $\mathcal{N}^{\geqq}$. So, $\mathbf{MB} \cup \mathbf{CB}$ contain all the $\mathbf{Y}_\sigma$ for $\sigma = 1$ to L_1.

Define $\hat{\pi}(\lambda) = (\hat{\pi}_i(\lambda)), \hat{\mu}(\lambda) = (\hat{\mu}_\sigma(\lambda))$ by

$$\hat{\pi}_i(\lambda) = \begin{cases} \pi_i + \lambda & \text{if } i \text{ is an outer current node, or is inside an outer current pseudonode in } \mathbf{MB}\text{, or is inside an inner current pseudonode in } \mathbf{CB} \\ \pi_i - \lambda & \text{if } i \text{ is an inner current node, or is inside an inner current pseudonode in } \mathbf{MB}\text{, or is inside an outer current pseudonode in } \mathbf{CB} \\ \pi_i & \text{if } i \text{ is inside an unlabeled node} \end{cases}$$

$$\hat{\mu}_\sigma(\lambda) = \begin{cases} \mu_\sigma + 2\lambda & \text{if } \mathbf{Y}_\sigma \in \mathbf{MB} \text{ forms a current inner labeled pseudonode, or if } \mathbf{Y}_\sigma \in \mathbf{CB} \text{ forms a current outer labeled pseudonode} \\ \mu_\sigma - 2\lambda & \text{if } \mathbf{Y}_\sigma \in \mathbf{MB} \text{ forms a current outer labeled pseudonode, or if } \mathbf{Y}_\sigma \in \mathbf{CB} \text{ forms a current inner labeled pseudonode} \\ \mu_\sigma & \text{otherwise} \end{cases}$$

Then it can be verified that $(\hat{\pi}(\lambda), \hat{\mu}(\lambda))$ is feasible to (10.57) with the blossom constraint specification classes **MB, CB** as at present, for all $\lambda \geqq 0$, and that $W(\hat{\pi}(\lambda), \hat{\mu}(\lambda)) = W(\pi, \mu) + \lambda\gamma$, where $\gamma =$ number of exposed nodes at this stage. Since $\gamma \geqq 1$ (by the Hungarian forest conditions there exists at least one exposed node at this stage) as $\lambda \to +\infty, W(\hat{\pi}(\lambda), \hat{\mu}(\lambda)) \to +\infty$. So, by the duality theory of LP (10.55) is infeasible with the present blossom constraint specification classes **MB, CB**. Using the arguments in the proof of Theorem 10.31, it can be shown that if $\overline{\mathbf{MB}}, \overline{\mathbf{CB}}$ is a blossom constraint specification obtained from **MB, CB** by moving some of the $\mathbf{Y}_\sigma$ for σ between L_2+1 to L_1 that were in **MB** into the set **CB** and vice versa, the same conclusion holds. As discussed earlier, if a 1-M/EC vector exists in G, it must satisfy either the matching or the covering blossom constraint corresponding to every $\sigma = L_2+1$ to L_1, including the other constraints in (10.55). So, the conclusion that (10.55) remains infeasible even when some of the $\mathbf{Y}_\sigma$ for σ between L_2+1 to L_1 that were in **MB** are moved into **CB** or vice versa, implies that there are no 1-M/EC vectors in G. ∎

Exercises

10.4 Specialize the blossom algorithms discussed in Sections 10.2.1, and this section to the case when G is a bipartite network.

10.5 Let $\underline{r}, \overline{r}$ denote the minimum and maximum cardinalities for a 1-M/EC in G $= (\mathcal{N}, \mathcal{A})$ with $(\mathcal{N}^{\leqq}, \mathcal{N}^{=}, \mathcal{N}^{\geqq}, \mathcal{N}^{0})$ as the partition of $\mathcal{N}$. Prove that there exists a 1-M/EC of cardinality r in G for every $\underline{r} \leqq r \leqq \overline{r}$. (Cartensen, Murty, and Perin [1981])

A FORTRAN implementation of the blossom algorithm in this section took on an average 0.333 CPU seconds (on an AMDAHL 470/V6 computer in 1980) to solve minimum cost 1-M/EC problems with randomly generated data in networks with 50 nodes and 250 arcs, and 4.127 seconds in networks with 100 nodes and 3000 arcs (see Perin [1980]).

The Convex Hull of 1-M/EC Vectors

We developed an algorithm for solving the minimum cost 1-M/EC problem in G $= (\mathcal{N}, \mathcal{A}, c)$ with $(\mathcal{N}^{\leqq}, \mathcal{N}^{=}, \mathcal{N}^{\geqq}, \mathcal{N}^{0})$ as the partition of $\mathcal{N}$, but we did not explicitly determine a system of linear constraints whose solution set is the convex hull of all 1-M/EC vectors in G. We will do that now.

Let $\mathbf{Y} \subset \mathcal{N} \backslash \mathcal{N}^0$ with $|\mathbf{Y}|$ odd and $\geqq 3$, and let $\overline{\mathbf{Y}} = \mathcal{N} \backslash \mathbf{Y}$. For any edge vector $x = (x_{ij})$ defined on $\mathcal{A}$ define $x(\mathbf{Y}; \overline{\mathbf{Y}})$ as in Chapter 1 to be the sum of x_{ij} over edges $(i; j)$ joining nodes in $\mathbf{Y}$ to those in $\overline{\mathbf{Y}}$. Verify that $x(\mathbf{Y}; \overline{\mathbf{Y}}) = \mathbf{Y}^{+}(x) - \mathbf{Y}^{-}(x)$. If $\mathbf{E} = \{(i; j) : x_{ij} = 1\}$, then $x(\mathbf{Y}; \overline{\mathbf{Y}}) =$ the number of edges in $\mathbf{E}$ joining a node in $\mathbf{Y}$ to one outside of $\mathbf{Y}$.

Let $s(i)$ be the slack variable corresponding to node i in the node degree constraints in 1-M/EC problems, i.e., for any 1-M/EC vector x

$$s(i) = \begin{cases} 1 - x(i) & \text{for } i \in \mathcal{N}^{\leqq} \\ 0 & \text{for } i \in \mathcal{N}^{=} \\ x(i) - 1 & \text{for } i \in \mathcal{N}^{\geqq} \\ 0 & \text{for } i \in \mathcal{N}^{0} \end{cases}$$

Clearly $s(i) \geqq 0$ whenever x is a 1-M/EC vector.

THEOREM 10.35 *Let* $\mathbf{Y} \subset \mathcal{N} \backslash \mathcal{N}^0$ *with* $|\mathbf{Y}|$ *odd and* $\geqq 3$*, and let* $\overline{\mathbf{Y}} = \mathcal{N} \backslash \mathbf{Y}$*. Let* $x, s = (s(i))$ *be a 1-M/EC vector and the corresponding vector of slack variables defined above. Then*

$$x(\mathbf{Y}; \overline{\mathbf{Y}}) + \sum (s(i) : \textit{ over } i \in \mathbf{Y}) \geqq 1 \tag{10.65}$$

Proof Let $\mathbf{E}$ be the 1-M/EC corresponding to x. Since $x(\mathbf{Y}; \overline{\mathbf{Y}})$ and $s(i)$ are nonnegative, the left hand side of (10.65) is $\geqq 0$. So, if (10.65) is violated, we must have $x(\mathbf{Y}; \overline{\mathbf{Y}}) = 0$, and $s(i) = 0$ for all $i \in \mathbf{Y}$. $x(\mathbf{Y}; \overline{\mathbf{Y}}) = 0$ implies that $\mathbf{E}$ contains no edges joining a node in $\mathbf{Y}$ to one outside it. $s(i) = 0$ for all $i \in \mathbf{Y} \subset \mathcal{N} \backslash \mathcal{N}^0$ implies that each node in $\mathbf{Y}$ is incident to exactly one edge in $\mathbf{E}$, and by the above, any such edge must join two nodes in $\mathbf{Y}$. Since $|\mathbf{Y}|$ is odd, this is impossible, hence (10.65) must hold. ∎

THEOREM 10.36 *If* x *is the 1-M/EC vector in* G *obtained when the minimum cost 1-M/EC problem is solved by the algorithm discussed above, and* $\mathbf{Y}$ *is the set of original nodes inside a pseudonode at the termination of the algorithm, then (10.65) will hold as an equation for* x *and* $\mathbf{Y}$.

Proof Let $s = (s(i))$ be the slack vector corresponding to x. If the pseudonode is type A or B, $x(\mathbf{Y}; \overline{\mathbf{Y}}) = 1$ and $s(i) = 0$ for all $i \in \mathbf{Y}$, so (10.65) holds. If the pseudonode is type C, D, or E, let p be its apex node, then $x(\mathbf{Y}; \overline{\mathbf{Y}}) = 0, s(p) = 1$, and $s(i) = 0$ for all $i \in \mathbf{Y}, i \neq p$, hence (10.65) holds again. Since every pseudonode at termination will be type A, B, C, D, or E if 1-M/EC vectors exist, this completes the proof of the theorem. ∎

We will show that (10.65) is in fact the blossom inequality for the 1-M/EC problem corresponding to the odd subset of nodes $\mathbf{Y}$ from $\mathcal{N}\backslash\mathcal{N}^0$. Define $a(\mathbf{Y}) = 1 + |\mathbf{Y} \cap \mathcal{N}^{\geqq}| - |\mathbf{Y} \cap \mathcal{N}^{\leqq}|$, and $g(x, \mathbf{Y}) = x(\mathbf{Y}; \overline{\mathbf{Y}}) + \sum(x(i) : \text{over } i \in \mathbf{Y} \cap \mathcal{N}^{\geqq}) - \sum(x(i) : \text{over } i \in \mathbf{Y} \cap \mathcal{N}^{\leqq})$. Then after rearranging terms, (10.65) can be written as

$$g(x, \mathbf{Y}) \geqq a(\mathbf{Y}) \tag{10.66}$$

(10.66), or the equivalent (10.65) is the blossom constraint corresponding to the odd subset of nodes $\mathbf{Y}$ for the 1-M/EC problem. In this problem, there is one blossom constraint of this type, for each subset of $\mathcal{N}\backslash\mathcal{N}^0$ of odd cardinality $\geqq 3$. Let $\mathbf{Y}_\sigma, \sigma = 1$ to L denote all the subsets of $\mathcal{N}\backslash\mathcal{N}^0$ of odd cardinality $\geqq 3$. Consider the LP (10.67) given below, obtained from the minimum cost 1-M/EC problem (10.6), (10.7) by replacing the integer requirements on the variables by the blossom inequalities of the form (10.66).

$$\begin{array}{rcl} \text{Minimize} \quad \sum(c_{ij}x_{ij} & : & \text{over } (i;j) \in \mathcal{A}) \\ \text{Subject to} \quad x(i) & & \begin{cases} \leqq 1, \text{ for } i \in \mathcal{N}^{\leqq} \\ = 1, \text{ for } i \in \mathcal{N}^{=} \\ \geqq 1, \text{ for } i \in \mathcal{N}^{\geqq} \end{cases} \\ g(x, \mathbf{Y}_\sigma) & \geqq & a(\mathbf{Y}_\sigma), \sigma = 1 \text{ to } L \\ 0 \leqq x_{ij} & \leqq & 1 \text{ for all } (i;j) \in \mathcal{A} \end{array} \tag{10.67}$$

To write the dual of (10.67), associate the dual variable ξ_i to the node constraint at node i (with ξ_i defined to be $= 0$ for all $i \in \mathcal{N}^0$ always), η_σ to the blossom constraint corresponding to $\mathbf{Y}_\sigma$, and ω_{ij}, ν_{ij} to the bounds on x_{ij}. Let $\xi = (\xi_i), \eta = (\eta_\sigma)$. The dual of (10.67) is

$$\begin{array}{rl} \text{Max.} & \sum_{i \in \mathcal{N}} \xi_i + \sum_{\sigma=1}^{L} a(\mathbf{Y}_\sigma)\eta_\sigma + \sum_{(i;j) \in \mathcal{A}} \omega_{ij} \\ \text{subject to} & \delta_{ij}(\xi, \eta) + \omega_{ij} + \nu_{ij} = c_{ij}, \quad \text{for} \quad (i;j) \in \mathcal{A} \\ & \xi_i \quad \begin{cases} \leqq 0, & \text{for} \quad i \in \mathcal{N}^{\leqq} \\ \geqq 0, & \text{for} \quad i \in \mathcal{N}^{\geqq} \\ = 0, & \text{for} \quad i \in \mathcal{N}^0 \end{cases} \end{array} \tag{10.68}$$

$$\eta_\sigma \geqq 0, \sigma = 1 \text{ to } L, \text{ and } \omega_{ij} \leqq 0, \nu_{ij} \geqq 0 \text{ for } (i;j) \in \mathcal{A}$$

where for $(i;j) \in \mathcal{A}$

$$b_{ij}(\mathbf{Y}_\sigma) = \begin{cases} 2 & \text{if both } i,j \in \mathcal{N}^{\geqq} \cap \mathbf{Y}_\sigma\text{, or if one of } i \text{ or } j \text{ is } \in \mathcal{N}^{\geqq} \cap \mathbf{Y}_\sigma\text{, and the other } \notin \mathbf{Y}_\sigma \\ 1 & \text{if one of } i,j \text{ is } \in \mathcal{N}^{=} \cap \mathbf{Y}_\sigma\text{, and the other is either } \in \mathcal{N}^{\geqq} \cap \mathbf{Y}_\sigma\text{, or } \notin \mathbf{Y}_\sigma \\ 0 & \text{if both } i,j \in \mathcal{N}^{=} \cap \mathbf{Y}_\sigma\text{; or if both } i,j \notin \mathbf{Y}_\sigma\text{; or if one of } i,j \text{ is } \in \mathcal{N}^{\geqq} \cap \mathbf{Y}_\sigma\text{, and the other is } \in \mathcal{N}^{\leqq} \cap \mathbf{Y}_\sigma\text{; or if one of } i,j \text{ is } \in \mathcal{N}^{\leqq} \cap \mathbf{Y}_\sigma\text{, and the other } \notin \mathbf{Y}_\sigma \\ -1 & \text{if one of } i,j \text{ is } \in \mathcal{N}^{\leqq} \cap \mathbf{Y}_\sigma\text{, and the other is } \in \mathcal{N}^{=} \cap \mathbf{Y}_\sigma \\ -2 & \text{if both } i,j \in \mathcal{N}^{\leqq} \cap \mathbf{Y}_\sigma \end{cases} \quad (10.69)$$

$$\delta_{ij}(\xi,\eta) = \xi_i + \xi_j + \sum_{\sigma=1}^{L} b_{ij}(\mathbf{Y}_\sigma)\eta_\sigma \quad (10.70)$$

From the structure of the dual problem (10.68), it is clear that at an optimum solution (ξ, η, ω, ν), we will have for each $(i;j) \in \mathcal{A}$

$$\omega_{ij} = \text{ min. } \{0, c_{ij} - \delta_{ij}(\xi,\eta)\}, \nu_{ij} = \text{ max. } \{0, c_{ij} - \delta_{ij}(\xi,\eta)\} \quad (10.71)$$

Using this, we can express the complementary slackness optimality conditions for the primal-dual pair (10.67), (10.68) in terms of x, ξ, η only. These are, for each $i \in \mathcal{N}$, $(i;j) \in \mathcal{A}$, and $\sigma = 1$ to L

$$\begin{aligned} \xi_i(x(i) - 1) &= 0 \\ \delta_{ij}(\xi,\eta) & \begin{cases} > c_{ij} \text{ implies } x_{ij} = 1 \\ < c_{ij} \text{ implies } x_{ij} = 0 \end{cases} \\ \eta_\sigma(g(x, \mathbf{Y}_\sigma) - a(\mathbf{Y}_\sigma)) &= 0 \end{aligned} \quad (10.72)$$

THEOREM 10.37 *Let $\tilde{x}, (\tilde{\pi}, \tilde{\mu}), \tilde{\mathbf{MB}}, \tilde{\mathbf{CB}}$ be the 1-M/EC vector, dual solution, and the blossom constraint specification sets obtained when the 1-M/EC problem in* G *is solved by the algorithm discussed above. So, $\tilde{\mu}_\sigma = 0$ if $\mathbf{Y}_\sigma \notin \tilde{\mathbf{MB}} \cup \tilde{\mathbf{CB}}$. Define $\tilde{\xi} = (\tilde{\xi}_i), \tilde{\eta} = (\tilde{\eta}_\sigma)$ where for $i \in \mathcal{N}, \sigma = 1$ to L*

$$\begin{aligned}
\tilde{y}_i &= \tilde{\pi}_i + \sum(\frac{1}{2}\tilde{\mu}_\sigma : \text{ over } \sigma \text{ s. t. } \mathbf{Y}_\sigma \in \tilde{\mathbf{MB}} \cup \tilde{\mathbf{CB}} \text{ and contains } i\ ) \\
\tilde{\eta}_\sigma &= \frac{1}{2}|\tilde{\mu}_\sigma| \\
\tilde{\xi}_i &= \begin{cases} \tilde{y}_i & \text{for } i \in \mathcal{N}^0 \cup \mathcal{N}^= \\ \tilde{y}_i + \sum(\tilde{\eta}_\sigma : \text{ over } \sigma \text{ s. t. } \mathbf{Y}_\sigma \text{ contains } i) & \text{for } i \in \mathcal{N}^{\leqq} \\ \tilde{y}_i - \sum(\tilde{\eta}_\sigma : \text{ over } \sigma \text{ s. t. } \mathbf{Y}_\sigma \text{ contains } i) & \text{for } i \in \mathcal{N}^{\geqq} \end{cases}
\end{aligned}$$

Then $\tilde{x}, (\tilde{\xi}, \tilde{\eta})$ are optimal to the primal dual pair of problems (10.67), (10.68); with the corresponding ω, ν given by (10.71).

Proof $\tilde{x}$ is clearly feasible to (10.67). $(\tilde{\xi}, \tilde{\eta})$ satisfy the sign constraints on them in (10.68) because of the results proved in Theorem 10.29. By considering the various cases corresponding to the nodes i, j on an arc $(i; j) \in \mathcal{A}$ lying in various subsets in the partition $(\mathcal{N}^{\leqq}, \mathcal{N}^=, \mathcal{N}^{\geqq}, \mathcal{N}^0)$ of $\mathcal{N}$ separately, it can be verified that $\delta_{ij}(\tilde{\xi}, \tilde{\eta}) = d_{ij}(\tilde{\pi}, \tilde{\mu})$ for all $(i; j) \in \mathcal{A}$.

By the complementary slackness optimality conditions (10.61) satisfied by $\tilde{x}, (\tilde{\pi}, \tilde{\mu})$, this implies that for each $(i; j) \in \mathcal{A}$, $\delta_{ij}(\tilde{\xi}, \tilde{\eta}) > c_{ij}$ implies that $\tilde{x}_{ij} = 1$; and $\delta_{ij}(\tilde{\xi}, \tilde{\eta}) < c_{ij}$ implies that $\tilde{x}_{ij} = 0$. From this, and Theorems 10.36, 10.29, it can be verified that $\tilde{x}, (\tilde{\xi}, \tilde{\eta})$ satisfy the complementary slackness optimality conditions (10.72); and since $\tilde{x}$ is feasible to (10.67), and $(\tilde{\xi}, \tilde{\eta})$ is feasible to its dual (10.68), we conclude that $\tilde{x}, (\tilde{\xi}, \tilde{\eta})$ are optimal to the primal dual pair of problems (10.67), (10.68). ∎

THEOREM 10.38 *The system of constraints in (10.67) provides a linear constraint representation for the convex hull of all the 1-M/EC vectors in* G.

Proof If 1-M/ECs exist in G, we have shown that for any cost vector c, the algorithm discussed in this section terminates with an $\tilde{x}$ which is a minimum cost 1-M/EC vector, and by Theorem 10.37, this $\tilde{x}$ is an optimum solution of (10.67). Hence, when 1-M/ECs exist in G, for every cost vector c, (10.67) has an optimum solution that is a 1-M/EC vector. And every 1-M/EC vector in G is feasible to (10.67).

When there exist no 1-M/ECs in G, we have shown that (10.57) is unbounded above, using this and the arguments in the proof of Theorem 10.37, it can be shown that the objective function in (10.68) is also unbounded above in this case, which implies that (10.67) is infeasible by the duality theorem of LP.

These facts imply that every extreme point of the set of feasible solutions of (10.67) is a 1-M/EC vector. Also, every 1-M/EC vector in G is a 0-1 vector which is feasible to (10.67), and since all the variables in (10.67) are bounded by 0 and 1, every 0-1 feasible solution for it is an extreme point of its set of feasible solutions. Thus every extreme point of the set of feasible solutions of (10.67) is a 1-M/EC vector in G and vice versa, so this set is the convex hull of all 1-M/EC vectors in G. ∎

Comment 10.4 The algorithm and the results in this section are taken from Perin [1980]. In it, he also derived an out-of-kilter type blossom algorithm for the 1-M/EC problem, and used it to develop efficient techniques for performing sensitivity analysis in the 1-M/EC problem.

10.4 The Minimum Cost 1-M/EC Problem with Specified Cardinality

Consider the undirected network G $= (\mathcal{N}, \mathcal{A}, c)$ with the partition $(\mathcal{N}^{\overset{\leq}{=}}, \mathcal{N}^{=}, \mathcal{N}^{\overset{\geq}{=}}, \mathcal{N}^{0})$ of $\mathcal{N}$, and $|\mathcal{N}| = n, |\mathcal{A}| = m$. Here we consider the problem of finding a minimum cost 1-M/EC in G having a specified cardinality r. This is the problem (10.6), (10.7), with the additional constraint $\sum(x_{ij}:$ over $(i;j) \in \mathcal{A}) = r$. Associate a single Lagrange multiplier λ to this constraint, and include it in the objective function.

This partial Lagrangian relaxation leads to the problem of finding a 1-M/EC in G which minimizes $(c - \lambda \mathbf{e}^T)x$, where $\mathbf{e}^T$ is the row vector in $\mathbb{R}^m$ all of whose entries are 1. When λ is given a specific value, this is a 1-M/EC problem that can be solved efficiently by the blossom algorithm of Section 10.3. Let $x(\lambda) = (x_{ij}(\lambda))$ denote an optimum 1-M/EC vector for this problem as a function of λ. Let $r(\lambda) = \sum(x_{ij}(\lambda)$: over $(i; j) \in \mathcal{A})$, the cardinality of $x(\lambda)$. Then it can be shown that $x(\lambda)$ is a minimum cost 1-M/EC vector when the specified cardinality $r = r(\lambda)$, and that $r(\lambda)$ increases with λ. Using this result, an algorithm for solving the specified cardinality minimum cost 1-M/EC problem, treating the cardinality, r, as an integer valued parameter, has been developed in Carstensen, Murty, and Perin [1981], based on the blossom algorithm for the 1-M/EC problem of Section 10.3. That algorithm finds $x(\lambda)$ for some specific value of λ first, and then varies λ, treating it as a parameter, through efficient parametric procedures. This produces minimum cost 1-M/ECs of all possible cardinalities, by the above result.

10.5 Degree Constrained Subnetworks, b-Matching Problems, and the General Matching Problem

Let G $= (\mathcal{N}, \mathcal{A}, c)$ be an undirected multinetwork (i.e., there may be parallel edges in $\mathcal{A}$). Let $b = (b_i)$ be a vector of specified positive integers defined on $\mathcal{N}$. The problem of finding a minimum cost subnetwork of G, subject to the constraints that for each $i \in \mathcal{N}$ the degree of i in the subnetwork should be equal to (or $\leqq$) b_i is known as a **minimum cost degree constrained subnetwork problem,** *or a* **minimum cost b-matching problem**. Verify that if $b_i = 1$ for all i, this problem becomes a minimum cost 1-matching problem. So, the b-matching problem is a generalization of the 1-matching problems discussed so far. It can be shown that every b-matching problem can be transformed into a 1-matching problem on an enlarged network, but this transformation leads to an inefficient algorithm for it. The 1-matching blossom algorithms have been generalized into blossom algorithms operating on the original network G itself for the b-matching problems, see Edmonds and Johnson [1970], and Edmonds and Pulleyblank [1975]. These algorithms can be used to solve integer programming problems of the following form

$$\begin{array}{lll} \text{Minimize} & \sum_{j=1}^{n} c_j x_j & \\ \text{subject to} & \sum_{j=1}^{n} a_{ij} x_j & \begin{cases} = b_i, i = 1, \ldots, m \\ \leqq b_i, i = m+1, \ldots, m+p \end{cases} \\ & \ell_j \leqq x_j & \leqq u_j, j = 1, \ldots, n \\ & x_j & \text{integer for all } j \end{array} \tag{10.73}$$

where all the data is integer, and the a_{ij} satisfy the conditions

$$\sum_{i=1}^{m+p} |a_{ij}| \leqq 2, \text{ for each } j = 1 \text{ to } n$$

An integer program of this form is called a **general matching problem**, and the b-matching blossom algorithms can be used to solve such a problem efficiently.

We have seen that the 1-matching/edge covering problems on general undirected networks are a generalization of the assignment problem on bipartite networks. In the same way, it can be verified that the b-matching problems on general undirected networks, are a generalization of the transportation problem on bipartite networks.

10.6 Exercises

10.6 Let Let $G = (\mathcal{N}, \mathcal{A}, c = (c_{ij}))$ be a connected undirected network with $c \geqq 0$ as the vector of edge lengths, and $\breve{s}, \breve{t}$ as the specified origin and destination nodes. Make 2 copies of G side by side, with the same edge lengths as above. To distinguish the 2 copies, add a prime, i', to the node numbers in the right hand side copy. On the left hand side copy delete $\breve{t}$ and all the edges incident at it. On the right hand side copy delete $\breve{s}'$ and all the edges incident at it. For each i introduce the new edge $(i; i')$, and define its length to be 0. Let $\tilde{\mathrm{G}}$ denote the resulting network.

(i) Let $\mathcal{P} = \breve{s}, (\breve{s}; i_1), i_1, (i_1; i_2), i_2, \dots, (i_{2r-2}; i_{2r-1}), i_{2r-1}, (i_{2r-1}; \breve{t}), \breve{t}$ be a simple path consisting of an even number, $2r$, edges between $\breve{s}$ and $\breve{t}$ in G. Define the set of edges, $\mathbf{M}(\mathcal{P})$, in $\tilde{\mathrm{G}}$, corresponding to the simple path $\mathcal{P}$ in G, to be, $\mathbf{M}(\mathcal{P}) = \{(\breve{s}; i_1), (i'_1; i'_2), (i_2; i_3), (i'_3; i'_4), \dots, (i_{2r-2}; i_{2r-1}), (i'_{2r-1}; \breve{t}'), \text{ and } (i; i') \text{ for each } i \in \mathcal{N} \text{ not on } \mathcal{P}\}$.

Whenever $\mathcal{P}$ is a simple path between $\breve{s}$ and $\breve{t}$ in G consisting of an even number of edges, the set $\mathbf{M}(\mathcal{P})$ constructed as above is a perfect matching in $\tilde{\mathrm{G}}$ of the same length as that of the path $\mathcal{P}$ in G. Conversely every perfect matching in $\tilde{\mathrm{G}}$ corresponds to a simple path consisting of an even number of edges in G between $\breve{s}$ and $\breve{t}$, of the same length. Using this develop an algorithm for finding a shortest path between $\breve{s}$ and $\breve{t}$ in G, subject to the constraint that the path consist of an even number of edges.

(ii) Put 2 copies of G side by side again, as discussed above. Delete nodes $\breve{s}'$, $\breve{t}'$ and all the edges incident at them from the right hand side copy. For each $i \neq \breve{s}, \breve{t}$, introduce the edge $(i; i')$ as above. Let the resulting network be called $\hat{\mathrm{G}}$. Show that a one to one correspondence, preserving lengths, can be established as above, between perfect matchings in $\hat{\mathrm{G}}$ and simple paths in G between $\breve{s}$ and $\breve{t}$ consisting of an odd number of edges. Using this develop an algorithm for finding a shortest path between $\breve{s}$ and $\breve{t}$ in G, subject to the constraint that the path consist of an odd number of edges.

10.7 Consider the network in Figure 10.30 with the wavy matching $\mathbf{M}_1$ marked in it. Is it possible to include another edge in the set $\mathbf{M}_1$ and still retain the matching property for it? From this, can you conclude that $\mathbf{M}_1$ is a maximum cardinality matching in this network? Why?

10.8 $\mathbf{M}$ is a perfect matching in the undirected network $\mathrm{G} = (\mathcal{N}, \mathcal{A}, c)$ with c as the vector of edge cost coefficients. Given an alternating path or cycle wrt $\mathbf{M}$, $\mathcal{P}$ say, define its cost to be $\sum(c_{ij} : \text{over } (i; j) \in \mathcal{P} \backslash \mathbf{M})$ $- \sum(c_{ij} : \text{over } (i; j) \in \mathcal{P} \cap \mathbf{M})$. Prove that $\mathbf{M}$ is a minimum cost perfect matching in G iff there exists no negative cost alternating cycle wrt it.

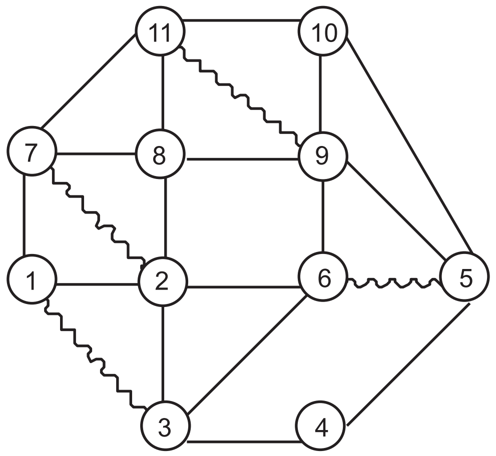

Figure 10.30: Matching edges are wavy.

10.9 $\mathbf{M}$ is a matching in the undirected network $\mathrm{G} = (\mathcal{N}, \mathcal{A}, c)$ with c as the vector of edge cost coefficients. Let $\mathbb{P}_i(\mathbf{M})$ denote the set of all augmenting paths wrt $\mathbf{M}$ beginning with an unmatched node i, and $\mathbb{P}(\mathbf{M})$ the set of all augmenting paths wrt $\mathbf{M}$. Define the cost of any path in $\mathbb{P}(\mathbf{M})$ exactly as in Exercise 10.8. Assume that there exists no negative cost alternating cycle wrt $\mathbf{M}$, and that $\mathcal{P}$ is a least cost augmenting path wrt $\mathbf{M}$ in some $\mathbb{P}_i(\mathbf{M})$ or in $\mathbb{P}(\mathbf{M})$. Let $\mathbf{M}'$ be the matching obtained by rematching $\mathbf{M}$ using $\mathcal{P}$. Prove that $\mathbf{M}'$ also satisfies the property that there exists no negative cost alternating cycle wrt it (Derigs [1981]).

10.10 A matching $\mathbf{M}$ in an undirected network $\mathrm{G} = (\mathcal{N}, \mathcal{A})$ is said to be a *maximal matching* if it is impossible to add another edge to the set $\mathbf{M}$ and still keep its matching property. Let $\underline{r}, \overline{r}$ denote the minimum and maximum cardinalities of maximal matchings in G. Prove that $\overline{r} \leqq 2\underline{r}$.

10.11 $\mathbf{M}$ is a perfect matching in the undirected network $\mathrm{G} = (\mathcal{N}, \mathcal{A}, c)$. Let $(i; j) \in \mathbf{M}$. Prove that there exists a negative cost alternating cycle $\mathbb{C}$ wrt $\mathbf{M}$ containing edge $(i; j)$ iff the cost of the shortest augmenting path wrt $\mathbf{M}' = \mathbf{M} \backslash \{(i; j)\}$ is $< c_{ij}$. Hence show that the problem of finding negative cost alternating cycles can be solved by computing shortest augmenting paths (Derigs [1986]).

10.12 A Delivery Problem A commodity has to be supplied to several users in a geographical region from a central depot by truck. Each truck has a finite capacity and hence can fulfill two users only on a trip. We are given the distance between the depot and each user, and the distance between every possible pair of users. Formulate the problem of fulfilling the user demands at minimum cost (which can be assumed to be proportional to the distance traveled by the trucks) as a matching problem (DeMaio and Roveda [1971]).

10.13 $\mathbf{E}$ is a specified subset of edges in an undirected network $\mathrm{G} = (\mathcal{N}, \mathcal{A})$. It is required to check whether there exists a perfect matching in G consisting of at most r edges from $\mathbf{E}$. Formulate this problem as a minimum cost maximum cardinality matching problem.

10.14 $\mathbf{M}_1$ is a perfect matching in an undirected network $\mathrm{G} = (\mathcal{N}, \mathcal{A})$. Prove that $\mathbf{M}_1$ is not the only perfect matching in G iff there exists an even alternating cycle wrt $\mathbf{M}$ in G.

10.15 $\mathbf{M}_1$ is a maximum cardinality matching that is not perfect in an undirected network G. Prove that $\mathbf{M}_1$ is not the only maximum cardinality matching in G iff at least one of the following things exists. (i) an even alternating cycle wrt $\mathbf{M}_1$ in G, (ii) a matching edge $(i; j)$ and an unmatched node k adjacent to i or j in G (Itai, Rodeh, and Tanimoto [1978]).

10.16 $\mathbf{M}$ is a matching in a rooted tree $\mathbb{T}$ with node 1 as the root. $\mathbf{M}$ is said to be a *proper matching* of $\mathbb{T}$ if it satisfies the following property: if a node $i \neq 1$ is unmatched, then there exists a brother j of i such that $(j; \mathrm{P}(j)) \in \mathbf{M}$ where $\mathrm{P}(j)$ is the parent node of j. Prove that every proper matching in $\mathbb{T}$ must be a maximum cardinality matching. Using this develop an $\mathrm{O}(n)$ algorithm (n is the number of nodes in $\mathbb{T}$) for finding a maximum cardinality matching in a tree (Savage [1980]).

10.17 $\mathbb{T}$ is a depth first search spanning tree in an undirected network G. $\mathbf{M}(\mathrm{G})$, $\mathbf{M}(\mathbb{T})$ are maximum cardinality matchings in G, $\mathbb{T}$ respectively. Prove $|\mathbf{M}(\mathrm{G})|/|\mathbf{M}(\mathbb{T})| \leqq 2$ (Savage [1980]).

10.18 A vertex cover or node cover in an undirected network is a subset of nodes satisfying the property that every edge in the network is incident to at least one node in the subset. The problem of finding a minimum cardinality vertex cover is NP-hard in general.

(i) In a bipartite network, prove that the cardinality of a minimum cardinality vertex cover is the same as the cardinality of a maximum cardinality matching. This result may not be true in non-bipartite networks.

(ii) $\mathbf{M}$ is a proper matching in a rooted tree $\mathbb{T}$ with node 1 as the root (see Exercise 10.16 for definition). Let $\mathbf{S}$ be the set of nonroot nodes i such that $(i; \mathrm{P}(i)) \in \mathbf{M}$, where $\mathrm{P}(i)$ is the parent of i in $\mathbb{T}$. Let $\mathbf{F} = \{\mathrm{P}(j)\colon\ j \in \mathbf{S}\}$. Show that any node not in $\mathbf{S}, \mathbf{F}$ is unmatched, and that $\mathbf{F}$ is a minimum cardinality vertex cover for $\mathbb{T}$.

(iii) Let $\mathbb{T}$ be a depth first search spanning tree in an undirected network G (the following results may not be true for arbitrary spanning trees in G). Let $\mathbf{L}(\mathbb{T}), \mathbf{NL}(\mathbb{T})$ be the sets of leaf, non-leaf nodes in $\mathbb{T}$ respectively. Let $\mathbf{M}(\mathbb{T})$ be a proper matching in $\mathbb{T}$ as defined in Exercise 10.16, and $\mathbf{M}(\mathrm{G})$ a maximum cardinality matching in G. Let $\mathbf{C}(\mathrm{G})$ be a minimum cardinality vertex cover in G.

Since $\mathbb{T}$ is a depth first search spanning tree, prove that $\mathbf{NL}(\mathbb{T})$ is a vertex cover for G. Prove the following string of inequalities.

$$|\mathbf{M}(\mathbb{T})| \leqq |\mathbf{M}(\mathrm{G})| \leqq |\mathbf{C}(\mathrm{G})| \leqq |\mathbf{NL}(\mathbb{T})| \leqq 2|\mathbf{M}(\mathbb{T})|$$

From this we get $|\mathbf{M}(\mathrm{G})|/|\mathbf{M}(\mathbb{T})| \leqq 2$ (result in Exercise 10.17) and $|\mathbf{NL}(\mathbb{T})|/|\mathbf{C}(\mathrm{G})| \leqq 2$. So, $\mathbf{NL}(\mathbb{T})$ forms a vertex cover of G guaranteed to be within a factor of 2 of the minimum vertex cover in size. Show that this bound is tight by considering the case in which G is a path of odd length and the depth first search is done from a vertex of degree 1.

(iv) Show that $|\mathbf{C}(\mathrm{G})|/\mathbf{M}(\mathrm{G})| \leqq 2$ for any undirected network G.

(v) Let $\mathbf{M}_1$ be any maximal matching in G (i.e., a matching satisfying the property that any pair of unmatched nodes are nonadjacent in G), and let $\mathbf{V}(\mathbf{M}_1)$ be the set of nodes on edges in $\mathbf{M}_1$. Then $\mathbf{V}(\mathbf{M}_1)$ is a vertex cover for G. Prove that $|\mathbf{C}(\mathrm{G})|/|\mathbf{V}(\mathbf{M}_1)| \leqq 2$.

(Savage [1982]).

10.19 Degree Constrained Subnetwork Problem in Undirected Networks $\mathrm{G} = (\mathcal{N}, \mathcal{A})$ is an undirected network with $\mathcal{N} = \{1, \ldots, n\}$. $b_1, \ldots, b_n$ are nonnegative integers. We are required to find a subnetwork $\mathrm{G}' = (\mathcal{N}, \mathcal{A}')$ of G such that $|\mathcal{A}'|$ is maximized subject to the constraint that for each $i = 1$ to n, the degree of i in G' is $\leqq b_i$. This is the undirected version of the same problem for directed networks discussed in Exercise 1.57. As stated there, in directed networks the problem easily reduces to a max-flow problem, this is not the case in undirected networks.

Construct an undirected network $\mathrm{H} = (\mathcal{N}_H, \mathcal{A}_H)$ from G by the following procedure: for each $i \in \mathcal{N}$ put b_i copies of node i, $i_1, \ldots, i_{b_i}$ say, in $\mathcal{N}_H$. On each edge $e \in \mathcal{A}$ introduce two new nodes u_e, v_e, and transform the edge e into the subnetwork $\mathrm{H}(e)$ as shown in the following Figure 10.31.

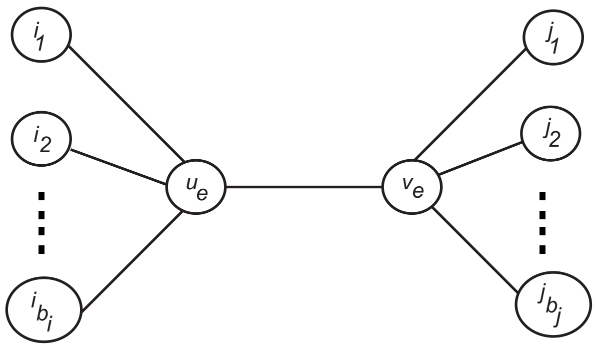

Figure 10.31:

Let $\overline{\mathbf{M}}$ be a maximum cardinality matching in H and let $\overline{\mathrm{G}} = (\mathcal{N}, \overline{\mathcal{A}}')$ be an optimum subnetwork of G to our degree constrained subnetwork problem. Then prove the following.

(a) $|\overline{\mathbf{M}}| = 2|\overline{\mathcal{A}}'| + |\mathcal{A}\backslash\overline{\mathcal{A}}'| = |\mathcal{A}| + |\overline{\mathcal{A}}'|$.

(b) Given a maximum cardinality matching $\overline{\mathbf{M}}$ in H, the set of edges $\overline{\mathcal{A}}'$ for an optimum subnetwork of G to our problem is obtained by the rule: $e \in \mathcal{A}$ is in $\overline{\mathcal{A}}'$ iff $|\mathrm{H}(e) \cap \overline{\mathbf{M}}| = 2$, and this satisfies $|\overline{\mathcal{A}}'| + |\mathcal{A}| = |\overline{\mathbf{M}}|$.

(c) Conversely, given an optimum subnetwork $\overline{\mathrm{G}} = (\mathcal{N}, \overline{\mathcal{A}}')$ for our problem, define a matching $\overline{\mathbf{M}}$ in H by the rules : (i) for each $e \in \mathcal{A}$, $(u_e; v_e) \in \overline{\mathbf{M}}$ iff $e \notin \overline{\mathcal{A}}'$, (ii) if $e = (i;j) \in \overline{\mathcal{A}}'$, then $(i_r; u_e), (v_e; j_s)$ are in $\overline{\mathbf{M}}$ for some $1 \leqq r \leqq b_i, 1 \leqq s \leqq b_j$. Since the degree of node i in $\overline{\mathrm{G}}'$ is $\leqq b_i$, this matching can be accomplished without any conflicts. Then prove that this matching will satisfy $|\overline{\mathbf{M}}| = 2|\overline{\mathcal{A}}'| + |\mathcal{A}\backslash\overline{\mathcal{A}}'|$.

(Shiloach [1981]).

10.20 Weighted Degree Constrained Subnetwork Problem in Undirected Networks Consider the degree constrained subnetwork problem discussed in Exercise 10.19, with the exception that the objective function is to maximize $\sum(w(e)$: over $e \in \mathcal{A}')$ rather than $|\mathcal{A}'|$, where $w(e)$ are specified weights for edges in G. Construct the undirected network H from G as in Exercise 10.19, and define edge weights in H by the rule: every edge in H(e) gets the same weight in H, as that of e in G. Given a feasible subnetwork $\mathrm{G}' = (\mathcal{N}, \mathcal{A}')$ of G, define a matching $\mathbf{M}$ in H as in (c) of Exercise 10.19. Show that $w(\mathbf{M}) = 2w(\mathcal{A}') + w(\mathcal{A}\backslash\mathcal{A}') = w(\mathcal{A}) + w(\mathcal{A}')$. So, maximizing $w(\mathcal{A}')$ is equivalent to maximizing $w(\mathbf{M})$. Using this, show that this problem in G gets transformed into a maximum weight matching problem in H (Shiloach [1981]).

10.21 General Degree Constrained Subnetwork Problems in Undirected Networks Let $\mathrm{G} = (\mathcal{N}, \mathcal{A})$ be an undirected network with $\mathcal{N} = \{1, \ldots, n\}$. We are given nonnegative integers $a_1, \ldots, a_n$; $b_1, \ldots, b_n$ satisfying $a_i \leqq b_i$ for all i. $w(e)$ is the weight of $e \in \mathcal{A}$. It is required to find a subnetwork $\mathrm{G}' = (\mathcal{N}, \mathcal{A}')$ of G, such that $d_{\mathrm{G}'}(i)$ = degree of node i in G$'$ is between a_i and b_i for all i; and either maximize $|\mathcal{A}'|$ in the cardinality problem, or $\sum(w(e)$: over $e \in \mathcal{A}')$ in the weighted problem. In Exercises 10.19, 10.20, we considered the cases in which all the a_i are 0.

(i) Consider the cardinality problem first. Replace all the a_i by 0 and solve the resulting problem which is now in the same form as that in Exercise 10.19, and let G^r be the resulting optimal subnetwork. If G^r is not feasible to our problem, there must be some nodes i whose degree in G^r is $< a_i$, call them deficient nodes. If our problem is feasible, show that we can use alternating paths (not necessarily simple) to transform G^r into an optimum solution of our problem, reducing the total deficiency by one in each step while preserving the cardinality.

(ii) Consider the weighted problem. Construct the network H as in Exercise 10.19 (this does not use the a_is), and for each $e \in \mathcal{A}$ make the weight of each edge in H(e) in H the same as $w(e)$. Define $\mathcal{N}_1 = \cup_{i=1}^n \{i_1, \ldots, i_{a_i}\}$. Show that our problem has a solution iff there exists a matching in H in which all the nodes in $\mathcal{N}_1$ are matched. Moreover, show that an optimum solution of our problem can be obtained from an optimum solution of the constrained weighted matching problem in H in which it is required to find a maximum weight matching subject to the constraint that every node in $\mathcal{N}_1$ must be matched, using the transformation in (b) of Exercise 10.19.

Change the data in H as follows: add α (2α) to the weight of each edge incident to exactly one node (two nodes) in $\mathcal{N}_1$, where α is a large positive number. Denote the network with the resulting data as H$'$. Find a maximum weight matching $\mathbf{M}'$ in H$'$. Show that if some nodes in $\mathcal{N}_1$ are unmathed in $\mathbf{M}'$, then the constrained matching problem in H discussed above has no solution. And if all the nodes in $\mathcal{N}_1$ are matched in $\mathbf{M}'$, then $\mathbf{M}'$ solves the constrained weighted matching problem in H. Using these, discuss an algorithm for solving our weighted subnetwork problem with any maximum weighted 1-matching algorithm (Shiloach [1981]).

10.22 It is required to find a minimum cost matching in an undirected network G $= (\mathcal{N}, \mathcal{A}, c)$ subject to the constraint that all the nodes in a specified subset $\mathcal{N}_1$ must be matched. Discuss a way of transforming this constrained matching problem into an unconstrained minimum cost matching problem.

10.23 The Edge Partitioning Problem For any undirected network L, let $\Delta(\mathrm{L})$ denote the maximum degree among its nodes. Let G $= (\mathcal{N}, \mathcal{A})$ be an undirected network. Let α, β be positive integers such that $\alpha + \beta = \Delta(\mathrm{G})$. It is required to partition $\mathcal{A}$ into disjoint subsets $\mathcal{A}_1, \mathcal{A}_2$ such that if $\mathrm{G}_t = (\mathcal{N}, \mathcal{A}_t)$, $t = 1, 2$, then $\Delta(\mathrm{G}_1) = \alpha, \Delta(\mathrm{G}_2) = \beta$. Formulate this as a special case of the degree constrained subnetwork problem of Exercise 10. 21 (Shiloach [1981]).

10.24 In a connected bipartite network, prove that there exists a matching **M**, and a node cover **N** such that every edge in **M** contains exactly one node in **N**; and every node in **N** is contained on exactly one edge in **M**.

10.25 $\mathbf{N}_t$, $t = 1, 2$ are two subsets of nodes in an undirected network G, satisfying $|\mathbf{N}_1| < |\mathbf{N}_2|$. $\mathbf{M}_t$ is a matching in G covering all the nodes in $\mathbf{N}_t$, $t = 1, 2$. Prove that there must exist a matching in G, that covers all the nodes in $\mathbf{N}_1$ and at least one node in $\mathbf{N}_2 \backslash \mathbf{N}_1$.

10.26 **N** is a specified subset of nodes in an undirected network G $= (\mathcal{N}, \mathcal{A}, c)$ with edge cost vector $c \geqq 0$. It is required to find a subset of edges **A** in G satisfying the constraint that in the subnetwork ($\mathcal{N}$, **A**) all nodes in **N** have odd degree and all nodes not in **N** have even degree. Formulate the problem of finding a minimum cost subset of edges subject to this constraint, as a matching problem.

Comment 10.5 Matchings have been studied by graph theorists since 1891. At that time it was known that the famous four color conjecture is true if every cubic (i.e., one in which every node has degree 3) planar graph can be factorized into three perfect matchings, this heightened interest in the study of matchings.

Then in the early 1900s matchings in bipartite graphs were studied extensively. Necessary and sufficient conditions for a bipartite graph to have a perfect matching have been derived. These can be explained easily using a marriage interpretation as follows. Think of the two sets of nodes of the bipartite graph as representing men, women respectively. Interpret each edge in the graph as representing a man, woman pair acquainted with each other. A matching in the graph corresponds to a set of man-woman couples (one couple determined by each matching edge) so that each couple is acquainted. To have a perfect matching, clearly the number of men and the number of women should be equal. In this context, the conditions state that if we have n men and n women, a set of n acquainted man-woman couples can be formed iff for each $1 \leqq r \leqq n$, each set of r men collectively are acquainted with at least r women. This result was the forerunner of another fundamental result in bipartite matchings, **Hall's theorem on distinct representatives**. It states that if $\mathbf{S}_t, t = 1$ to n are finite sets, there is a set of distinct elements, $x_1, \ldots, x_n$, such that $x_t \in \mathbf{S}_t, t = 1$ to n, iff for each $1 \leqq r \leqq n$, the union of any r of the $\mathbf{S}_t$s contains at least r elements. Also around this time, all the theory necessary for finding a maximum cardinality matching in a bipartite graph had been worked out by König and Egerváry.

Then in 1947, a big step in the study of matchings was taken by Tutte. He gave a characterization of general (i.e., nonbipartite) graphs that have a perfect matching, this can be viewed as a generalization of the corresponding result in bipartite graphs. Tutte's characterization states that a connected graph has no perfect matching iff there exists a set $\mathbf{S}$ of nodes in it, the deletion of which leaves more than $|\mathbf{S}|$ components having an odd number of points each. Tutte's characterization is not algorithmic, it prompted attempts to find efficient algorithms for perfect (or maximum cardinality) matchings in nonbipartite networks. Berge [1957] and Norman and Rabin [1959] have shown that the concepts of alternating and augmenting paths do generalize from the bipartite to the nonbipartite networks, and in fact the augmenting path theorem holds in nonbipartite networks without any change, i.e., that a matching in a nonbipartite network is of maximum cardinality iff it admits no augmenting path. By this time matching problems in bipartite networks became efficiently solvable through the Hungarian method. People believed for a long time that the result of Berge, Norman and Rabin leads to an efficient algorithm for the maximum cardinality matching problem in nonbipartite networks. One begins with the empty matching and repeatedly performs augmentations until there are no augmenting paths. Matters stood there until 1965 when Edmonds showed that the search for augmenting paths in nonbipartite networks is very intricate, and could take exponential time unless special techniques are developed for it. We begin at an unmatched node, and advance along an AP. If this reaches another unmatched node, we have found an augmenting path. This procedure always works nicely in bipartite networks, but in nonbipartite networks, odd cycles create subtle difficulties for it. A node can appear on an AP in either parity (outer or inner); and if we allow two visits to a node, one in each parity, the path is no longer simple; if we don't, we miss the augmenting path.

The algorithmic study of matchings took a giant step with the work of Edmonds [1965a, b]. There, he developed the elegant solution to the problem of finding augmenting paths efficiently by detecting and shrinking blossoms as they appear. In these papers he introduced the characterization of the convex hull of matching incidence vectors through the system of blossom inequalities. Using this characterization, he extended the Hungarian method for the bipartite minimum cost matching problem into the blossom algorithm for minimum cost matching problem in nonbipartite networks, and showed that it is polynomially bounded. In these papers, Edmonds has also proposed **polynomial time** property as the prime characteristic for designating algorithms as **good algorithms**. This has become a fundamental tenet of theoretical computer science ever since.

Another major outgrowth from these papers is the study of other combinatorially defined polyhedra with the aim of characterizing them through a system of linear constraints, this area is now called **polyhedral combinatorics**.

Earlier in 1962, the Chinese mathematician Kwan Mei-Ko posed the important postman's route problem. Hence Edmonds called this the **Chinese postman problem** and developed the efficient procedure for solving it using the blossom algorithm for the minimum cost perfect matching problem on the complete network defined on the set of odd degree nodes in the original network, as described in Section 1.3.8.

With the Hungarian method in bipartite networks, and the blossom algorithm in nonbipartite networks, the study of optimization problems involving matchings has taken off with a vigorous start. Several new algorithms for matching problems, as well as efficient implementations based on appropriate data structures, have been developed. Many applications in a variety of areas have come up. Some recent applications in VLSI chip design etc. lead to matching problems on such large networks, that it is impractical to solve them with the $O(n^3)$ blossom or other exact algorithms. Hence, even though mathematically efficient exact algorithms exist for them, heuristic algorithms of low order complexity are being developed for tackling very large scale matching problems.

For other algorithms for matching problems see Avis [1983], Ball and Derigs [1983], Cunningham and Marsh [1978], Derigs [1981, 1985, 1986, 1988 of Chapter 1, 1991], Even and Kariv [1975], Gabow [1976], Kameda and Munro [1974], Murty and Perin [1982], and Perin [1980].

10.7 References

Y. A. ALYAHYA, 1984, "Matching and Covering Algorithms," Ph. D. dissertation, Dept. of Industrial and Operations Engineering, University of Michigan, Ann Arbor, Mich., USA,

D. AVIS, 1983, "A Survey of Heuristics for the Weighted Matching Problem," *Networks*, 13(475-493).

M. O. BALL and U. DERIGS, 1983, "An Analysis of Alternate Strategies for Implementing Matching Algorithms," *Networks*, 13(517-549).

C. BERGE, 1957, "Two Theorems in Graph Theory," *Proceedings of the National Academy of Sciences, USA*, 43(842-844).

J. BROWN, 1977, "Shortest Alternating Path Algorithms," *Networks*, 4(311-334).

P. CARSTENSEN, K. G. MURTY, and C. PERIN, 1981, "Parametric Specified Cardinality 1-Matching/Edge Covering Problems and Intermediate Feasibility Property," Technical Report 81-6, Dept. of Industrial and Operations Engineering, University of Michigan, Ann Arbor, Mich., USA,

N. CHRISTOFIDES, 1976, "Worst-case Analysis of a New Heuristic for the Traveling Salesman Problem," Technical Report, GSIA, Carnegie-Mellon University, Pittsburgh, PA, USA.

W. H. CUNNINGHAM and A. B. MARSH III, 1978, "A Primal Algorithm for Optimum Matching," *MPS*, 8(50-72).

A. O. DEMAIO and C. A. ROVEDA, 1971, "The Minimal Cost Maximum Matching of a Graph," *Unternehmens-Forschung Operations Research*, 15, no. 3(196-210).

U. DERIGS, 1981, "A Shortest Augmenting Path Method for Solving Minimal Perfect Matching Problems," *Networks*, 11(379-390).

U. DERIGS, Nov. 1985, "An Efficient Dijkstra-like Labeling Method for Computing Shortest Odd/Even Paths," *IPL*, 21, no. 5(253-258).

U. DERIGS, 1986, "Solving Large-Scale Matching Problems Efficiently: A New Primal Matching Approach," *Networks*, 16(1-16).

U. DERIGS and A. METZ, Mar. 1991, "Solving (Large Scale) Matching Problems Combinatorially," *MP*, Series A, 50, no. 1(113-121).

J. EDMONDS, 1962, "Covers and Packings in a Family of Sets," *BAMS*, 68(494-499).

J. EDMONDS, 1965a, "Paths, Trees, and Flowers," *Canadian Journal of Mathematics*, 17(449-467).

J. EDMONDS, 1965b, "Maximum Matching and a Polyhedron With 0, 1 Vertices," *Journal Research NBS*, 69(B)(130-165).

J. EDMONDS, 1965c, "The Chinese Postman Problem," *OR*, 13, Suppl. 1(373).

J. EDMONDS and E. L. JOHNSON, 1970, "Matching: A Well Solved Class of Integer Linear Programs," PP 89-92 in R. Guy(ed.) *Combinatorial Structures and Their Applications* , Gordon and Breach, N.Y.

J. EDMONDS and W. PULLEYBLANK, 1975, "The Matching Problem and the Blossom Algorithm," Lecture notes, John Hopkins University, Baltimore, MD, USA.

S. EVEN and O. KARIV, 1975, "An O($n^{\mathbf{2}.5}$) algorithm for the Maximum Matching in General Graphs," *Proceedings of the 16th Annual Symposium on Foundations of Computer Science*, IEEE, N.Y. (100-112).

M. FUJII, T. KASAMI, and K. NINOMIYA, 1969, "Optimal Sequencing of Two Equivalent Processors," *SIAM Journal on Applied Mathematics*, 17(784-789), Erratum ibid 20, 1971(141).

H. N. GABOW, 1976, "An Efficient Implementation of Edmond's Algorithm for Maximum Matching on Graphs," *JACM*, 23(221-234).

A. ITAI, M. RODEH, and J. L. TANIMOTO, Oct. 1978, "Some Matching Problems for Bipartite Graphs," *JACM*, 25, no. 4(517-525).

T. KAMEDA and I. MUNRO, 1974, "An O($|V|\ |E|$) Algorithm for Maximum Matching of Graphs," *Computing*, 12(91-98).

A. S. LAPAUGH and C. H. PAPADIMITRIOU, 1984, "The Even-Path Problem for Graphs and Digraphs," *Networks*, 14, no. 4(507-513).

KWAN MEI-KO, 1962, "Graphic Programming Using Odd or Even Points," *Chinese Math.* , 1(273-277).

B. MONTREUIL, H. D. RATLIFF, and M. GOETSCHALCKX, Sept. 1987, "Matching Based Interactive Facility Layout," *IIE Transactions*, 19, no. 3(271-279).

K. G. MURTY and C. PERIN, 1982, "A Blossom Type Algorithm for the Minimum Cost Edge Covering Problem," *Networks*, 12(379-391).

R. Z. NORMAN and M. O. RABIN, 1959, "An Algorithm for a Minimum Cover of a Graph," *Proceedings of the American Math. Soc.*, 10(315-319).

M. W. PADBERG and M. R. RAO, 1982, "Odd Minimum Cut-Sets and b-Matchings," *MOR*, 7(67-80).

C. PERIN, 1980, "Matching and Edge Covering Algorithms," Ph. D. dissertation, Dept. of Industrial and Operations Engineering, University of Michigan, Ann Arbor, Mich., USA,

W. R. PULLEYBLANK, 1983, "Polyhedral Combinatorics," PP 312-345 in A. Bachem, M. Grotschel, and B. Korte (eds.), *Mathematical Programming, The State of the Art: Bonn 1982*, Springer-Verlag, Berlin.

C. SAVAGE, 1980, "Maximum Matchings and Trees," *IPL*, 10, no. 4/5(202-205).

C. SAVAGE, July 1982, "Depth First Search and the Vertex Cover Problem," *IPL*, 14, no. 5(233-235).

A. SCHRIJVER, 1983, "Short Proofs of the Matching Polyhedron," *Journal of Combinatorial Theory, (B),* 34(104-108).

Y. SHILOACH, April 1981, "Another Look at the Degree Constrained Subgraph Problem," *IPL*, 12, no. 2(89-92).

W. T. TUTTE, 1947, "The Factorization of Linear Graphs," *Journal of the London Math. Soc.* , 22(107-111).

L. J. WHITE, 1971, "Minimum Covers of Fixed Cardinality in Weighted Graphs," *SIAM Journal of Applied Math.*, 21(104-113).

L. J. WHITE and M. L. GILLENSON, 1975, "An Efficient Algorithm for Minimum k-Covers in Weighted Graphs," *MP*, 8(20-42).

Index

For each index entry we provide the page number where it is defined or discussed first.

AP 681
Alternating tree 682
Alternating forest 715
Augmenting tree 685

Blossom algorithm 670, 714
Blossom constraints 679, 718
  Covering 760
  Matching 718
Blossoms 687
  BCI node of 784
  Revision of 722
  Rooted simple 689
  Shrinking of 690
  Simple 687

Contraction 691
Current 692
  Blossoms 699
  Edges 699
  Equality subnetwork 720, 763, 783
  Matching 692
  Network 692
  Nodes 699

Degree constrained subnetwork 808
Dominating edge set 670, 671

Edges 670
  Equality 763
  Matching 670
  Nonmatching 670
  Solution 762
1-Edge cover 670, 757
Edge independence no. 669
Edge vector 673
  Edge covering 673, 778
  1-M/EC 673
  Matching 673
  Perfect matching 673

1-Factor 669

Hungarian 708, 730, 749, 767
  Forest conds. 730, 749, 768, 786
  Trees 708

Independent edge set 669

1-M/EC 672, 779
1-Matching 669

Perfect 669
Mate 670

Nested class 703
Nodes 670
Apex 764
Base 688, 700, 764
Exposed 747
Inner 685
Matched 670
Nonexposed 747
Outer 685
Unmatched 670
Node cover 671

Odd alternating cycle 688

Paths 681
Alternating 681
Augmenting 681
Planted forest 713
Pseudonode 691
Dormant 701
Expanding of 695
Level of 702
Prices 719
Rooted 692
Unshrinking of 728

Reducing paths 759
Rematching 682

Star 757
Symmetric assignments 680