Mathematical Programming Study 8 (1978) 50–72.
North-Holland Publishing Company

A PRIMAL ALGORITHM FOR OPTIMUM MATCHING

W.H. CUNNINGHAM*

Carleton University, Ottawa, Canada

A.B. MARSH, III

The Johns Hopkins University, Baltimore, MD U.S.A.

Received 7 December 1976
Revised manuscript received 12 May 1978

Dedicated to the memory of D. Ray Fulkerson

An algorithm for finding an optimum weight perfect matching in a graph is described. It differs from Edmonds' "blossom" algorithm in that a perfect matching is at hand throughout the algorithm, and a feasible solution to the dual problem is obtained only at termination. In many other respects, including its efficiency, it is similar to the blossom algorithm. Some advantages of this "primal" algorithm for certain post-optimality problems are described. The algorithm is used to prove that, if the weights are integers, then the dual problem has an optimal solution which is integer-valued. Finally, some graph-theoretic results on perfect matchings are derived.

Key words: Optimum Matching, Primal Algorithm, Polyhedral Combinatorics, Integer Programming, Post-Optimality, Graph Theory.

1. Introduction

Let G be a finite, undirected, loopless graph. We denote its vertex-set by $V(G)$ and its edge-set by $E(G)$. A *matching* in G is a subset of edges, no two of which are incident with a common vertex. A matching M is *perfect* if every vertex is incident with exactly one member of M. Given real weights c_j for $j \in E(G)$, the *optimum perfect matching problem* is to maximize $(\Sigma\,(c_j\colon j \in M)\colon M$ a perfect matching).

Optimum matching problems, which consist of this problem and a number of close relatives, constitute the only class of genuine integer programs for which good solution algorithms are known[1]. These solution algorithms, usually called

* Research was done while this author was with the Department of Mathematical Sciences, Johns Hopkins University, and was supported in part by National Science Foundation Grant MCS76-08803.

[1] In response to a referee's query, the following explanation of this remark is provided. Every other class of well-solved combinatorial problems is not "genuine" because either: (a) No explicit formulation as an integer program using a reasonable amount of data is known (example: minimum spanning tree problems); or (b) When such a formulation is known, the resulting linear program already has integer-valued optimal solutions (example: network flow problems).

"blossom" algorithms, are due to Jack Edmonds; blossom methods for the problem treated here are described in [6], [7], [8], while those for more general matching problems occur in [9], [11], [12]. Aside from its intrinsic importance, optimum matching has been applied in the solution of certain shortest path problems and the Chinese postman problem [10], and in a heuristic algorithm for the Euclidean traveling salesman problem [4].

In this paper we describe a new algorithm for optimum perfect matching, which we call a *primal* algorithm. It is "primal" because it maintains a feasible solution (that is, a perfect matching), and obtains a feasible solution to a certain dual problem only at termination. In contrast, the blossom algorithm maintains a feasible solution to the dual problem, and obtains a perfect matching only at termination. The blossom algorithm for optimum perfect matching, specialized to the instance in which G is a bipartite graph, is the well-known Hungarian Method of Kuhn [15]. The primal algorithm, similarly specialized, is an algorithm of Balinski and Gomory [2]. While the generalization from the method of [2] to the present one is substantial, it is closely analogous to Edmonds' generalization of the Hungarian Method; many of the techniques used here were introduced in [6] and [7].

Sections 2 through 6 contain the description, justification, and discussion of the primal algorithm. The next three sections contain applications of the primal algorithm. In Section 7, we demonstrate the usefulness of the algorithm as a post-optimality procedure. We consider the situation in which an optimum perfect matching problem has been solved, and then the weights have been changed on a subset A of the edges. We show that, given a set $U \subseteq V(G)$ such that U "covers" A, a variant of the primal algorithm has a computation bound for re-optimizing which is the bound for solving "from scratch", multiplied by $|U|/|V(G)|$. In another application, we show that, whenever the weights c_j are all integers, there exists an integer-valued optimal dual solution, strengthening a result of Edmonds and Johnson. In Section 9 we use the primal algorithm to prove a purely graph-theoretic result on perfect matchings; this result is used to derive some known lower bounds on the number of perfect matchings in certain graphs. In the final section some computational results are reported. Except for the use of some standard graph-theoretic terminology, the paper is self-contained; a familiarity with the blossom algorithm on the part of the reader would be useful but not essential.

2. Preliminaries: Polyhedral combinatorics

In this section we describe conditions which provide a good characterization of optimal perfect matchings. These are the same conditions which the blossom algorithm uses, but the primal algorithm uses them in a different way.

For $S \subseteq V(G)$, $\delta_G(S)$ denotes the set of edges having exactly one end in S, and

$\gamma_G(S)$ denotes the set of edges having both ends in S. Whenever we can do so without loss of clarity, we will drop the subscript from δ and γ. For $v \in V(G)$, we abbreviate $\delta(\{v\})$ to $\delta(v)$. Where I is a finite set, J is a subset of I, and $p = (p_j : j \in I)$ is a real-valued vector, $p(J)$ denotes $\Sigma\,(p_j : j \in J)$.

The optimum weight perfect matching problem may be stated as an integer linear program in the following way.

$$\begin{array}{lll} & \text{maximize} & c \cdot x = \sum (c_j x_j : j \in E(G)), \\ (1) & \text{subject to} & x(\delta(v)) = 1, v \in V(G); \\ & & x_j \geq 0, \quad j \in E(G); \\ & & x_j \text{ integer}, \quad j \in E(G). \end{array}$$

Let $Q = \{S \subseteq V(G) : |S| \geq 3,\ |S| \text{ odd}\}$. For $S \in Q$, let q_S denote $\frac{1}{2}(|S| - 1)$. For any x feasible to (1), it is obvious that

$$(2) \qquad x(\gamma(S)) \leq q_S, \quad S \in Q.$$

By dropping the integrality requirement from (1) and adding the linear constraints (2), we obtain a linear program whose feasible solutions include all feasible solutions of (1). (In particular, if this linear program has an integer-valued optimal solution, then that solution is optimal to (1).) The dual of this linear program is

$$\begin{array}{lll} & \text{minimize} & \left(\sum (y_v : v \in V(G)) + \sum (q_S Y_S : S \in Q)\right) \\ (3) & \text{subject to} & Y_S \geq 0, \quad S \in Q; \\ & & \sum (y_v : j \in \delta(v)) + \sum (Y_S : j \in \gamma(S)) \geq c_j, \quad j \in E(G). \end{array}$$

If x is feasible to (1) and (y, Y) is feasible to (3) and the following "complementary slackness" conditions are also satisfied, then it is easy to show that x is optimal to (1) (and (y, Y) is optimal to (3)).

$$(4) \qquad \text{If} \quad x_j > 0, \text{ then } \sum (y_v : j \in \delta(v)) + \sum (Y_S : j \in \gamma(S)) = c_j, \quad j \in E(G);$$

$$(5) \qquad \text{If} \quad Y_S > 0, \quad \text{then } x(\gamma(S)) = q_S, S \in Q.$$

Given $y = (y_v : v \in V(G))$ and $Y = (Y_S : S \in Q)$, for $j \in E(G)$, let $d_j(y, Y)$ denote $\Sigma\,(y_v : j \in \delta(v)) + \Sigma\,(Y_S : j \in \gamma(S)) - c_j$. (Whenever possible we will abbreviate $d_j(y, Y)$ to d_j.) Since any x feasible to (1) is the incidence vector of a perfect matching M of G, we can translate (4), (5) into (4′), (5′):

$$(4') \qquad \text{If } j \in M, \quad \text{then } d_j = 0, \quad \text{for } j \in E(G);$$

$$(5') \qquad \text{If } Y_S > 0, \quad \text{then } |M \cap \gamma(S)| = q_S, \quad \text{for } S \in Q.$$

Edmonds has proved that, if G has a perfect matching, then there exists a perfect matching M and (y, Y) feasible to (3) satisfying (4′), (5′). The proof is an efficient algorithm, called the *blossom* algorithm, which maintains a matching M, and (y, Y) feasible to (3) such that (4′) and (5′) are satisfied, and terminates when M becomes perfect (or it is shown that no perfect matching exists). Thus the blossom algorithm relaxes one of the optimality conditions, the equality constraint of (1), and works toward satisfying it. The algorithm we will describe maintains a perfect matching M and (y, Y) satisfying (4′), (5′) and $Y \geq 0$, but initially does not require that $d_j \geq 0$ for $j \in E(G)$. In consequence, the values of the objective functions of (1) and (3) at each stage of the primal algorithm will be equal. (This does not happen in the blossom algorithm until optimality is reached.)

3. Preliminaries: Graph theory

Some of the graph-theoretic terminology used in this paper will not be defined here because we believe its use to be standard. In what follows we introduce some definitions and notation which are, perhaps, not so well-known. A *path* in a graph G is a sequence $P = v_0, e_1, v_1, \dots, e_n, v_n$ such that $\{v_0, v_1, \dots, v_n\}$, denoted $V(P)$, is a subset of $V(G)$; $\{e_1, e_2, \dots, e_n\}$, denoted $E(P)$, is a subset of distinct elements of $E(G)$; for $1 \leq i \leq n$, the ends of e_i in G are v_{i-1} and v_i. The path is said to be *from* v_0 *to* v_n and to have *length* n. P is *simple* if $|V(P)| = n + 1$. We say that $e_i \in E(P)$ is an *even* or *odd* edge of P according to whether i is even or odd. The path P is a *circuit* if $|V(P)| = n$, $v_0 = v_n$, and $n \geq 1$. A *polygon* is a subgraph whose edges and vertices are the edges and vertices of a circuit; the polygon is *even* or *odd* according to whether its vertex-set has even or odd cardinality. A *rooted tree* of G is a tree T having a distinguished *root* vertex r. An edge (or vertex) of T is *even* or *odd* according to whether it is an even or odd edge (or vertex) of a path in T from r to a vertex of T.

Where G is a graph, let S be a subset of $V(G)$. The graph $G[S]$ obtained by *restricting* G *to* S is the subgraph of G having $V(G[S]) = S$ and $E(G[S]) = \gamma(S)$. The graph $G \times S$ obtained from G by *shrinking* S is defined by: $V(G \times S) = (V(G) \setminus S) \cup \{S\}$, $E(G \times S) = E(G) \setminus \gamma(S)$, and $\delta_{G \times S}(v) = \delta(v)$ for each $v \in V(G \times S)$. A family $\mathcal{S}$ of subsets of $V(G)$ is said to be *nested* if $S_1, S_2 \in \mathcal{S}$ and $S_1 \cap S_2 \neq \emptyset$ implies that $S_1 \subseteq S_2$ or $S_2 \subseteq S_1$. Given a nested family $\mathcal{S}$ of subsets of $V(G)$ having maximal members $S_1, S_2, \dots, S_k$, the graph $G \times \mathcal{S}$ is defined to be $(\dots((G \times S_1) \times S_2) \dots \times S_k)$. It is easy to see that the order in which these (disjoint) sets are shrunk is irrelevant. Given a member S of a nested family $\mathcal{S}$, the nested family $\mathcal{S}[S]$ obtained by *restricting* $\mathcal{S}$ to S is defined to be $\{R \in \mathcal{S}: R \subset S\}$. The maximal members of $\mathcal{S}$ are called *pseudo vertices* of $G \times \mathcal{S}$; the other vertices of $G \times \mathcal{S}$, that is, the elements of $V(G \times \mathcal{S}) \cap V(G)$, are called *real* vertices of $G \times \mathcal{S}$.

The nested families $\mathcal{S}$ which we will be using have the following property:

(6) For each $S \in \mathcal{S}$, $G[S] \times \mathcal{S}[S]$ is spanned by an odd polygon $P(S)$.

A nested family $\mathcal{S}$ satisfying (6) is called a *shrinking* family. We always assume that we actually have $P(S)$ at hand for each $S \in \mathcal{S}$; that is, the graphs $P(S)$, for $S \in \mathcal{S}$, are part of the information we remember with $\mathcal{S}$. The reason that such families are appropriate to the study of perfect matchings is given by the following result.

(7) Theorem. *Let $\mathcal{S}$ be a shrinking family of G and let M be a perfect matching of $G \times \mathcal{S}$. Then M is contained in a perfect matching M_1 of G.*

Proof. The result is true if $\mathcal{S} = \emptyset$, so assume $\mathcal{S} \neq \emptyset$ and let S be a maximal member of $\mathcal{S}$. Then $\mathcal{S}' = \mathcal{S} \setminus \{S\}$ is a shrinking family of G. Moreover, M is a matching of $G \times \mathcal{S}'$; let v be the vertex of $G[S] \times \mathcal{S}[S]$ incident with the edge $e \in M$ which is incident with S in $G \times \mathcal{S}$. Then the set of vertices of $G \times \mathcal{S}'$ not incident with a member of M is just $V(G[S] \times \mathcal{S}[S]) \setminus \{v\}$. But $G[S] \times \mathcal{S}[S]$ is spanned by the odd polygon $P(S)$, so there is a (unique) matching M_2 of $P(S)$ such that v is the only vertex of $P(S)$ not incident with an element of M_2. When M is replaced by $M \cup M_2$ and $\mathcal{S}$ by $\mathcal{S}'$, then (6) is still satisfied. Continuing the process, we obtain a perfect matching M_1 of G, as required.

(8) It is implicit in the proof of (7), that the matching M_1 constructed there has the properties that $|M_1 \cap \gamma(S)| = q_S$ for each $S \in \mathcal{S}$, and $M_1 \setminus M \subseteq \bigcup(E(P(S)) \colon S \in \mathcal{S})$.

Another result on nested families, which will be useful, is a bound on their cardinality. The following is easy to prove by induction on $|\mathcal{S}|$; a proof is given in [18]. It is clear that any shrinking family $\mathcal{S}$ satisfies the hypothesis of (9).

(9) If $\mathcal{S}$ is a nested family of subsets of $V(G)$ such that $|S| \geq 3$ for each $S \in \mathcal{S}$ and $|S_1| \geq |S_2| + 2$ whenever $S_1, S_2 \in \mathcal{S}$ and $S_1 \supset S_2$, then $|\mathcal{S}| \leq \frac{1}{2}(|V(G)| - 1)$.

4. Some graph-theoretic subroutines

The primal algorithm will keep a perfect matching M' of G implicitly, by keeping a shrinking family $\mathcal{S}$ of G and a perfect matching M of $G \times \mathcal{S}$. In this section we introduce the main subroutines for changing $\mathcal{S}$ and M, and the main device used for finding these changes, the growth of alternating trees. We will not concern ourselves in this section with the dual problem or the optimality conditions introduced in Section 2; ultimately these will be combined with the methods of this section.

Let $\mathcal{S}$ be a shrinking family of G and let M be a perfect matching of $G \times \mathcal{S}$. A rooted tree T of $G \times \mathcal{S}$ is an *M-alternating tree* if

(10) $M \cap E(T)$ is a perfect matching of T;

(11) Every odd edge of T is in M.

It is an easy consequence of the definition that every even vertex of an M-alternating tree other than the root is incident with exactly two edges of the tree, and that the root is incident with just one edge of the tree. An M-alternating tree having root r is illustrated in Fig. 1(a). (In Fig. 1, except for 1(b), the edges of the matching are thick edges, the edges of the tree are the solid edges, the even vertices of the tree are solid, and the odd vertices of the tree are square.) The alternating trees used here are closely related to, but different from, those of [6]. Given an M-alternating tree T, we define $O(T)$ to be the subset of $V(G)$ consisting of real odd vertices of T and elements of odd pseudo vertices of T; $I(T)$ is defined similarly, with "odd" replaced by "even".

Aside from the dual solution (y, Y), the main objects kept by the algorithm will be a shrinking family $\mathcal{S}$ of G, a perfect matching M of $G \times \mathcal{S}$, and an M-alternating tree T of $G \times \mathcal{S}$. The tree T will always have root r, and r will be determined by $\mathcal{S}$ and a distinguished vertex $u \in V(G)$ as follows: $r = u$ if u is a real vertex of $G \times \mathcal{S}$, and otherwise r is the pseudo vertex of $G \times \mathcal{S}$ of which u is an element. We now describe several basic subroutines for manipulating $\mathcal{S}$, M, and T. It is straightforward to verify that each of (12)–(15) produces $\mathcal{S}$, M, T having the required structure.

(12) *Grow T using e.* Given an edge e joining in $G \times \mathcal{S}$ an odd vertex v of T to a vertex w not in $V(T)$, let f be the element of M incident with w. Let T' be the tree in $G \times \mathcal{S}$ whose edge-set is $E(T) \cup \{e, f\}$. Replace T by T'.

(Procedure (12) is illustrated in Fig. 1; beginning with T of Fig. 1(a) and using e_1 to grow T, we obtain T of Fig. 1(c).)

(13) *Shrink using e.* Given an edge e joining in $G \times \mathcal{S}$ two odd vertices v, w of T, let P be the (odd) polygon whose edge-set is $E(P_1) \cup \{e\}$, where P_1 is the path in T from v to w. Let

$$S = (V(P) \cap V(G)) \cup (\bigcup \{R : R \in V(P) \cap \mathcal{S}\});$$

let $\mathcal{S}' = \mathcal{S} \cup \{S\}$. Let T' be the tree in $G \times \mathcal{S}'$ whose edge-set is $E(T) \cap E(G \times \mathcal{S}')$; let $M' = M \cap E(G \times \mathcal{S}')$. Replace $\mathcal{S}$ by $\mathcal{S}'$, M by M', and T by T'.

(Procedure (13) is illustrated in Fig. 1; beginning with T of Fig. 1(a) and carrying out "Shrink using e_2", we obtain T' of Fig. 1(d). In labeling the new pseudo vertex $S \cup \{p, q\}$, we are implicitly assuming that p, q are real vertices of $G \times \mathcal{S}$.)

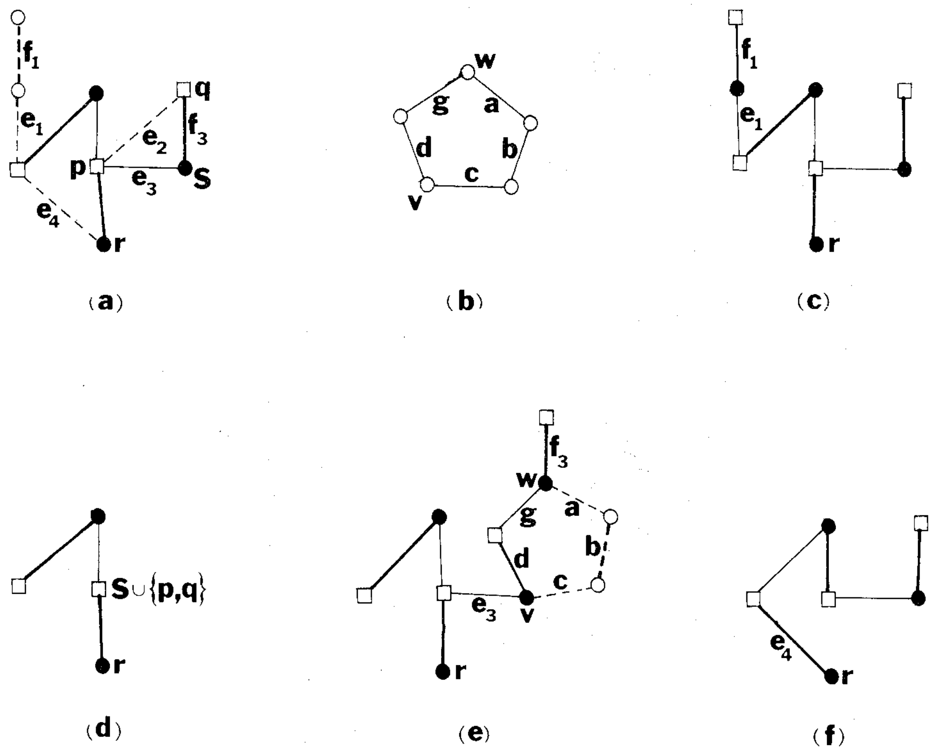

Fig. 1.

(14) *Expand even pseudo vertex S.* Given an even vertex S of T such that $S \in \mathcal{S}$, let f be the edge of M incident with S in $G \times \mathcal{S}$ and let e be the other edge of T, if any, incident with S. Let $\mathcal{S}' = \mathcal{S} \setminus \{S\}$. Let v be the end of e in $G \times \mathcal{S}'$ which is in $V(P(S))$, if e exists; otherwise let v be the member of $\mathcal{S}'$ containing u or, if none, let $v = u$. Let w be the end of f in $G \times \mathcal{S}'$ which is in $V(P(S))$. Let P_1 be the even-length path from v to w in $P(S)$, and let P_2 be the odd-length path from v to w in $P(S)$. Let M' denote M together with the odd edges of P_1 and the even edges of P_2. Let T' be the tree in $G \times \mathcal{S}'$ such that $E(T') = E(T) \cup E(P_1)$. Replace $\mathcal{S}$ by $\mathcal{S}'$, M by M', and T by T'. If $S = r$, replace r by v.

(Procedure (14) is illustrated in Fig. 1; beginning with T of Fig. 1(a) and using $P(S)$ shown in Fig. 1(b), we obtain T' of Fig. 1(e). Here e_3, f_3 are the e, f of (14), and v, w have the same meaning as in (14). In Fig. 1(e) some relevant non-tree edges have also been drawn. The special case in which $S = r$ has not been illustrated; the reader is encouraged to investigate this case for himself.)

(15) *Augment using e.* Given an edge e joining in $G \times \mathcal{S}$ an odd vertex v of T to r, let f be the element of M incident with r and let P be the path in T from r to v. Let M' be the matching obtained by deleting from M the odd edges of P and adding e and the even edges of P. Let T' be the tree in $G \times \mathscr{S}$ such that $E(T') = (E(T) \cup \{e\}) \setminus \{f\}$. Replace M by M' and T by T'.

(Procedure (15) is illustrated in Fig. 1; beginning with T of Fig. 1(a), "Augment using e_4" has been carried out, resulting in the changes in T and M shown in Fig. 1(f).)

(16) *Extend M to perfect matching M_1 of G.* This subroutine is the procedure which is implicit in the proof of (7).

5. The primal algorithm

In this section we relate the optimality conditions described in Section 2 to the graph-theoretic routines of the last section. This leads to the statement of the primal algorithm. The algorithm maintains real vector $y = (y_v : v \in V(G))$, non-negative real vector $Y = (Y_S : S \in Q)$, shrinking family $\mathscr{S}$, and perfect matching M of $G \times \mathscr{S}$. In addition, we require

(17) If $j \in M$ or $j \in E(P(S))$ for some $S \in \mathscr{S}$, then $d_j = 0$;

(18) If $S \notin \mathscr{S}$, then $Y_S = 0$.

It follows from (17) together with (7) and (8) that (4′) is satisfied implicitly. Similarly, it follows from (18) together with (7) and (8) that (5′) is satisfied implicitly. The algorithm works toward the optimality criterion $d_j \geq 0$, $j \in E(G)$; once an edge j attains this "dual feasibility" property, d_j never again becomes negative. The general strategy is to choose a vertex $u \in V(G)$ and work toward obtaining $d_j \geq 0$ for all $j \in \delta(u)$. We do this by growing an M-alternating tree T (the root r of T is determined by u and $\mathscr{S}$ as previously indicated). In order to facilitate maintaining (17) after changes in $\mathscr{S}$ or M, we require that $d_j = 0$ for $j \in E(T)$. The main components of the algorithm which have not yet been described are changes in (y, Y): the "dual change" of (24), and the "mini dual change" contained in (25)–(27). The dual change step enables further application of the steps (21), (22), or (23) and can also help directly to attain $d_j \geq 0$ for $j \in \delta(u)$. The bound α on the "amount" ϵ of the dual change (24) reflects the fact that no $d_j \geq 0$ is allowed to become negative; the bound β ensures that the amount ϵ does not exceed the maximum amount for which the dual change will be beneficial in achieving $d_j \geq 0$ for $j \in \delta(u)$. Notice that the objective value in (3) is not altered by a dual change, but is raised by a mini dual change. The algorithm has been constructed so that a mini dual change will be done only if it will result in $d_j \geq 0$ for all $j \in \delta(u)$; this is the reason for the calculation of σ in (20). In taking maxima and minima of sets of non-negative real numbers, we

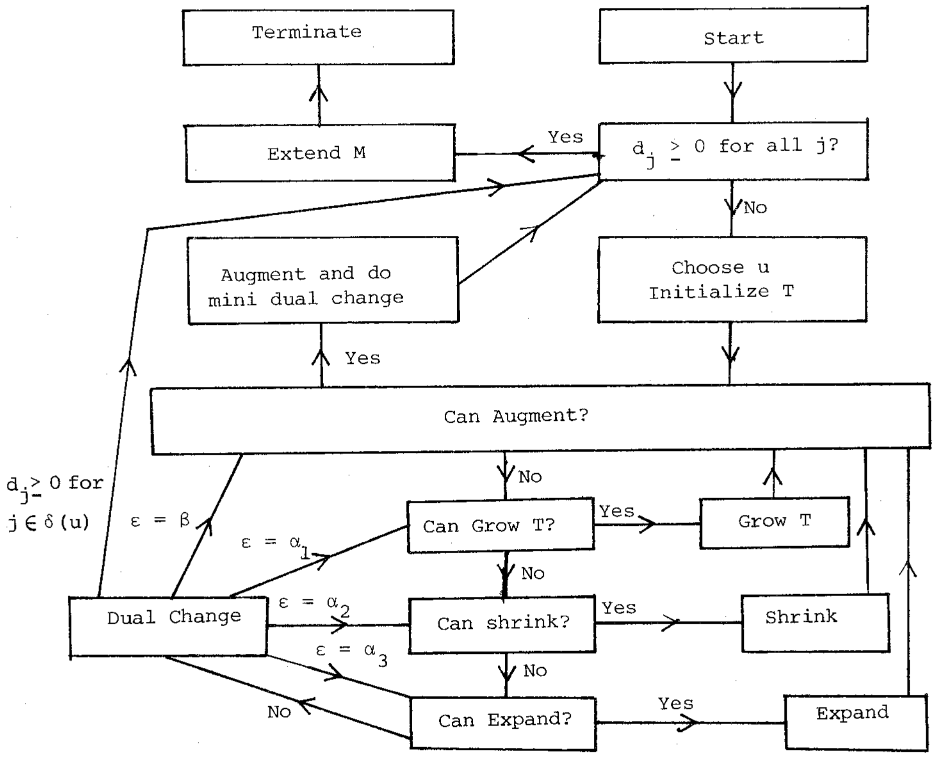

Fig. 2.

observe the convention that the maximum of an empty set of numbers is zero, and the minimum of an empty set of numbers is infinity. A flow chart of the primal algorithm is shown in Fig. 2.

Primal algorithm for optimum perfect matching

(We assume that we have initially y, Y, $\mathcal{S}$, and M as described above.)

(19) If $d_j \geq 0$ for all $j \in E(G)$, go to (28). Otherwise choose $u \in V(G)$ such that, for some $j \in \delta(v)$, $d_j < 0$. Let r be the maximal member of $\mathcal{S}$ containing u, if one exists; otherwise let $r = u$. Let T be the tree in $G \times \mathcal{S}$ having root r whose only edge is the element of M incident with r.

(20) Let $\sigma_1 = \max(-d_j\colon d_j < 0, j \in \delta(u)$, and, in $G \times \mathcal{S}$, j joins r to a vertex not in $V(T))$; let $\sigma_2 = \max(-d_j\colon d_j < 0,\ j \in \delta(u)$, and in $G \times \mathcal{S}$, j joins r to an even vertex of $T)$; let $\sigma_3 = \max(\frac{1}{2}\Sigma\,(Y_S\colon j \in \gamma(S)) - d_j\colon d_j < 0,\ j \in \delta(u) \cap \gamma(r))$; let $\sigma = \max(\sigma_1, \sigma_2, \sigma_3)$. If there exists $e \in \delta(u)$ such that, in $G \times \mathcal{S}$, e joins r to an odd vertex v of T and $-d_e \geq \sigma$, go to (25). Otherwise, go to (21).

(21) If there exists $e \in E(G \times \mathcal{S})$ such that $d_e = 0$ and, in $G \times \mathcal{S}$, e joins an odd vertex of T to a vertex not in $V(T)$, grow T using e and go to (20). Otherwise, go to (22).

(22) If there exists $e \in E(G \times \mathcal{S})$ such that $d_e = 0$ and, in $G \times \mathcal{S}$, e joins two odd vertices of T, shrink using e and go to (20). Otherwise, go to (23).

(23) If there exists $S \in \mathcal{S}$ such that S is an even vertex of T and $Y_S = 0$, expand S and go to (20). Otherwise, go to (24).

(24) Let $\alpha_1 = \min(d_j\colon d_j \geq 0$; in $G \times \mathcal{S}$, j joins an odd vertex of T to a vertex not in $V(T))$; let $\alpha_2 = \min(\frac{1}{2}d_j\colon d_j \geq 0$; in $G \times \mathcal{S}$, j joins two odd vertices of $T)$; let $\alpha_3 = \min(\frac{1}{2}Y_S\colon S \in \mathcal{S}$, S an even vertex of $T)$; let $\alpha = \min(\alpha_1, \alpha_2, \alpha_3)$. Let $\beta_1 = \max(-d_j\colon d_j < 0$, $j \in \delta(u)$; in $G \times \mathcal{S}$, j joins r to a vertex not in $V(T))$; let $\beta_2 = \max(-\frac{1}{2}d_j\colon d_j < 0, j \in \delta(u)$; in $G \times \mathcal{S}$, j joins r to an even vertex of $T)$; if there exists $j \in \delta(u) \cap \gamma(r)$ with $d_j < 0$, let $\beta_3 = Y_r$, and otherwise let $\beta_3 = 0$; let $\beta = \max(\beta_1, \beta_2, \beta_3)$. Let $\epsilon = \min(\alpha, \beta)$. Let $y'_v = y_v + \epsilon$ for each $v \in I(T)$; let $y'_v = y_v - \epsilon$ for each $v \in O(T)$; for every other $v \in V(G)$, let $y'_v = y_v$. Let $Y'_S = Y_S - 2\epsilon$ for every $S \in \mathcal{S}$ such that S is an even vertex of T; let $Y'_S = Y_S + 2\epsilon$ for every $S \in \mathcal{S}$ such that S is an odd vertex of T; for every other $S \in Q$, let $Y'_S = Y_S$. Replace (y, Y) by (y', Y'). If $d_j \geq 0$ for all $j \in \delta(u)$, go to (19). Otherwise, if $\epsilon = \alpha_1$, go to (21); if $\epsilon = \alpha_2$, go to (22); if $\epsilon = \alpha_3$, go to (23); if $\epsilon = \beta$, go to (20).

(25) If $r = u$, replace y_u by $y_u - d_e$, augment using e, and go to (19).

(26) If $r \neq u$ and $Y_r \geq -2d_e$, replace Y_r by $Y_r + 2d_e$ replace y_v by $y_v - d_e$ for each $v \in r$, augment using e, and go to (19). Otherwise go to (27).

(27) If $r \neq u$ and $Y_r < -2d_e$, replace Y_r by 0, replace y_v by $y_v + \frac{1}{2}Y_r$ for each $v \in r$, expand r, and go to (25).

(28) Extend M to a perfect matching M_1 of G. Terminate; M_1 is an optimal perfect matching, and (y, Y) is an optimal solution to (3).

6. Discussion of the algorithm

(29) *Correctness and bound.* It is straightforward to check that each step of the algorithm which modifies any of $\mathcal{S}$, (y, Y), M, or T preserves the properties required of them. Given that this is so, we can see that, if the algorithm terminates, it finds an optimal perfect matching. We now show that the algorithm is finite. It is easy to see that, once $j \in E(G)$ satisfies $d_j \geq 0$, d_j never becomes negative. Since the value of u is not changed until we have $d_j \geq 0$ for all $j \in \delta(u)$, the algorithm goes through (19) at most $|V(G)|$ times. (If we have a relatively small set $U \subseteq V(G)$ such that $E(G) = \gamma(U) \cup \delta(U)$, we can obtain a smaller bound by letting u run through U. However, it is not advisable to attempt to find a minimum cardinality such U: this is well-known to be a difficult problem.)

Now consider a stage of the algorithm during which u does not change. Let $\mathcal{J}$ denote $\{S \in \mathcal{S}$: S is not contained in a pseudo vertex of $G \times \mathcal{S}$ which is an odd vertex of $T\}$. Either a tree-growth step or a shrinking step increases $|O(T)|$ without increasing $|\mathcal{J}|$; an expanding step decreases $|\mathcal{J}|$ without decreasing $|O(T)|$. Thus each iteration of one of these increases $|O(T)| - |\mathcal{J}|$ by at least one. Now $\mathcal{J}$ is a nested family of subsets of $V(G) \setminus O(T)$, so by (9), since $1 \leq |O(T)| \leq |V(G)| - 1$, we have $0 \leq |\mathcal{J}| \leq \frac{1}{2}(|V(G)| - 2)$. Thus $|O(T)| - |\mathcal{J}|$ has a maximum value of $|V(G)| - 1$ and a minimum value of $2 - \frac{1}{2}|V(G)|$. Since, between occurrences of (20), a tree-growth, shrinking, or expanding step is performed, there can be at most $\frac{3}{2}|V(G)| - 3$ occurrences of (20), during a stage in which u does not change. Also, an occurrence of (24) enables a tree-growth, shrinking, expanding, or augmenting step. Thus, during the entire algorithm, there are at most $\frac{3}{2}|V(G)|^2$ occurrences of (20), (21), (22), (23), or (24); also, there are at most $|V(G)|$ occurrences of (25)–(27). An ample bound for the work involved in an application of (20), (21), (22), (23), or (24) is $O(|E(G)|)$; an ample bound for the work involved in an application of (25)–(27) is $O(|V(G)| \cdot |E(G)|)$. Since the amount of work involved in (19) and (28) is dominated by the amount of work in the rest of the algorithm, a computation bound for the primal algorithm of $O(|V(G)|^2 \cdot |E(G)|)$ is established.

(30) *Implementations.* The further details of implementing the algorithm are not discussed here. It is not difficult to give an implementation which achieves a computation bound of $O(|V(G)|^2 \cdot |E(G)|)$, as claimed. Further improvement is possible; a bound of $O(|V(G)|^3)$ can be achieved, but considerable care is required.

Almost everything that can be said about implementational details for this algorithm applies equally to the blossom algorithm. Detailed discussions of an implementation of the blossom algorithm can be found in Lawler [16].

(31) *Initial solutions.* If we are given a perfect matching M of G, we can begin the algorithm by setting $\mathcal{S} = \emptyset$, $Y = 0$, and choosing y so that $d_j = 0$ for $j \in M$. If no perfect matching of G is known initially, we can form a graph G' from G by adding a set A of "artificial" edges, such that for some matching M of G (perhaps empty), $M \cup A$ is a perfect matching of G'. If the primal algorithm solves the optimum perfect matching problem on G' beginning with $M \cup A$, where the elements of A are given sufficiently small weights, it will terminate with either an optimal perfect matching of G, or a perfect matching containing at least one element of A; in the latter case G has no perfect matching. (It can be shown that any number less than

$$\tfrac{1}{2}|V(G)| \cdot (\min(c_j: j \in E(G)) - \max(0, \max(c_j: j \in E(G))))$$

is sufficiently small for this purpose.)

A particular instance of the above procedure occurs when we wish only to find some perfect matching in G, so that each $c_j = 1$ for $j \in E(G)$, and $c_j = 0$ for $j \in A$. The primal algorithm can be used here to find a maximum cardinality matching of G. It is natural to ask whether the resulting algorithm is equivalent to the blossom algorithm for maximum cardinality matching, described in [6]. In general, the answer is no, for in this case the primal algorithm can perform augmentations which extend the cardinality of the (non-artificial) matching by more than one. However, if the initial (non-artificial) matching is maximal, this cannot happen, and the resulting algorithm can be seen to be equivalent to the maximum cardinality version of the blossom algorithm. We encourage the reader to verify these claims.

(32) *A variant.* The augmentation and mini dual change can be carried out whenever $r = u$ and $e \in \delta(u)$ joining, in $G \times \mathscr{S}$, r to an odd vertex of T and having $d_e < 0$, is discovered. This could result in as many as $|\delta(u)|$ such steps during the growth of a single tree; however, in practice the number could be expected to be much smaller, and the number of tree-growing, shrinking, expanding, and dual-change steps might well be decreased. This variant of the primal algorithm has the same worst-case bound as the algorithm itself; tests have indicated that it offers no computational advantage. A similar variant for the bipartite case was introduced in [2] and an inferior bound was obtained; the reason is that, in [2], T is discarded rather than modified after an augmentation.

(33) *The simplex algorithm.* The alternating trees used in the primal algorithm are structurally the same as the basis trees encountered when a special form of the network simplex method [3], [5] is applied to the optimum perfect matching problem for bipartite graphs. (An obvious difference is that in the simplex method the trees are always spanning trees.) The augmentation step, in this case, is a special kind of "non-degenerate pivot" in the simplex method; a general non-degenerate pivot yields a more general sort of augmentation, engendered by an edge e with $d_e < 0$ joining an even vertex v of T to an odd vertex w of T, with the following important priviso: v must be a vertex of the path in T from w to the root. A "degenerate pivot", engendered by an edge e joining even and odd nodes but not satisfying the proviso, changes T and y without changing M. A property of these more general simplex pivots, which distinguishes them from the augmentation step of the primal algorithm, is that they can cause an edge j satisfying $d_j \geq 0$ to lose this property.

7. Post-optimality

It seems unlikely that the primal algorithm offers any computational advantage over the blossom algorithm in the solution of optimum perfect matching problems "from scratch". In this section we show, on the other hand, that the primal algorithm is computationally attractive in certain post-optimality situations. Given initial solutions that are "good" in a certain sense, we can significantly improve the worst-case performance of the primal algorithm. We wish to emphasize that "good initial solutions" does not mean merely an optimal or near-optimal perfect matching; we have no evidence to indicate, for example, that proving a matching to be optimal is easier than finding an optimal matching.

Suppose that the optimum perfect matching problem on G with weight vector c has been solved, yielding y, Y, $\mathscr{S}$, and M. Suppose further that it is desired to solve a similar problem on G, but with weight vector $c' = (c'_j \colon j \in E(G))$. For the old solution to be of some value in solving the new problem, we must assume that c' is "near" to c; the sense of nearness which we will assume is that we have $U \subseteq V(G)$ which is small in cardinality compared to $V(G)$, and such that, if $c_j \neq c'_j$, then $j \in \delta(U) \cup \gamma(U)$. We will describe a method, based on the primal algorithm, for solving the new problem which requires the growth of at most $|U|$ trees.

Perhaps the easiest way to explain the method is to change c to c', a few components at a time. We choose $u \in U$ and solve the problem obtained by replacing c_j by c'_j for each $j \in \delta(u)$ by growing at most one tree. After doing this for each $u \in U$ we will have solved the problem of interest. Therefore, we assume that $U = \{u\}$ for some $u \in V(G)$.

The solution to this simpler problem can be broken into two phases. *Phase* 2 handles the case in which u is a real vertex of $G \times \mathscr{S}$. Then, where f is the element of $M \cap \delta(u)$ and v is its other end in G, we replace y_u by $c'_f - y_v$. Then Y, $\mathscr{S}$, M and the new y are acceptable input to the primal algorithm for the new problem; since (where d'_j is defined as expected) $\{j \colon d'_j < 0\} \subseteq \delta(u)$, at most one tree will need to be grown to complete the solution to the new problem. (It should be pointed out that, in this special case in which u is a real vertex of $G \times \mathscr{S}$, the blossom algorithm gives a similarly simple solution to this post-optimality problem.)

Phase 1 handles the case in which u is an element of a pseudo vertex r of $G \times \mathscr{S}$. We will obtain alternative final solutions $y', Y', \mathscr{S}', M'$ for the problem with weight vector c, such that u is a real vertex of $G \times \mathscr{S}'$, thus enabling the application of Phase 2. The algorithm to do this is a "stripped-down" version of the primal algorithm.

Post-optimality, *Phase* 1

(34) Let r be the maximal member of $\mathscr{S}$ containing u. Let T be the tree in $G \times \mathscr{S}$ having root r whose only edge is the element of M incident with r in $G \times \mathscr{S}$.

(35) If u is a real vertex of $G \times \mathcal{S}$, stop. Otherwise, go to (36).

(36) If there exists $e \in E(G \times \mathcal{S})$ such that $d_e = 0$ and, in $G \times \mathcal{S}$, e joins an odd vertex of T to a vertex not in $V(T)$, grow T using e and go to (36). Otherwise, go to (37).

(37) If there exists $e \in E(G \times \mathcal{S})$ such that $d_e = 0$ and, in $G \times \mathcal{S}$, e joins two odd vertices of T, shrink using e and go to (36). Otherwise, go to (38).

(38) If there exists $S \in \mathcal{S}$ such that S is an even vertex of T and $Y_S = 0$, expand S and go to (35). Otherwise go to (39).

(39) Calculate α_1, α_2, α_3, and α as in (24). Let $\epsilon = \alpha$ and define y', Y' as in (24). Replace (y, Y) by (y', Y'). If $\epsilon = \alpha_1$, go to (36); if $\epsilon = \alpha_2$, go to (37); if $\epsilon = \alpha_3$, go to (38).

It is easy to see that the above algorithm will terminate with $y, Y, \mathcal{S}, M$ satisfying the optimality conditions and such that u is a real vertex of $G \times \mathcal{S}$. The computation bound is the same as the bound for a single stage (single value of u) of the primal algorithm; the proof is similar to the proof for the primal algorithm. We can now begin phase 2 of the post-optimality procedure; namely, we replace c by c', adjust y_u as indicated above, and apply the primal algorithm. The tree T with which phase 1 of the procedure terminated is a valid tree with which to begin phase 2; this justifies the claim that only one tree need be grown. In general, at most $|U|$ trees will be grown, and we obtain a bound for the post-optimality procedure of the order of $|U| \cdot |V(G)| \cdot |E(G)|$. (The techniques which enable the replacement of $|E(G)|$ by $|V(G)|$ in the bound for the primal algorithm would provide a similar improvement in this bound.)

8. Integer-valued optimal dual solutions

Two fundamental results in the theory of optimum matching are the following theorems of Edmonds.

(40) Theorem. *If G has a perfect matching, then*

$$\max(c(M)\colon M \text{ a perfect matching})$$

$$= \min(y(V(G)) + \sum (q_S Y_S\colon S \in Q)\colon Y_S \geq 0 \quad \text{for } S \in Q;$$

$$\sum (y_v\colon j \in \delta(v)) + (Y_S\colon j \in \gamma(S)) \geq c_j \quad \text{for } j \in E(G)).$$

(41) Theorem. *The convex hull of incidence vectors of perfect matchings (the "matching polytope") is*

$$\{x = (x_j : j \in E(G)) : x_j \geq 0 \text{ for } j \in E(G);$$
$$x(\delta(v)) = 1 \text{ for } v \in V(G);$$
$$x(\gamma(S)) \leq q_S \text{ for } S \in Q\}.$$

Theorem (40) was first proved in [7] using the blossom algorithm; the primal algorithm also yields a proof. Theorem (41) is a consequence of (40), in view of the standard linear programming result that any extreme point of a (convex) polyhedron uniquely maximizes some linear function over the polyhedron. (Recently, Pulleyblank and Edmonds [20] have strengthened (41) by characterizing those inequalities which are essential to the definition of the matching polytope.) On the other hand, in view of the linear programming duality theorem, (40) is a consequence of (41). Thus, for example, a number of recent direct (non-algorithmic) proofs of (41) ([1], [14], [22]), some of them elegant, provide proofs of (40). However, the algorithmic proofs of (40) have the advantage of yielding results rather stronger than (40) itself. The following result is an example; Theorem M of [7] goes even farther in this direction.

(42) Theorem. *There exists an optimal* (y, Y) *in* (40) *satisfying* (43) *and* (44) *below.*

(43) There exists a shrinking family $\mathcal{S}$ of G such that $Y_S > 0$ implies $S \in \mathcal{S}$;

(44) If a number d divides c_j for each $j \in E(G)$, then d divides Y_S for each $S \in Q$ and d divides $2y_v$ for each $v \in V(G)$.

The first part of (42) is an immediate consequence of either the blossom algorithm or the primal algorithm; it was proved in [7]. The second part of (42) was stated in [9] (there is a proof in [18]). It is not difficult to see that the condition (44) can be replaced in Theorem (42) by (45) below.

(45) If c is integer-valued, then Y is integer-valued and $2y$ is integer-valued.

We extend the result (42) with the following theorem.

(46) Theorem. *There exists an optimal* (y, Y) *in* (40) *satisfying* (43) *and such that, if* c *is integer-valued, then* (y, Y) *is integer-valued.*

We begin the proof of (46) by verifying (42) using the primal algorithm. The proof of the following result is analogous to the method of proof of (42) given in [18].

(47) Proposition. *If c is integer-valued, and the primal algorithm is begun with (y, Y) such that Y and $2y$ are integer-valued, then this property is preserved throughout the algorithm.*

Proof. It is enough to show that a dual change or a mini dual change cannot destroy the property of Y and $2y$ being integer-valued. With regard to the dual change operation (24), it suffices to show that 2ϵ will be an integer. We observe that, where T is an alternating tree constructed by the algorithm and v, w are elements of $I(T) \cup O(T)$, we have $y_v \equiv y_w \pmod 1$. It follows that $2\alpha_2$ and $2\beta_2$ will be integers. It is easy to see that $2\alpha_1$, $2\alpha_3$, $2\beta_1$, and $2\beta_3$ will also be integers, and thus that 2ϵ will be an integer. Thus a dual change step will not destroy the integrality of Y and $2y$. It is similarly easy to see that a mini dual change will not affect this property. The proof is complete.

Before describing an algorithmic proof of (46), we observe that a method similar to the proof of (42) will not suffice to prove (46), for it is possible for a dual change to destroy the property that y is integer-valued. (This could occur when $\epsilon = \alpha_2$ or α_3 in (24).) We give a description of the algorithm for finding an integer-valued optimal dual solution which is somewhat less formal than previous descriptions; the reader who has successfully reached this point in the paper should have no trouble understanding the method. We begin with $y, Y, \mathcal{S}, M$ at termination of the primal algorithm. We assume that the initial dual solution has been chosen so that (47) will ensure that Y and $2y$ are integer-valued. If y is integer-valued, we are done. Otherwise, we choose $u \in V(G)$ such that $y_u \equiv \frac{1}{2} \pmod 1$. We define r and T as in (20). We then perform steps (21), (22), (23) of the primal algorithm repeatedly, until no more such steps can be carried out. At this point, for every $v \in I(T) \cup O(T)$, we have $y_v \equiv \frac{1}{2} \pmod 1$. We now put $\epsilon = \frac{1}{2}$ and replace (y, Y) by (y', Y'), as in (24). This step maintains an integer-valued Y, keeps y_v integral if it was before, and makes y_v an integer for each $v \in I(T) \cup O(T)$. If there remains some $u \in V(G)$ with $y_u \equiv \frac{1}{2} \pmod 1$, the process is repeated.

The above algorithm is clearly finite, and so (46) is proved. In fact, the present algorithm has a computation bound which is better than the computation bound for the primal algorithm by a factor of $|V(G)|$. A vertex v enters $O(T)$ only if $y_v \equiv \frac{1}{2} \pmod 1$, and does not leave $O(T)$ until y_v becomes an integer. Since the steps (21) and (22) increase $|O(T)|$ and step (23) does not decrease $|O(T)|$, therefore steps (21) and (22) are performed at most $|V(G)|$ times. A set S which is added to $\mathcal{S}$ by (22) will never be a subset of $I(T)$ for any T; this is because S will remain a subset of $O(T)$ until y_v becomes an integer for each $v \in S$, at which point S will never become part of another tree T. Thus each S which is expanded by (23) must be a member of the $\mathcal{S}$ with which the present algorithm is initiated. It follows that (23) will be performed at most $\frac{1}{2}(|V(G)| - 1)$ times. Therefore, we have a bound of $O(|V(G)|)$ for the number of times (21), (22), (23) are performed in the present algorithm; the analogous bound in the primal algorithm is $O(|V(G)|^2)$.

We wish to obtain similar results for optimum not-necessarily-perfect matching. Here, in analogy to (40), Edmonds has proved the following.

(48) Theorem. *The following holds,*

$$\max(c(M)\colon M \text{ a matching}) =$$
$$= \min(y(V(G)) + \sum (q_S Y_S\colon S \in Q)\colon y_v \geq 0 \text{ for } v \in V(G);$$
$$Y_S \geq 0 \text{ for } S \in Q;$$
$$\sum (y_v\colon j \in \delta(v)) + \sum (Y_S\colon j \in \gamma(S)) \geq c_j \text{ for } j \in E(G)).$$

We will prove the following analogue of (46), extending a result of [7] and [9] which is analogous to (42).

(49) Theorem. *There exists an optimal* (y, Y) *in* (48) *satisfying* (43) *and such that, if* c *is integer-valued, then* (y, Y) *is integer-valued.*

Proof. Let U be a set of vertices of G such that there is an optimal matching M_1 of G with $\delta(u) \cap M_1 = \emptyset$ if and only if $u \in U$. Form a graph G' by adding to G a vertex u' and an edge $e(u)$ joining u' to u for each $u \in U$. Put $c_{e(u)} = 0$ for each $u \in U$. Then G' has an optimal matching which is perfect, namely, $M_2 = M_1 \cup \{e(u)\colon u \in U\}$. If we begin the primal algorithm with M_2 and with (y, Y) such that Y and $2y$ are integer-valued, we can obtain a corresponding optimal dual solution for which Y and $2y$ are integer-valued. We wish also to have $y \geq 0$. Choose u such that $y_u < 0$, and define r and T as in (20). We perform steps (21), (22), (23) of the primal algorithm repeatedly, until no more such steps can be carried out. We then calculate α as in (24), put $\epsilon = \min(\alpha, -y_u)$, and replace (y, Y) by (y', Y') as in (24). We continue this procedure until $y_u = 0$. We claim that there is no $v \in V(G)$ such that y_v becomes negative during the procedure. If this is not true, then v must be an element of $O(T)$. We assume that there is an edge e with $c_e = 0$ joining u to v in G. (Since M_2 has maximum weight among all matchings of G', perfect or not, adding such an edge e with weight 0 will not change the fact that M_2 is an optimal perfect matching.) But then $d_e = y_u + y_v < 0$; this implies that an augmentation could be done, obtaining a perfect matching of G' having larger weight than M_2. This cannot be true, so there cannot exist such a vertex v. It follows that we have a finite algorithm which will find an optimal dual solution (y, Y) to the perfect matching problem on G' such that $y \geq 0$. Also, by reasoning similar to that in the proof of (47), Y and $2y$ are integer-valued.

If we now apply the algorithm of the proof of (46) to obtain an integer-valued dual solution to the perfect matching problem on G', non-negativity will also be maintained; the reason is that a y_v is lowered in that algorithm only if $y_v \equiv \frac{1}{2} (\text{mod } 1)$ and in that case y_v is lowered by $\frac{1}{2}$. Now, that integer-valued, non-negative dual solution (y, Y) for G' can be restricted to G without changing the value of the dual objective function. This is because an edge $e(u)$ cannot be in $\gamma(S)$, where $S \in \mathcal{S}$ and $\mathcal{S}$ is a shrinking family of G'. Thus, since $c_{e(u)} = 0$, it must be that $y_{u'} = 0$ for each $u \in U$. The proof is complete.

We wish to point out some connections between the results of this section and work on optimum b-matching. Given a vector $b = (b_v: v \in V(G))$ of positive integers, a *b-matching* is a non-negative integer-valued vector $x = (x_j: j \in E(G))$ such that $x(\delta(v)) \leq b_v$ for $v \in V(G)$; a *perfect* b-matching satisfies each of the latter inequalities with equality. We have been dealing with that instance of b-matching in which each b_v is 1. Where Q is generalized to be $\{S \subseteq V(G): |S| \geq 3,\ b(S) \text{ odd}\}$ and q_S for $S \in Q$ becomes $\frac{1}{2}(b(S) - 1)$, the results (40), (41), (48) extend to general b-matching (see [9], [12], [18]). But while (42) and its analog for not-necessarily-perfect matchings are also special cases of results on b-matching, this is not true of our stronger results, (46) and (49). A well-known counterexample is provided by a polygon having 3 vertices, with each $b_v = 2$ and each $c_j = 1$. In this example, the only optimal dual solution have $y_v = \frac{1}{2}$ for each v.

Recently, Hoffman and Oppenheim [14] have considered adding to the constraints defining the b-matching polytope, constraints of the form $x(\gamma(S)) \leq \frac{1}{2} b(S)$, where $b(S)$ is even. These constraints are redundant to the definition of the polytope, but they allow more freedom in the choice of optimal dual solutions. It follows from the main result of [14] that, when c is integer-valued, there exists an integer-valued optimal dual solution, allowing the use of dual variables corresponding to these redundant constraints. (The reader can check that this result is not contradicted by the counterexample of the previous paragraph.) This result can also be derived directly from the b-matching analogue of (42), as has been pointed out to us by R. Giles and W. Pulleyblank. It is not difficult to show that, if at least one b_v is 1, then the "extra" dual variables can be eliminated without losing the integrality property. Thus, part of (49) can be deduced from previously known results. Subsequent work related to the subject of this section can be found in [19] and [21].

9. Some results on perfect matchings

In this section we apply the methods of the last section to study perfect matchings from a graph-theoretic, rather than optimizational, point of view. This subject has received considerable attention recently, from Lovász [17] and others.

Given a graph G and a subset S of $V(G)$, we use $G - S$ to denote $G[V(G) \setminus S]$. We say that G is *k-connected*, for k a positive integer, if $|V(G)| \geq$

$k+1$ and $G-S$ is connected for every $S \subseteq V(G)$ with $|S|<k$. We say that G is *hypomatchable* if $G-\{v\}$ has a perfect matching for every $v \in V(G)$, and that G is *bicritical* if $G-\{u, v\}$ has a perfect matching whenever $u, v \in V(G)$ and $u \neq v$. A prime example of a hypomatchable graph (in fact, by a result of [20], the only example) is a graph G having a shrinking family $\mathscr{S}$ such that $|V(G \times \mathscr{S})| = 1$. We state an old result (50) of Hall [13] and a result (51) derivable from the proof of (46). These will be applied to prove theorems of Zaks and Lovász on the number of perfect m?tchings in a k-connected graph.

(50) Theorem. *If G is a bipartite graph having bipartition $\{U, W\}$ such that G has a perfect matching and each $u \in U$ is adjacent to at least k elements of W, then G has at least $k!$ perfect matchings.*

(51) Theorem. *Let G be a graph having a perfect matching and let $u \in V(G)$. Then there exists a set $I \subseteq V(G)$ such that*

(51a) $u \in I$;

(51b) the components $C_1, C_2, \ldots, C_n$ of $G-I$ are hypomatchable;

(51c) $|I| = n$.

Proof. Take $c_j = 1$ for each $j \in E(G)$. Begin the procedure of the proof of (46) with any perfect matching M and the optimal dual solution given by $Y_S = 0$ for $S \in Q$ and $y_v = \frac{1}{2}$ for $v \in V(G)$, and let the first choice of u in the algorithm be the u of (51). At termination of the algorithm we will have an integer-valued optimal dual solution (y, Y) and a shrinking family $\mathscr{S}$ of G. Since each $y_v \equiv \frac{1}{2} \pmod 1$ initially, there is a spanning forest F of $G \times \mathscr{S}$ whose components are alternating trees $T_1, T_2, \ldots, T_m$ constructed by the algorithm. The vertices $v \in V(G)$ such that $y_v = 1$ will be precisely the elements of $\bigcup(I(T_i)\colon i \leq 1 \leq m)$; the remaining vertices $v \in V(G)$, the elements of $\bigcup(O(T_i)\colon 1 \leq i \leq m)$, will have $y_v = 0$. The sets $S \in \mathscr{S}$ which have $Y_S > 0$ will be precisely the pseudo vertices of $G \times \mathscr{S}$ which are odd vertices of some T_i; each of these will have $Y_S = 1$. No two odd vertices of the same T_i or of two different T_i can be joined by an edge in $G \times \mathscr{S}$, for such an edge e would violate $d_e \geq 0$. Thus if we choose I to be $\bigcup(I(T_i)\colon 1 \leq i \leq m)$, the components C_i of $G-I$ will be graphs $G[S]$ for S a pseudo vertex of $G \times \mathscr{S}$ which is an odd vertex of some T_i, and $G[\{v\}]$ for v a real vertex of $G \times \mathscr{S}$ which is an odd vertex of some T_i. For this choice of I, each of the requirements (51a), (51b), (51c) is easily seen to be satisfied. This completes the proof.

Theorem (51), which has been derived using the primal algorithm, bears a resemblance to an important result, called the Edmonds–Gallai theorem in [17], which was derived using the blossom algorithm in [6]. However, the Edmonds–Gallai theorem is much superior, in that it identifies *uniquely* certain structure related to the maximum cardinality matchings of G. Nevertheless, in the case in which G has a perfect matching, the Edmonds–Gallai theorem gives no additional information, while (51) does say something further.

We will use (51) to prove the following two results on the number of perfect matchings in k-connected graphs. The first theorem is due to Lovász [17], and the second to Zaks [23]. We point out that the idea of our proof of (52), namely, appealing to the result (50) for bipartite graphs, is the same as in [17]; however, we have used the primal algorithm to make the reduction, while Lovász applies a structural theory of graphs having perfect matchings, which is described in [17].

(52) Theorem. *Let G be a k-connected graph having a perfect matching. If G is not bicritical, then G has at least $k!$ perfect matchings.*

Proof. Suppose that, for some choice of u in (51), we obtain $n > 1$. For any C_i, let $N(i)$ be the set of vertices in I adjacent to at least one element of $V(C_i)$. Then $G - N(i)$ is not connected, so $|N(i)| \geq k$. It follows that the bipartite graph G', obtained from G by shrinking each $V(C_i)$ and deleting the elements of $\gamma(I)$, satisfies the hypothesis of (50). Thus G' has at least $k!$ perfect matchings. Since each C_i is hypomatchable, each such perfect matching is extendable to a perfect matching of G, so G has at least $k!$ perfect matchings. On the other hand, if $n = 1$ in (51) for every choice of u, it follows that $G - \{u\}$ is hypomatchable for each $u \in V(G)$, and thus that G is bicritical. The proof is complete.

(53) Theorem. *If G is k-connected and has a perfect matching, then G has at least $k(k-2)(k-4)\cdots$ perfect matchings.*

Proof. If G is not bicritical, the result follows from (52). Otherwise, suppose that $k \geq 3$, G is bicritical, and every $(k-2)$-connected graph has at least $m = (k-2)(k-4)\cdots$ perfect matchings. For any edge e joining vertices u and v, e is an element of at least m perfect matchings of G, since G is bicritical and $G - \{u, v\}$ is $(k-2)$-connected. Since each perfect matching contains $\frac{1}{2}|V(G)|$ edges, G has at least $(|E(G)| \cdot m)/\frac{1}{2}|V(G)|$ perfect matchings. It follows from the k-connectivity of G that $2|E(G)| \geq |V(G)| \cdot k$. Thus G has at least $m \cdot k$ perfect matchings. To complete the proof, we need only show that (53) holds for $k = 1$ and 2. The case $k = 1$ is trivial. For $k = 2$, we may assume from (52) that every edge is in a perfect matching. By 2-connectivity, a vertex must have at least 2 edges incident with it and these cannot be in the same perfect matching, so there are at least two perfect matchings. The proof is complete.

10. Computational results

In this section we report computational experience for a computer implementation of the primal algorithm and a new implementation of the blossom algorithm, and compare these results with those obtained using earlier codes. We also provide empirical evidence of the value of the post-optimality procedure introduced in Section 7.

BLOSSOM I is the Fortran code described in [11]; it solves extremely general matching problems, involving arbitrary integers b_v, arbitrary positive integer capacities (upper bounds) on variables x_j, and both directed and undirected edges.

BLOSSOM II is the PL/1 code described in [18]; it solves optimum b-matching problems, as defined at the end of Section 8.

BLOSSOM III and PRIMAL are our Fortran implementations of the blossom and primal algorithms for the problems treated in this paper. (The second author, Burton Marsh, did the computer programming.) Table 1 reports IBM 370/158 solution times for each of these codes. The problems were randomly generated simple graphs with integer weights; in each case the codes were given the same problems to solve.

The clear superiority of BLOSSOM III over PRIMAL confirms our conjecture, and can be explained as follows. The blossom algorithm will grow precisely $\frac{1}{2}|V(G)|$ trees (provided a perfect matching exists), whereas the primal algorithm can grow as many as $|V(G)|-1$ trees. (Our statistics indicate that it often grows approximately $\frac{3}{4}|V(G)|$ trees.) More importantly, because the primal algorithm maintains a perfect matching, it performs many more tree-growing, shrinking and expanding steps than does the blossom algorithm.

The superiority of BLOSSOM III over the other two blossom codes is, of course, partly attributable to the fact that it solves less general problems. We also point out a characteristic of the other two codes which has a marked influence on the computational results; namely, they are extremely sensitive to changes in the range of the weights. Lines 5 through 7 of Table 1 illustrate this sensitivity (for BLOSSOM II), and demonstrate the robustness of BLOSSOM

Table 1
Computational comparison of matching codes

Number			Weight	Average 370/158 CPU seconds			
of graphs	$\lvert V(G)\rvert$	$\lvert E(G)\rvert$	range	BI	BII	BIII	P
10	10	25	1–100	00.70	000.55	000.27	00.38
5	50	125	1–100	13.2	8.25	1.35	4.14
5	50	1225	1–100	58.15	22.87	3.75	7.49
2	100	500	1–100	70.08	35.14	5.12	18.69
2	100	2500	1–10	—[a]	23.63	7.77	23.31
2	100	2500	1–100	—	44.7	9.69	23.46
2	100	2500	1–1000	—	173.19	9.30	29.39
2	100	4950	1–100	—	84.58	19.29	43.96
1	500	5000	1–100	—	331.6	125.49	—[b]

[a] BLOSSOM I does not accept problems having $|E(G)| > 1500$.
[b] $|V(G)|^3$ implementation exceeded storage allocated.

III and PRIMAL in this regard. Also, the relatively good performance of BLOSSOM II on the 500-vertex problem is explained by the fact that the weight-range is small relative to $|E(G)|$.

Two innovations in BLOSSOM III and PRIMAL are worth mentioning here. First, unlike the other two blossom codes, which use a triply-linked representation for trees, BLOSSOM III and PRIMAL use only a single predecessor label. This simpler data structure is sufficient, at least for the less general matching problems treated here, and is, of course, much cheaper to maintain. Second, an idea of Lawler [16] has been used to achieve order of $|V(G)|^3$ implementations in both of the new codes. However, our experience indicates that these implementations are actually slower than corresponding order of $|V(G)|^2 \cdot |E(G)|$ implementations for small problems ($|V(G)| \leq 50$) and are only about 25% faster for large problems ($|V(G)| = 500$, $|E(G)| = 5000$). Moreover, the $|V(G)|^3$ implementations require considerably more storage (theoretically, order of $|V(G)|^2$, as opposed to order of $|V(G)| + |E(G)|$), and this can be a serious practical problem.

In order to test the value of the primal algorithm in post-optimality situations, we generated a new problem from a previously-solved one by generating new weights for the edges incident with a set $U \subseteq V(G)$. We then solved this new problem by the method suggested in Section 7. The time required for BLOSSOM III to solve the original problem was taken as a reasonable estimate of the time required to solve the new problem "from scratch". This time is compared with times for various sizes of U in Table 2. (The weight-range for all of these problems was 1–100. The times are in CDC 7600 CPU seconds.) These results provide strong evidence that the primal algorithm is to be preferred when $|U|$ is not too large.

Table 2
Post-optimality results

Number of graphs	$\lvert V(G)\rvert$	$\lvert E(G)\rvert$	BLOSSOM III average time	$\lvert U\rvert$	PRIMAL average time
10	30	435	0.088	1	0.017
				3	0.035
				10	0.102
				1	0.140
5	100	1650	0.637	10	0.648
				33	2.044
				1	0.291
2	300	1495	5.566	30	1.039
				100	2.707

Acknowledgment

We gratefully acknowledge the strong influence of Professor Jack Edmonds on this work, not only through his research, but also through his teaching and encouragement. We also thank Professor M.L. Balinski, who brought the paper [2] to our attention.

References

[1] M.L. Balinski, "Establishing the matching polytope", *Journal of Combinatorial Theory* B 13 (1972) 1–13.
[2] M.L. Balinski and R.E. Gomory, "A primal method for the assignment and transportation problems", *Management Science* 10 (1964) 578–593.
[3] R.S. Barr, F. Glover and D. Klingman, "The alternating basis algorithm for assignment problems", *Mathematical Programming* 13 (1977) 1–13.
[4] N. Christofides, "Worst-case analysis of a new heuristic for the traveling salesman problem", GSIA, Carnegie-Mellon University (1976).
[5] W.H. Cunningham, "A network simplex method", *Mathematical Programming* 11 (1976) 105–116.
[6] J. Edmonds, "Paths, trees, and flowers", *Canadian Journal of Mathematics* 17 (1965) 449–467.
[7] J. Edmonds, "Maximum matching and a polyhedron with (0, 1) vertices", *Journal of Research of the National Bureau of Standards* 69B (1965) 125–130.
[8] J. Edmonds, "An introduction to matching", Lecture notes, Univ. of Michigan Summer Engineering Conf. (1967).
[9] J. Edmonds and E.L. Johnson, "Matching: a well-solved class of integer linear programs", in: R.K. Guy et al., eds., *Combinatorial structures and their applications* (Gordon and Breach, New York, 1970).
[10] J. Edmonds and E.L. Johnson, "Matching, Euler tours, and the Chinese postman", *Mathematical Programming* 5 (1973) 88–124.
[11] J. Edmonds, E.L. Johnson and S.C. Lockhart, "Blossom I, A computer code for the matching problem", to appear.
[12] J. Edmonds and W.R. Pulleyblank, *Optimum matching* (Johns Hopkins Press) to appear.
[13] M. Hall, "Distinct representatives of subsets", *Bulletin of the American Mathematical Society* 54 (1948) 922–926.
[14] A.J. Hoffman and R. Oppenheim, "Local unimodularity in the matching polytope", *Annals of Discrete Mathematics* 2 (1978) 201–209.
[15] H.W. Kuhn, "The Hungarian method for the assignment problem", *Naval Research Logistics Quarterly* 2 (1955) 83–97.
[16] E.L. Lawler, *Combinatorial optimization* (Holt-Rinehart-Winston, New York, 1976).
[17] L. Lovász, "On the structure of factorizable graphs", *Acta Mathematica Academiae Scientiarum Hungaricae* 23 (1972) 179–195.
[18] W.R. Pulleyblank, Faces of matching polyhedra, Thesis, University of Waterloo (1973).
[19] W.R. Pulleyblank, "Dual integrality in b-matching problems", CORE Discussion Paper, Louvain (1977).
[20] W.R. Pulleyblank and J. Edmonds, "Facets of 1-matching polyhedra", in: C. Berge and D. Ray-Chaudhuri, eds., *Hypergraph seminar*, Lecture Notes in Mathematics, No. 411 (Springer, Berlin, 1974).
[21] A. Schrijver and P.D. Seymour, "A proof of total dual integrality of matching polyhedra", Mathematical Centre, Amsterdam (1977).
[22] P. Seymour, "On the 1-factors of regular graphs", to appear.
[23] J. Zaks, "On the 1-factors of n-connected graphs", in: R.K. Guy et al., eds., Combinatorial structures and their applications (Gordon and Breach, New York, 1970).