eorem 4.24 *The Scale-and-Shrink Algorithm solves the minimum-cost υ problem in time $O((m_0 + n)n \log nS(n,m))$, where m_0 is the number of s having finite capacity.* ∎

We remark that Orlin has discovered an improved version of the Scale- l-Shrink Algorithm, having a running time of $O(m \log nS(n,m))$ for the ımum-cost flow problem; see Orlin [1988].

ercises

1. Show that contracting an arc of a transshipment problem does not, in general, preserve the existence of an optimal solution.

2. Solve the transshipment problems of Exercise 4.17 with the Successive-Scaling Algorithm.

3. If the Successive-Scaling Algorithm were replaced by the Primal-Dual Algorithm in the statement of the Scale-and-Shrink Algorithm, what would ɔe the running time of the new algorithm?

CHAPTER 5

Optimal Matchings

5.1 MATCHINGS AND ALTERNATING PATHS

We have seen some examples of matching problems in Sections 1.1 and 3.3. In this chapter we develop the theory, algorithms, and further applications of optimal matching.

We begin by introducing and reviewing some terminology. A *matching* in a graph $G = (V, E)$ is a set M of edges such that no node of G is incident with more than one edge in M. Given a matching M, we say that M *covers* a node v (or that v is M*-covered*) if some edge of M is incident with v. Otherwise, v is M*-exposed.* Note that the number of nodes covered by M is exactly $2|M|$, and that the number of M-exposed nodes is $|V| - 2|M|$. A *maximum* matching is one of maximum cardinality, or equivalently, one that has the fewest exposed nodes. We denote by $\nu(G)$ the cardinality of a maximum matching of G, and by def(G) (the "deficiency of G") the minimum number of exposed nodes for any matching of G. So def$(G) = |V| - 2\nu(G)$. A *perfect* matching is one that covers all the nodes. See Figure 5.1.

A basic problem is to decide whether a graph has a perfect matching. A slightly more general one is to find a maximum matching. Finally, we might want a perfect matching having minimum weight with respect to some given edge-weights. The last problem includes as a special case the matching problem we introduced in Chapter 1, arising from the minimization of plotter pen motion.

A graph G is *bipartite* if its nodes can be partitioned into two sets V_1, V_2 so that every edge joins a node in V_1 to a node in V_2. We have seen that matching problems in bipartite graphs can be solved by network flow methods. In Section 3.3 we reduced the problem of finding a maximum matching in a

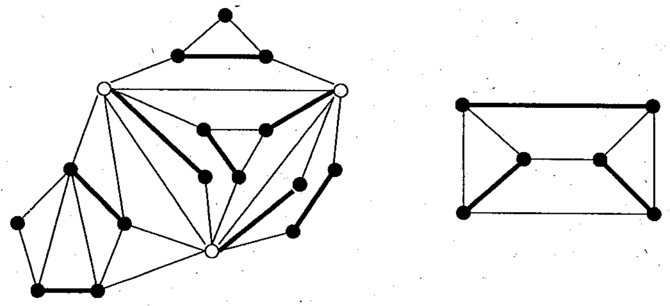
Figure 5.1. A maximum matching and a perfect matching

partite graph to an integral maximum flow problem. Similarly, the problem finding a minimum-weight perfect matching in a bipartite graph can be lved by solving a minimum-cost flow problem. (See Exercises 4.3 and 4.33.) we already know how to solve matching problems for bipartite graphs. this chapter we seldom rely on network flow methods, instead treating atching problems directly. We do frequently point out the special case when e graph is bipartite, because it offers an important and rather easy contrast the general case.

In the remainder of this section we give a few more definitions and basic sults. Given a matching M of G, a path P is *M-alternating* if its edges are ternately in and not in M. (Note that we do not specify the type of the st and last edges of P.) If, in addition, the end nodes of P are distinct and e both M-exposed, then P is an *M-augmenting* path. Figure 5.2 shows a atching M and an M-augmenting path. The reader should be able to verify e connection between this use of the term and that in Chapter 3. Namely, if is the integral flow corresponding to M in the formulation of the maximum partite matching problem as a maximum integral flow problem, then x-ıgmenting paths correspond to M-augmenting paths. This justifies the use similar terminology. It is a consequence of the Augmenting Path Theorem maximum flows that a matching M of a bipartite graph is maximum if ıd only if there is no M-augmenting path. It is a fundamental fact, due to ɛrge [1957], that this holds for all graphs.

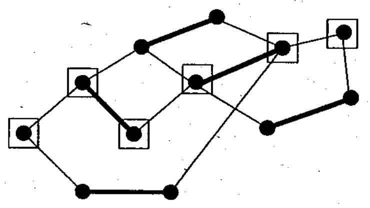
Figure 5.2. Augmenting path

Theorem 5.1 *(Augmenting Path Theorem of Matchings) A matching M in a graph $G = (V, E)$ is maximum if and only if there is no M-augmenting path.*

For sets S and T, we let $S \triangle T$ denote their *symmetric difference*, that is, $S \triangle T$ is the set of elements belonging to one, but not both, of S, T.

Proof of the Augmenting Path Theorem of Matchings: Suppose that there exists an M-augmenting path P joining nodes v and w. Then $N = M \triangle E(P)$ is a matching that covers all nodes covered by M, plus v and w. Therefore, M is not maximum.

Conversely, suppose that M is not maximum and so some other matching N satisfies $|N| > |M|$. Let $J = N \triangle M$. Each node of G is incident with at most two edges of J, so J is the edge-set of some node disjoint paths and circuits of G. For each such path or circuit, the edges alternately belong to M or N. Therefore, all circuits are even, and contain the same number of edges of M and N. Since $|N| > |M|$, there must be at least one path with more edges of N than M. This path is an M-augmenting path. (See Figure 5.3.) ∎

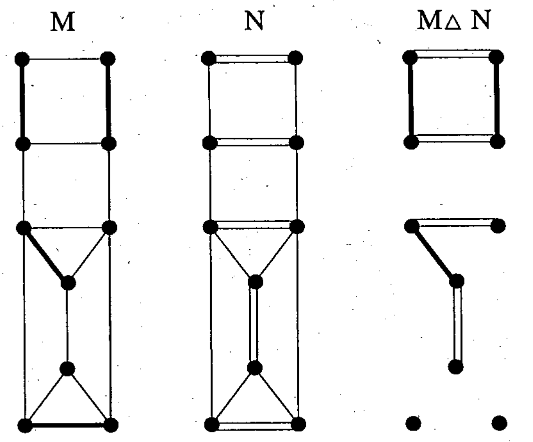

Figure 5.3. Augmenting paths and larger matchings

The Augmenting Path Theorem of Matchings suggests a possible approach to constructing a maximum matching, analogous to the augmenting path algorithm for the maximum flow problem. Namely, we repeatedly find an augmenting path and obtain a new matching using the path, until we discover a matching for which there is no augmenting path. This is the approach that we shall take. It requires having a way of recognizing when a matching is maximum, so that the search for augmenting paths can be abandoned.

Circuits and the Tutte-Berge Formula

call that a *cover* of G is a set A of nodes such that every edge has st one end in A. The cardinality of a cover provides an upper bound e size of any matching. In Section 3.3 we discussed Kőnig's Theorem, states that, if G is bipartite, there is a matching and a cover of equal iality. Therefore, in that special case, one can prove that a matching ximum by exhibiting an appropriate cover. In general, however, this is lways possible; for example, when G consists of a circuit of odd length l, the maximum size of a matching is k and the minimum size of a cover 1. Therefore, we need a stronger upper bound on the size of a maximum ing.

bound arises from the following ideas. Let A be a subset of nodes for $G\backslash A$ has k components $H_1, \ldots, H_k$ having an odd number of nodes. Figure 5.4.) Let M be a matching of G; for each i, either H_i has an

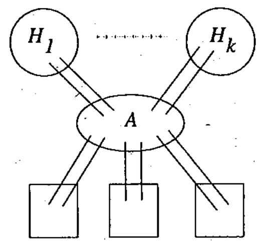

Figure 5.4. An upper bound for the size of a maximum matching

posed node, or M contains an edge having just one end in $V(H_i)$. All edges have their other ends in A, and, since M is a matching, all these must be distinct. Therefore, there can be at most $|A|$ such edges, and e number of M-exposed nodes must be at least $k - |A|$. This gives the ving general upper bound on the size of a matching of G. (We denote by) the number of odd components of a graph H.)

$$\text{For any } A \subseteq V,\ \nu(G) \leq \tfrac{1}{2}(|V| - \mathrm{oc}(G\backslash A) + |A|). \qquad (5.1)$$

choose A in (5.1) to be a cover of G, then there are $|V| - |A|$ odd ponents of $G\backslash A$ (each having a single node), so the right-hand side reduces |. In other words, the bound provided by (5.1) is at least as strong as provided by covers. In fact, it is better, for example, in the case of a it of length $2k+1$. Then we can choose $A = \emptyset$ to show that there is no hing of size bigger than k, whereas the smallest cover has size $k+1$. Less lly, one can see that choosing A to be the white nodes in the first graph gure 5.1 gives $\mathrm{oc}(G\backslash A) - |A| = 2$, showing that every matching will leave ast two nodes exposed and hence proving that the indicated matching, of size 8, is maximum. (The smallest cover has size 11.) The central result of the theory of maximum matching states that this upper bound can be achieved.

Theorem 5.2 *(Tutte-Berge Formula) For a graph $G = (V, E)$ we have*

$$\max\{|M| : M \text{ a matching}\} = \min\{\tfrac{1}{2}(|V| - \mathrm{oc}(G\backslash A) + |A|) : A \subseteq V\}.$$

It immediately implies a condition for the existence of a perfect matching.

Theorem 5.3 *(Tutte's Matching Theorem) A graph $G = (V, E)$ has a perfect matching if and only if for every subset A of nodes we have $\mathrm{oc}(G\backslash A) \leq |A|$.*

Theorem 5.3 was proved by Tutte [1947], and is quite easily seen to be equivalent to the min-max theorem, which was stated by Berge [1958]. So Theorem 5.2 is often called the *Tutte-Berge Formula.*

Just as we need a more general concept than in the bipartite case to get a min-max formula, we also need a new algorithmic idea. It is the idea of shrinking odd circuits. Let C be an odd circuit in G. We define $G' = G \times C$, the subgraph obtained from G by shrinking C, as follows. $G \times C$ has node-set $(V\backslash V(C)) \cup \{C\}$ and edge-set $E\backslash\gamma(V(C))$; for each edge e of G' and end v of e in G, v is an end of e in G' if $v \notin V(C)$, and otherwise C is an end of e in G'. (Notice that this is related to the concept of node identification used in Section 3.5, in that we could obtain essentially the same graph by a sequence of node identifications. But here we have a special convention for naming the new node of G'.) We call $G \times C$ the graph obtained from G by *shrinking* C. We illustrate in Figure 5.5 the result of shrinking an odd circuit. A fundamental observation about odd-circuit shrinking is the following: If we

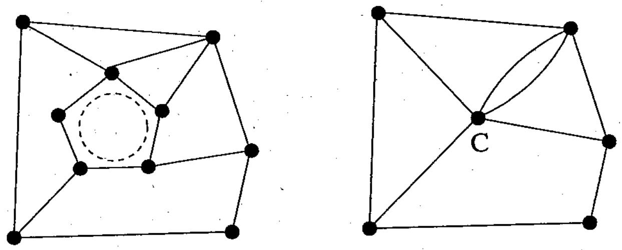

Figure 5.5. Shrinking an odd circuit

have a matching in $G \times C$, we can extend it to a matching in G in an obvious way. See the example in Figure 5.6. We state this formally, as follows.

Proposition 5.4 *Let C be an odd circuit of G, let $G' = G \times C$, and let M' be a matching of G'. Then there is a matching M of G such that $M \subseteq M' \cup E(C)$ and the number of M-exposed nodes of G is the same as the number of M'-exposed nodes of G'.*

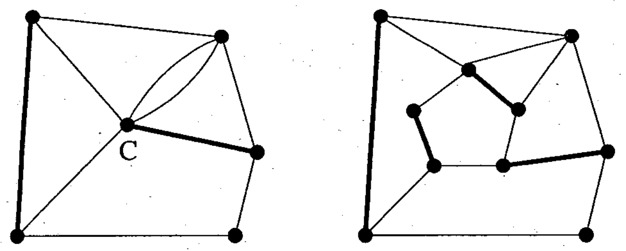

Figure 5.6. Extending a matching

Proof: Choose a node $w \in V(C)$ as follows. If C is covered by $e \in M'$, then choose w to be the node in $V(C)$ that is an end of e, and otherwise, choose w arbitrarily. Now deleting w from C results in a subgraph having a perfect matching M''. Take $M = M' \cup M''$. It is easy to check that M has the required properties. ▮

Proposition 5.4 gives the inequality

$$\nu(G) \geq \nu(G \times C) + (|V(C)| - 1)/2, \tag{5.2}$$

or equivalently, $\operatorname{def}(G) \leq \operatorname{def}(G \times C)$. It would be nice to know that this holds with equality; this is equivalent to the existence of a maximum matching using $(|V(C)| - 1)/2$ edges from $E(C)$. If this were true, finding a maximum matching in G could be reduced to finding a maximum matching in $G \times C$. Let us call an odd circuit C of G *tight* if (5.2) holds with equality. Not every odd circuit is tight; see the example of Figure 5.7. In fact, the algorithm we

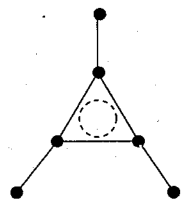

Figure 5.7. An odd circuit that is not tight

will describe will shrink odd circuits that are not necessarily tight, since it is difficult to identify tight ones. But we can use the idea of tight circuits to give a nonconstructive proof of the Tutte-Berge Formula, and this is what we do in the rest of the section. (We point out that the algorithm of the next section will give a proof of the formula that is independent of this one.) Let us call a node v of G *inessential* if there is a maximum matching of G not covering v; otherwise, we call v *essential.* Suppose that A satisfies (5.1) with equality. Choose $v \in A$ and consider $G' = G\backslash v$. Then $G'\backslash(A\backslash\{v\})$ has the

same odd components as $G\backslash A$, and so $\nu(G') < \nu(G)$. In other words

$$\text{If } A \text{ satisfies (5.1) with equality, then every } v \in A \text{ is essential.} \tag{5.3}$$

Lemma 5.5 *Let $G = (V, E)$ be a graph and let $vw \in E$. If v, w are both inessential, then there is a tight odd circuit C using vw. Moreover, C is an inessential node of $G \times C$.*

Proof: There are maximum matchings M_1 and M_2 of G not covering v and w, respectively. Note that M_1 covers w and M_2 covers v; otherwise we could add vw to one of them and get a bigger matching. Consider the subgraph $H = (V, M_1 \triangle M_2)$. The component containing v consists of a path P beginning at v, since v is M_1-exposed. If P ends at an M_1-exposed node, it is M_1-augmenting, a contradiction. If P ends at an M_2-exposed node other than w, then attaching vw to it, we get an M_2-augmenting path, a contradiction. Therefore, P ends at w and so it forms with vw an odd circuit C. There is a maximum matching (M_1, for example) that contains $(|V(C)| - 1)/2$ edges from $E(C)$, so C is tight. Finally, $M_1\backslash E(C)$ is a maximum matching of $G \times C$ not covering C, proving the last statement. ▮

Proof of the Tutte-Berge Formula (Theorem 5.2): We already observed that the left-hand side is at most the right, so we need only show that there is a matching M and a set A such that the number of M-exposed nodes is exactly $\operatorname{oc}(G\backslash A) - |A|$. The proof is by induction on the number of edges. The result is certainly true for graphs having no edges, so choose an edge vw.

Suppose that $\nu(G\backslash v) = \nu(G) - 1$. (The case for w is similar.) Then applying induction to $G\backslash v$, we have that there exists a set A' of nodes and a matching M' of $G\backslash v$ such that there are exactly $\operatorname{oc}((G\backslash v)\backslash A') - |A'|$ M'-exposed nodes. Therefore, there is a matching M of G for which the number of M-exposed nodes is exactly $\operatorname{oc}(G\backslash(A' \cup \{v\})) - |A' \cup \{v\}|$, and the result is proved for G.

Now we can suppose that both v and w are inessential. By Lemma 5.5, G has a tight circuit C. Applying induction to $G' = G \times C$, we get a matching M' and a set A' of nodes such that the number of M'-exposed nodes is exactly $\operatorname{oc}(G'\backslash A') - |A'|$. Notice that the node C of G' cannot be in A', by (5.3), since it is an inessential node of G', by Lemma 5.5. Now we know from Proposition 5.4 that we can extend M' to a matching M of G having the same number of exposed nodes. But when we delete A' from G, we get the same number of odd components, because if one contains C, that node will now be replaced by $|V(C)|$ nodes and the component of $G\backslash A'$ containing them will still be odd. Thus we get a matching M and a set A' of nodes such that the number of M-exposed nodes is exactly $\operatorname{oc}(G\backslash A') - |A'|$, as required. ▮

Exercises

5.1. Let x be the integral feasible flow corresponding to the matching M of the bipartite graph G. Show that there is a one-to-one correspondence between

x-augmenting paths and M-augmenting paths. What is the connection between x-incrementing paths and M-alternating paths?

5.2. Let M be a matching of G and let p be the cardinality of a maximum matching. Using the method of proof of the Augmenting Path Theorem of Matchings, show that there are at least $p-|M|$ node-disjoint M-augmenting paths.

5.3. Use the result of Exercise 5.2 to show that, if M is a matching of cardinality 4000 in a graph having a matching of size 5000, then there is an M-augmenting path of length at most 9.

5.4. Suppose that $p > 0$ is the cardinality of a maximum matching of G, and let M be a matching of cardinality at most $p - \sqrt{p}$. Use the result of Exercise 5.2 to show that there is an M-augmenting path having at most $\sqrt{p}$ edges from M.

5.5. Let G be a graph and k a positive integer, $k \leq |V|/2$. Construct a graph G' having the property that G' has a perfect matching if and only if G has a matching of size k. Use this construction to prove the Tutte-Berge Formula from Tutte's Matching Theorem.

5.6. Let B denote the set of inessential nodes of $G = (V, E)$, C denote the set of nodes not in B but adjacent to at least one element of B, and D denote $V \setminus (B \cup C)$. ($\{B, C, D\}$ is called the *Gallai-Edmonds Partition* of G.) Prove that

(a) C is a minimizer in the Tutte-Berge formula;

(b) For every maximum matching M and every node $v \in C$, there is an edge $vw \in M$ with $w \in B$;

(c) Every maximum matching contains a perfect matching of $G[D]$.

5.7. Use Tutte's Matching Theorem to prove Petersen's Theorem, namely, if every node of G has degree 3 and $G \setminus e$ is connected for every edge e, then G has a perfect matching. Hint: First show that if $C \subseteq V$, $|C|$ odd, then $|\delta(C)| \geq 3$.

5.8. Show that Lemma 5.5 applied to a bipartite graph is equivalent to Kőnig's Theorem.

5.9. Suppose that G is connected and has the property that every node v is inessential. Use Lemma 5.5 to show that $\nu(G) = (|V| - 1)/2$.

5.2 MAXIMUM MATCHING

In this section we describe an algorithm, originally due to Edmonds [1965], to solve the following problem.

Maximum Matching Problem
Given a graph G, find a maximum matching of G.

The main techniques of the algorithm have been introduced in Section 5.1. We actually direct most of our efforts to showing how to find a perfect matching, if possible. Later we show how this method can be used to find a maximum matching.

Alternating Trees

We define a basic structure maintained by matching algorithms. Suppose we have a matching M of G and a fixed M-exposed node r of G. We build up iteratively sets A, B of nodes, such that each node in A is the other end of an odd-length M-alternating path beginning at r, and each node in B is the other end of an even-length M-alternating path beginning at r. A motivation for wanting such a set is that, if we find an edge vw such that $v \in B$ and $w \notin A \cup B$ is M-exposed; then the M-alternating path P from r to v together with vw gives an M-augmenting path. Such sets A, B could be built up, beginning with $A = \emptyset$, $B = \{r\}$, by the following rule:

$$\text{If } vw \in E,\ v \in B,\ w \notin A \cup B,\ wz \in M, \text{ then add } w \text{ to } A,\ z \text{ to } B. \quad (5.4)$$

The set $A \cup B$ and the edges used in its construction have the structure indicated in Figure 5.8. Namely, they form a tree T with "root" r having the

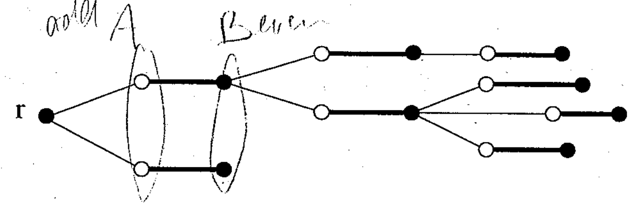

Figure 5.8. An alternating tree

$|B| = |A|+1$

following properties:

(a) Every node of T other than r is covered by an edge of $M \cap E(T)$;

(b) For every node v of T, the path in T from v to r is M-alternating.

Such a tree is called an *M-alternating tree.* Given an M-alternating tree T, we denote by $A(T)$ and $B(T)$ the sets of nodes at odd and even (respectively) distance from the root. We may refer to the elements of $A(T)$ as the *odd* nodes of T and to the elements of $B(T)$ as the *even* nodes of T. (In Figure 5.8 the odd nodes are white and the even nodes are black.) Notice that $|B(T)| = |A(T)| + 1$, since nodes other than r come in matched pairs, one in $A(T)$ and one in $B(T)$.

Let us state as subroutines written in the language of these trees, two of the techniques that we have already introduced. The first subroutine is an encoding of the rule (5.4), and the second is a statement of the augmentation step.

Use vw to extend T

Input: A matching M' of a graph G', an M'-alternating tree T, and an edge vw of G' such that $v \in B(T)$, $w \notin V(T)$ and w is M'-covered.

Action: Let wz be the edge in M' covering w. (Note that z is not a node of T.) Replace T by the tree having edge-set $E(T) \cup \{vw, wz\}$.

Use vw to augment M'

Input: A matching M' of a graph G', an M'-alternating tree T of G' with root r, and an edge vw of G' such that $v \in B(T)$, $w \notin V(T)$ and w is M'-exposed.

Action: Let P be the path obtained by attaching vw to the path from r to v in T. Replace M' by $M' \triangle E(P)$.

Here is a simple condition under which we can conclude that G has no perfect matching. We call an M-alternating tree T in a graph G *frustrated* if every edge of G having one end in $B(T)$ has the other end in $A(T)$.

Proposition 5.6 *Suppose that G has a matching M and an M-alternating tree T that is frustrated. Then G has no perfect matching.*

Proof: Clearly, every element of $B(T)$ is a single-node odd component of $G \backslash A(T)$. Since $|A(T)| < |B(T)|$, therefore G has no perfect matching. ∎

The Bipartite Case

The matching algorithm will need some other ideas, involving the handling of odd circuits. But since odd circuits do not arise when the graph is bipartite, we can now state an algorithm for perfect matching in a bipartite graph.

Perfect Matching Algorithm for Bipartite Graphs

```
Set $M = \emptyset$;
Choose an $M$-exposed node $r$ and put $T = (\{r\}, \emptyset)$;
While there exists $vw \in E$ with $v \in B(T)$, $w \notin V(T)$
    If $w$ is $M$-exposed
        Use $vw$ to augment $M$;
        If there is no $M$-exposed node in $G$
            Return the perfect matching $M$ and stop;
        Else
            Replace $T$ by $(\{r\}, \emptyset)$, where $r$ is $M$-exposed;
    Else
        Use $vw$ to extend $T$;
Stop; $G$ has no perfect matching.
```

The correctness of the algorithm is based on the following observation.

Proposition 5.7 *Suppose that G is bipartite, that M is a matching of G, and that T is an M-alternating tree such that no edge of G joins a node in $B(T)$ to a node not in $V(T)$. Then T is frustrated, and hence G has no perfect matching.*

Proof: We must show that every edge having an end in $B(T)$ has an end in $A(T)$. From the hypothesis, the only possible exception would be an edge joining two nodes in $B(T)$. But this edge, together with the paths joining them to the root of T, would form a closed path of odd length, which is clearly impossible in a bipartite graph. Hence T is frustrated, and so by Proposition 5.6, G has no perfect matching. ∎

The Blossom Algorithm for Perfect Matching

The basic ideas for a bipartite matching algorithm go back at least to Kőnig, and algorithms were well known by the decade of the 1950s. For the general problem, Tutte's Matching Theorem appeared in the 1940s, and approaches based on augmenting paths were suggested in the 1950s. However, the problem of finding an efficient algorithm remained open until the 1960s, when it was solved in the landmark paper of Edmonds [1965]. A main new idea that he used is the shrinking of odd circuits, which we have already introduced; now we will show the interesting way in which this concept and that of alternating trees come together in an algorithm.

Suppose that we have a graph G' obtained from G by a sequence of odd-circuit shrinkings. We call G' a *derived graph* of G. Nodes of G' are of two types: those that are nodes of G, which we call *original nodes* of G', and those that are not nodes of G, which we call *pseudonodes*. Given a node v of G', there corresponds a set $S(v)$ of nodes of G: If $v \in V$, that subset is just $\{v\}$, but if $v = C$ is a pseudonode, it is defined recursively to be the union of the sets $S(w)$ for all the nodes $w \in V(C)$. Clearly, the cardinality of this set will always be odd, since it is the sum of an odd number of odd numbers. Moreover, the collection of sets $S(v)$ for all nodes v of G' forms a partition of V. There is a simple extension of Proposition 5.6, which sometimes allows us to recognize, when we are working with one of its derived graphs, that G has no perfect matching.

Proposition 5.8 *Let G' be a derived graph of G, let M' be a matching of G', and let T be an M'-alternating tree of G' such that no element of $A(T)$ is a pseudonode. If T is frustrated, then G has no perfect matching.*

Proof: When we delete $A(T)$ from G, we get a component with node-set $S(v)$ for each $v \in B(T)$. Therefore, $\text{oc}(G\backslash A(T)) > |A(T)|$, so G has no perfect matching. ∎

So the algorithm works with derived graphs of G, since if we find a perfect matching in a derived graph, we know there is a perfect matching in G, and if we find a certain kind of frustrated tree in a derived graph, we can conclude that G has no perfect matching. But how do we decide which circuits to shrink? The answer can be discovered by considering what happens if we try to solve the problem with only the tree extension and augmentation steps. For example, in Figure 5.9 we see an M-alternating tree in a graph G to which neither of the steps applies, yet it is clear that there does exist a perfect matching. Of course the tree is not frustrated, because of the existence of the

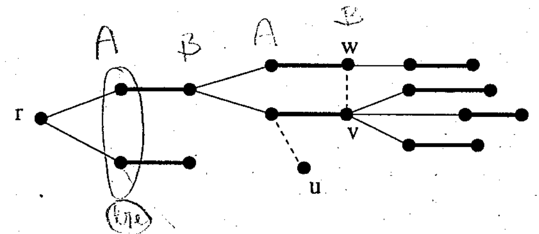

Figure 5.9. Augmenting path not found by tree extension

edge vw. In general, if there exists an edge vw with $v, w \in B(T)$, then this edge together with the path in T from v to w forms an odd circuit, which we can shrink to form a graph G'. (An odd circuit encountered in this way was called a "blossom" by Edmonds, and his algorithm is often called the "Blossom Algorithm".) Then something very nice happens: The matching and the alternating tree easily yield similar structures in G'. Consider, for example, the graph of Figure 5.10 obtained when we shrink the odd circuit in Figure 5.9. The matching edges that remain form a matching in G', and the tree edges that remain form a tree that is alternating with respect to this

matching. Notice also that in G', an augmenting path becomes detectable by the straightforward approach we used before. Here is a formal description of

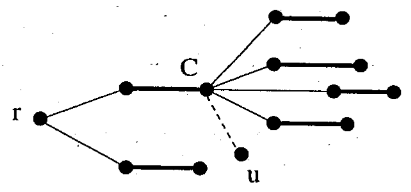

Figure 5.10. Tree structure after shrinking

the shrinking subroutine, including the updates of the matching and tree.

Use vw to shrink and update M' and T

Input: A matching M' of a graph G', an M'-alternating tree T, and an edge vw of G' such that $v, w \in B(T)$.

Action: Let C be the circuit formed by vw together with the path in T from v to w. Replace G' by $G' \times C$, M' by $M'\backslash E(C)$, and T by the tree (in G') having edge-set $E(T)\backslash E(C)$.

The next result confirms the intuition suggested by the examples. Its proof is Exercise 5.11.

Proposition 5.9 *After application of the shrinking subroutine, M' is a matching of G', T is an M'-alternating tree of G', and $C \in B(T)$.* ∎

We are now ready to state the algorithm. In addition to the subroutines we have mentioned, it uses an algorithm to extend a matching of a derived graph of G to a matching of G. Of course, this algorithm repeatedly uses the proof of Proposition 5.4.

Blossom Algorithm for Perfect Matching

```
Input graph $G$ and matching $M$ of $G$;
Set $M' = M$, $G' = G$;
Choose an $M'$-exposed node $r$ of $G'$ and put $T = (\{r\}, \emptyset)$;
While there exists $vw \in E'$ with $v \in B(T)$, $w \notin A(T)$
    Case: $w \notin V(T)$, $w$ is $M'$-exposed
        Use $vw$ to augment $M'$;
        Extend $M'$ to a matching $M$ of $G$;
        Replace $M'$ by $M$, $G'$ by $G$;
        If there is no $M'$-exposed node in $G'$
            Return the perfect matching $M'$ and stop;
        Else
            Replace $T$ by $(\{r\}, \emptyset)$, where $r$ is $M'$-exposed;
    Case: $w \notin V(T)$, $w$ is $M'$-covered
        Use $vw$ to extend $T$;
    Case: $w \in B(T)$
        Use $vw$ to shrink and update $M'$ and $T$;
Return $G', M', T$ and stop, $G$ has no perfect matching.
```

Theorem 5.10 *The Blossom Algorithm terminates after $O(n)$ augmentations, $O(n^2)$ shrinking steps, and $O(n^2)$ tree-extension steps. Moreover, it determines correctly whether G has a perfect matching.*

Proof: It is easy to see that M' remains a matching throughout. Therefore, since each augmentation decreases the number of exposed nodes, there will be $O(n)$ augmentation steps. Between augmentations, each shrinking step decreases the number of nodes of G' while not changing the number of nodes not in T, and each tree-extension step decreases the number of nodes not in T while not changing the number of nodes of G'. Hence, the number of shrinking and tree-extension steps between augmentations is $O(n)$, and the total number of these steps is $O(n^2)$.

Finally, since each G' is a derived graph of G, if the algorithm halts with the conclusion that G has no perfect matching, then it has found a tree with the properties of Proposition 5.8, and so G has no perfect matching. ∎

The Blossom Algorithm for Maximum Matching

We have yet to explain how to solve the maximum matching problem. In fact, we can use the algorithm for perfect matching. Of course, when we apply that algorithm to G, if it terminates with a perfect matching, then that matching is maximum. If it terminates with a matching and a frustrated

tree T in some derived graph, then we can extend the current matching to a matching of G, but it need not be maximum, because there may exist augmenting paths in other parts of G. Instead, we delete $V(T)$ from G'. If any exposed node remains, then we apply the Blossom Algorithm to the new G' beginning with the matching consisting of the edges of M' that were not deleted. (Notice that the new G' has no pseudonodes.) We continue this process until there are no exposed nodes left. Suppose that we now restore to G' all of the deleted parts including all of the matching edges. Let $T_1, \ldots, T_k$ be the frustrated trees we generated. Notice that the exposed nodes are exactly the roots of the T_i, so there are exactly k of them. Therefore, M' can be extended to a matching M of G having just k exposed nodes. Now let A be the union of the sets $A(T_i)$ for all i. If we delete A from G, each of the nodes of $B(T_i)$ for every i will give an odd component, and so we will have $\mathrm{oc}(G \backslash A) \geq |A| + k$. (In fact, this will turn out to hold with equality.) It follows that M is a maximum matching of G. (Notice that this argument provides the promised algorithmic proof of the Tutte-Berge Formula.)

Implementation of the Blossom Algorithm

Of course, the main new issue that comes up when we consider how to implement the matching algorithm, is how to represent the current derived graph. One natural way is to keep an explicit representation. That is, each time we shrink a circuit C, we would construct a representation for $G' \times C$ from the (known) representation of G'. This approach can be quite expensive in the worst case, but it also has some advantages. We will discuss it in more detail when we consider implementation of algorithms for minimum-weight perfect matching in Section 5.3. Another approach is to use a representation for G and to represent G' implicitly by keeping a representation of the partition of V determined by the sets $S(v)$ for all $v \in V(G')$. This is the approach that we will discuss here. The representation of G that we need will allow in constant time, given an edge e, to find its ends in G, as well as to find, for each $v \in V$, the edges in $\delta(v)$ in time proportional to their number.

When we shrink, the effect on the partition we are keeping is to merge some of its blocks. The operation that, given an edge $vw \in E$, finds the ends of that edge in the derived graph just requires identifying the sets $R(v)$ and $R(w)$, where $R(v)$ is the member of the partition containing v. Notice, then, that two of the basic operations required in the algorithm are the "find" and "merge" operations that we needed in Section 2.1 to implement Kruskal's Algorithm for the minimum spanning tree problem. This is an extremely well studied problem; see Tarjan [1983] for more details. We saw a method in Section 2.1 that can do $O(m)$ finds and $O(n)$ merges in time $O(m \log n)$. (In fact, a second method that accomplishes this was the subject of Exercise 2.15. That method turns out to be more important, in that it is the basis for more sophisticated methods which have better complexity. See Tarjan [1983] for more details.) We will evaluate the complexity of the matching algorithm