The construction of an augmenting path in the original graph G, starting from the augmenting path in G ctr B_1 ctr B_2 ctr B_3, proceeds as follows. First, backtracing in the final graph yields the sequence of nodes B_3, 10. It is then necessary to find an alternating path through B_3 in the graph G ctr B_1 ctr B_2. The appropriate alternating path passes through nodes 1, 3, 5, B_2, 2. The desired path through B_2 in G ctr B_1 is B_1, 8, 9, and the path through B_1 in G is 7, 6, 4. Putting all these pieces together, we obtain the desired sequence 1, 3, 5, 9, 8, 7, 6, 4, 2, 10.

It is seen that there are two principal elaborations required of the bipartite cardinality matching algorithm. First, it is necessary to detect and shrink blossoms. Second, it must be possible to discover appropriate alternating paths through shrunken blossoms, so that an augmenting path in the original graph can be reconstructed. The detection of blossoms is simple, and shrinking is really no problem. (The reader should be able to think of mbre than one way to write a subroutine for graphical contraction.) However, it is a nontrivial matter to perform these operations in the most efficient manner.

In the next section, we go into some details of implementation of the algorithm, and we carry out an analysis of its complexity. We show that the algorithm can be programmed in such a way that its complexity is $O(n^3)$, as in the case of bipartite matching.

6

Cardinality Matching Algorithm

We now concern ourselves with the implementation of Edmonds' algorithm for the computer. We shall develop a labeling procedure that does not require the actual contraction of blossoms in the graph; instead blossoms are treated "as though" they were shrunk. The labeling technique provides a systematic and efficient method for backtracing through blossorns.

RECORDING OF BLOSSOMS

We need to keep a record only of outermost blossoms, and these blossoms are identified by their base nodes. Associated with each node i is am index b(i) indicating the base node of the (outermost) blossom in which it is contained. If a node i is not contained in a blossom, then $b(i) = i$. Thus two nodes i, j are in the same outermost blossom if and only if $b(i) = b(j)$.

When a new blossom is formed, the base node b of the new blossom is identified, and $b(i)$ is set to b, for all nodes i in the blossom. This means that it is necessary to maintain a listing of all the nodes within a given blossom, and it must be possible to merge these listings efficiently.

DETECTION OF AUGMENTING PATHS AND BLOSSOMS

It is possible to grow one alternating tree at a time, and when one tree becomes Hungarian to begin another at a new root node. Or, we may begin by rooting an alternating tree at each exposed node and grow all alternating trees simultaneously. There are technical reasons, concerning the modification of dual variables, why the latter alternative is preferable for the weighted matching problem. Hence we adopt this plan here.

Initially the label "$S:\varnothing$" is given to all exposed nodes. Thereafter, S-labels and T-labels are applied to nodes. An S-label indicates the existence of an even-length alternating path to the root node, and a T-label indicates the existence of an odd-length path. (A node receives both types of labels if and only if it is a nonbase node of an outermost blossom.) Augmenting paths extend between the root nodes of two different trees, as suggested in Figure 6.10.

Now, suppose the labeling procedure discovers an arc $(i, j) \notin X$ where i and j have S-labels or an arc (i, j) $\in$ X, where i and j have T-labels. Assume h(i) $\neq b(j)$, i.e., nodes i and j are not contained within the same blossom. Then an augmenting path has been found if i and j a.re in different alternating trees, and a new outermost blossom has been formed if i and j are in the same tree. The question of which one of these situations exists is resolved by backtracing from the labels on i and j. If different root nodes are reached, then an augmenting path has been found. If the same root node is reached in both cases, then a blossom has been formed.

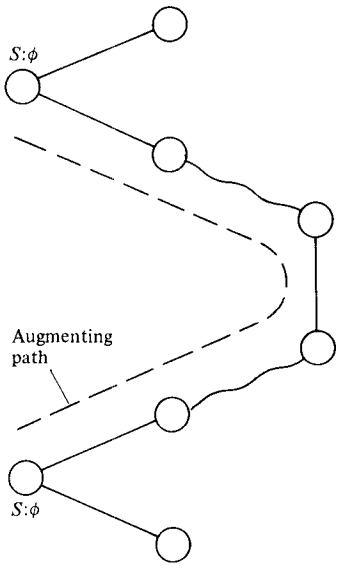

Figure 6.10 Example of augmentmg path

LABELING PROCEDURE

Rules covering the detection of augmenting paths and blossoms, in accordance with the preceding paragraph, are incorporated into the labeling procedure. Other than these, the rules for labeling are quite similar to those for bipartite matching.

That is, when an S-label on node i is scanned, the following procedure is carried out for each arc $(i, j) \notin X$ incident to i. If $b(i) = b(j)$, then nothing is done, because i and j are contained within the same blossom. (All possible labels are applied within a blossom at the time the blossom is formed; see below.) Otherwise, if node j has an S-label, backtracing is carried out from i and from j to detect either an augmenting, path or a blossom. If node j has neither an S-label nor a T-label., then the label "$T: i$" is applied to j.

When a T-label on node i is scanned the unique arc $(i, j) \in X$ incident to i is found. If $b(i) = b(j)$, then nothing is done. Otherwise, if node j has a T-label, backtracing is carried out from i and from j to detect either an augmenting path or a blossom. If node j has neither an S-label nor a T-label, then the label "S: i" is applied to j.

CONSTRUCTION OF BLOSSOMS

Once a new blossom has been detected, it is necessary to determine its membership and the identity of its base node. This is done as follows.

Backtracing from nodes i and j produces two sequences of nodes

$$i_1, i_2, \ldots i_p,$$

$$j_1, j_2 \ldots, j_q,$$

where $i_1 = j_1$ (the root node of the alternating tree) and $i_p = i$, $j_q = j$ (where backtracmg began). Since $i_1 = j_1$ and $i_p \neq j_q$, there is some index m, such that $i_1 = j_1, i_2 = j_2, \ldots, i_m = j_m$, and either $i_m = i$ or $j_m = j$, or $i_{m+1} \neq j_{m+1}$. The base of the new blossom is $i_l, 1 \leq m$, where $i_l = b(i_m)$, and its stern passes through the nodes $i_1, i_2, \ldots, i_l$.

The new blossom contains all nodes k, such that

$$b(k) \in \{b(i_m), b(i_{m+1}), \ldots, b(i_p), b(j_{m+1}), b(j_{m+2}), \ldots, b(j_q)\}. \quad (6.1)$$

Accordingly, $b(k)$ is set to i_l for all nodes k in the new blossom. This, plus the addition of missing labels to nodes within the blossom (to be described below), is all that is necessary to "shrink" the blossom.

As an example consider the situation shown in Figure 6.11. There are T-labels on nodes 8 and 9, and $(8, 9) \in X$. Hence a blossom has been

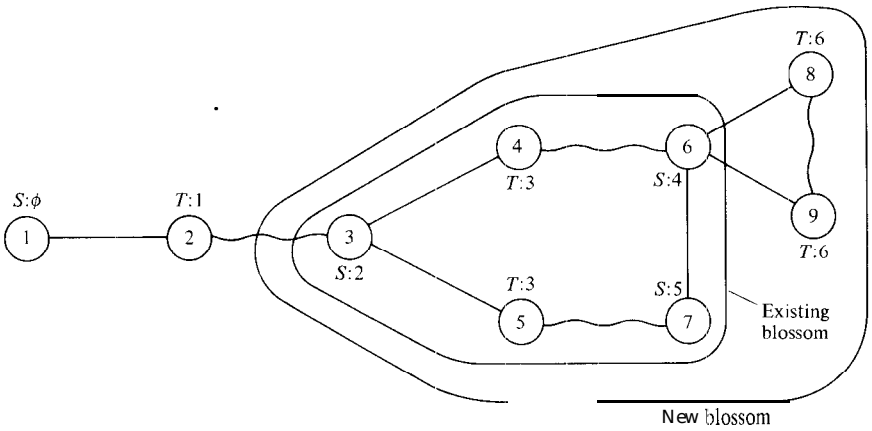

Figure 6.1 1 Example of blossom construction

detected. Backtracing from nodes 8 and 9 yields the sequences

$$1, 2, 3, 4, 6, 8$$

and

$$1, 2, 3, 4, 6, 9.$$

In this case, $i_m = j_m = 6$ and b(6) = 3, since node 6 is already part of a blossom, with node 3 as its base. The nodes 3, 4, 5, 6, 7, 8, and 9 are in the new blossom, node 3 is its base, and nodes 1, 2, and 3 are in its stem.

LABELING OF NODES IN BLOSSOMS

Between each nonbase node in the new blossom and the root of the alternating tree there exists both an even-length and an odd-length alternating path. This fact should be indicated by the existence of both an S-label and a T-label on each such node.

Suppose the blossom was detected by backtracing from nodes $i = i_p$ and j = j_q, where i and j have S-labels and $(i, j) \notin$ X. (We leave it to the reader to supply rules for the case i and j have T-labels and $(i, j) \in X$.) We concern ourselves only with nodes $i_{m+1}, i_{m+2}, \ldots, i_p$. (The rules for nodes $j_{m+1}, j_{m+2}, \ldots, j_q$ are, of course, similar.) The S-labels on $i_p, i_{p-2}, \ldots, i_{m+2}$, and the T-labels on $i_{p-1}, i_{p-3}, \ldots, i_{m+1}$ were actually used in backtracing. Hence any missing labels must be T-labels on $i_p, i_{p-2}, \ldots, i_{m+2}$, or S-labels on $i_{p-1}, i_{p-3}, \ldots, i_{m+1}$. The label assigned to any node i_r will be such that backtracing from that label yields the sequence of nodes $i_r, i_{r+1}, \ldots, i_p, j_q, j_{q-1}, \ldots, j_1$.

Let us assign missing labels to $i_{m+}, \ldots, i_{m+2}, \ldots, i_p$ in order. Suppose i_r lacks an S-label. We assert that necessarily $(i_r, i_{r+1}) \in$ X and that i_{r+1}

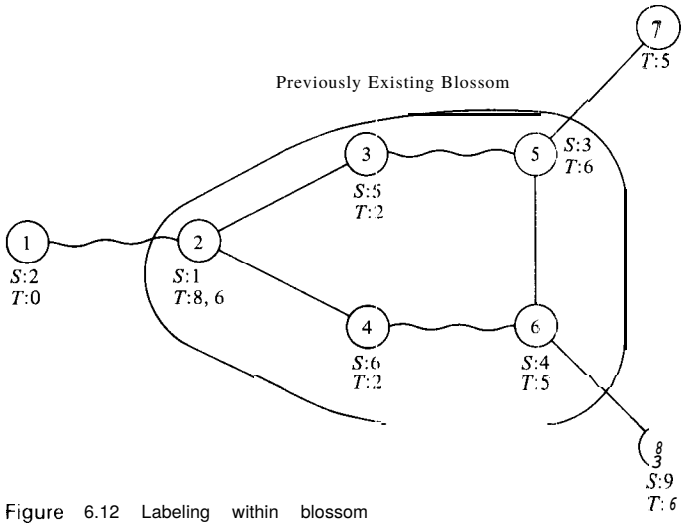

Figure 6.12 Labeling within blossom

lacks a T-label. We give i_r the label "S: i_{r+1}". The T-label we shall assign to i_{r+1} will cause backtracing to be carried out correctly.

Now suppose i_r lacks a T-label. We assert that necessarily $(i_r, i_{r-1}) \notin X$. If also $i_r + 1$ lacks an S-label, then we give the label "$T: i_{r+1}$" to i_r. The S-label we shall assign to i_{r+1} will cause backtracing to be carried out correctly.

But now suppose i_r lacks a T-label and i_{r+1} has an S-label already. Then i_r must be the base node of a previously existing blossom, containing i_{r+1}, and backtracing from the S-label on $i_r + 1$ will lead back to i_r. It is therefore quite wrong to give the label "$T: i_{r-1}$" to node i_r.

What we do to resolve this problem is to find the last node i_k in the sequence $i_{r+1}, i_{r+2}, \ldots, i_p$ that is contained in this previously existing outermost blossom with i_r as base. Necessarily $k \geq r + 2$. We then assign a special label "$T: i_k + 1, i_k$" to i_r. This label is interpreted as follows: There exists an odd-length alternating path between i_r and the root node. To find this path, backtrace from the S-label on node i_{k+1} to the root, and also from node i_k to i_r itself. The arcs thus discovered, together with the arc (i_k, i_{k+1}), properly ordered, constitute the desired alternating path.

An example of the application of labels within a blossorn is shown in Figure 6.12.

BACKTRACING ROUTINE

The introduction of special T-labels with double indexes does complicate backtracing a bit, and a recursive routine is called for. For example, in backtracing one may encounter the label "$T: i, j$" at node k. Backtracing

from j to k, one may then encounter "T: i_1, j_1" at k_1. Backtracing from j_1 to k_1, one may encounter "T: i_2, j_2" at k_2, and so on. This may continue for as many levels as blossoms are nested. Suffice it to say that the backtracing routine can be efficiently and elegantly implemented on a computer, and that backtracing from a given node to the root of the alternating tree is no more than O(n) in running time.

The complete cardinality matching algorithm can now be summarized as follows.

NONBIPARTITE CARDINALITY MATCHING ALGORITHM

Step 0 (Start) The graph G = (N, A) is given. Let X be any matching, possibly the empty matching. Set $b(i) = i$, for a.11 nodes $i \in N$. No nodes are labeled.

Step 1 (Labeling)

(1 .0) Apply the label "S: $\varnothing$" to each exposed node.

(1.1) If there are no unscanned labels, go to Step 4. Otherwise, find a node i with an unscanned label. If the label is an S-label, go to Step 1.2: if it is a T-label, go to Step 1.3.

(1.2) Scan the S-label on node i by carrying out the following procedure for each arc $(i, j) \notin X$ incident to node i. If $b(i) = b(j)$., do nothing. Otherwise, if node j has an S-label, backtrace from the S-labels on nodes i and j and if different root nodes are reached go to Step 2; if the same root node is reached, go to Step 3. If node j has neither an S-label nor a T-label, apply the label "T: i" to j.

When the scanning of node i is complete, return to Step 1.1.

(1.3) Scan the T-label on node i as follows. Find the unique arc $(i, j) \in X$ incident to node i. If $b(i) = b(j)$, do nothing. Otherwise, ifnodej has a T-label. backtrace from the T-labels on nodes i and j and if different root nodes are reached, go to Step 2; if the same root node is reached, go to Step 3. If node j has neither an S-label nor a T-label, apply the label "S: i" to j.

Return to Step 1.1.

Step 2 (*Augmentation*) An augmenting path has been found in Step 1.2 or 1.3. Augment the matching X. Remove all labels from nodes and set $b(i) = i$, for all i. Return to Step 1 .0.

Step 3 (Blossoming) A blossom has been formed in Step 1.2 or 1.3. Determine the membership and base node of the new blossom, as described in the text. Supply missing labels for all nonbase nodes in the new blossom.

Reset b(i) for all nodes i in the new blossom. Return to Step 1.2 or 1.3, as appropriate.

Step 4 (Hungarian Labeling) The labeling is Hungarian. No augmenting path exists, and the matching X is of maximum cardinality. The labels and blossom numbers can be used to construct an optimal dual solution (cf. Section 7). Halt.//

Let us consider the complexity of the algorithm. For a graph with n nodes, there can be no more than $O(n)$ augmentations and applications of the labeling procedure. Each application of the labeling procedure calls for the labels on each of the n nodes to be scanned at most once, and each scanning operation requires at most $O(n)$ steps (ignoring backtracing, and so on). Hence simple scanning and labeling contributes $O(n^3)$ steps overall to the algorithm.

There can be no more than O(n) blossoms formed per augmentation, or $O(n^2)$ overall. Each augmentation and each blossom requires, backtracing, which is O(n) in complexity. Hence backtracing contributes $O(n^3)$ steps overall. The other operations for blossom construction, including the determination of blossom membership by (6.1) and the application of missing labels, require 0 (n) steps per blossom or $O(n^3)$ steps overall. The complexity of other operations is dominated by those mentioned above. Hence we conclude that the overall running time of the algorithm is $O(n^3)$.

7
Duality Theory

We now wish to formulate and prove a duality theorem for nonbipartite matching, generalizing the König-Egervary theorem for bipartite matching. The appropriate dual structure is suggested by the notion of blossoms, and the cardinality intersection algorithm provides a constructive proof for the duality theorem.

Let G = (N, A) be a given graph and let $\mathcal{N} = \{N_1, N_2, \ldots, N_p\}$ be a family of subsets of nodes, i.e., $N_i \subseteq$ N, where each N_i contains an odd number of elements. If $|N_i| = 1$, then N_i is said to cover all arcs incident to the node in N_i, and the capacity of N_i is one. If $|N_i| = 2r + 1$, $r \geq 1$, then N_i is said to *cover* all arcs, both ends of which are incident to nodes in N_i, and the *capacity* of N_i is r. The family $\mathcal{N}$ is said to be an *odd-set cover* if each arc of the graph is covered by at least one subset $N_i \in \mathcal{N}$. The *capacity* of $\mathcal{N}$, denoted $c(\mathcal{N})$, is the sum of the capacities of the odd sets contained within it.