M.O. Ball et al., Eds., *Handbooks in OR & MS, Vol. 7*

Chapter 3

Matching

A.M.H. Gerards
CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

1. Introduction

Matching is pairing: dividing a collection of objects into pairs. Typically the objective is to maximize total profit (or minimize cost), where the profit of each possible pair is known in advance.

For a more formal definition, let G be an undirected graph with node set V and edge set E. A subset M of E such that no two edges in M are incident to a common node is a *matching*. If M has exactly one edge incident to each node $v \in V$, we call M a *perfect matching*. The *maximum weight matching problem* with respect to the *weights* w on the edges is:

$$\text{Find a matching } M \text{ with total weight } \sum_{e \in M} w_e \text{ as large as possible.} \tag{1}$$

The *minimum weight perfect matching problem* is:

$$\text{Find a perfect matching } M \text{ with total weight } \sum_{e \in M} w_e \text{ as small as possible.} \tag{2}$$

A matching problem is defined by two parameters: the graph G and the weights w. We classify matching problems by these different parameters:

The weights – cardinality or weighted matching. Finding a maximum cardinality matching, i.e. considering all edges to have weight one, is easier than dealing with more general weights. Moreover, an algorithm for finding a maximum cardinality matching can be, and in our presentation is, used as a subroutine for solving the general weighted problem. Therefore, we discuss the cardinality case first, in Section 2, and study general weights later, in Section 6.

The graph – bipartite or non-bipartite matching. Matching problems are significantly easier in bipartite graphs. So, we discuss several topics in bipartite matching before venturing into the complications of the non-bipartite case.

Matching theory is one of the cornerstones of mathematical programming. Yet, matchings are not as ubiquitous in practice as network flow problems (for applications of network flows, see Ahuja, Magnanti & Orlin [1989]) and, when they do arise in practice, it is most often in bipartite graphs where they can be

modeled as network flow problems anyway (see Section 3.1). So, to what does 'Matching' owe its prominence?

First, there is the position of matching problems between the 'easier' problems like network flows, and the hard (NP-hard) problems like general integer linear programming. This position has been pointed out by Edmonds & Johnson [1970]. It is probably best expressed by the qualification: "*Optimum matching problems ... constitute the only class of genuine integer programs for which a good solution is known*" [Cunningham & Marsh, 1978]. In a footnote, Cunningham and Marsh clarify this statement with: "*... 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).*"

Second, and perhaps more intrinsic to the importance of matching, there is the intricate structure of matching theory. Just a glimpse in the excellent book *Matching Theory* by Lovász & Plummer [1986] should convince the reader of this. For instance, there are the structural descriptions of the class of all maximum cardinality matchings in a graph. In this chapter we see how one of these, the Edmonds–Gallai structure (see Section 4.1), helps in understanding an algorithm for finding a minimum weight perfect matching (see Section 6.2). Third, there is the 'self-refining' property of matching theory: it contains a wide class of its generalizations as special cases (see Sections 3 and 7).

1.1. Examples of matching problems

We begin with four examples of matching problems. A classic example is:

The assignment problem. Suppose n tasks are to be carried out and each must be assigned to a single person. We have a staff of n people available and each person can be assigned only one task. Moreover, we know for each person p and task t a number $w_{p,t}$ quantifying the productivity of p when carrying out t. Now, the question is: How do we assign the tasks to the people, i.e. make task-person pairs, so that the total productivity is as large as possible?

Clearly, this is a bipartite matching problem. An example of a non-bipartite matching problem is:

The oil well drilling problem [Devine, 1973]. Suppose we are given the locations of oil deposits. We want to exploit all the deposits at minimum drilling cost. It is technically feasible to access two deposits with one well: drill a hole to the first deposit, and then continue drilling from that same hole, possibly at a different angle, to the second deposit. The oil is then brought up from both deposits via concentric pipes. We know the savings possible from combined drilling operations for each pair of deposits. The question is: How do we combine the drilling operations so that the savings are as large as possible?

In both of these cases, the matching character of the problem is immediately obvious from the description. A more disguised matching problem is:

Plotting street maps [Iri & Taguchi, 1980; Iri, Murota & Matsui, 1983]. A pen-plotter has to draw a street map of a city. Since the total length of the lines to be drawn is fixed, this amounts to minimizing the total distance of the 'non-drawing' moves, or *shifts*, the pen makes to change position. A natural assumption is that the pen starts from some prescribed point and returns to it when the drawing is finished. For simplicity, we assume that this common starting and ending point is a point of the map to be drawn. Moreover, we assume that the graph of the map (edges are streets, nodes are intersections) is connected, i.e., that you can travel between any two intersections in the city along streets. The question is: In what order should the edges be drawn to minimize the total drawing time?

An old and famous theorem of Euler's [1736] asserts that if all the nodes in a connected graph have even degree, one can draw the graph without making any shifts. However, when there are nodes with odd degree, shifts are unavoidable. In this case, the problem amounts to adding edges to the graph, which will become the shifts, so that every node in the resulting graph has even degree. Distances in the city are Euclidean. So we may assume, by the triangle inequality, that no two shifts have a common endpoint. Hence, finding the fastest way to draw the map amounts to pairing up the nodes with odd degree so that the total length of the line segments connecting the two nodes in each pair is as short as possible. This is a minimum weight perfect matching problem.

The following application is even less obvious. Again, we solve this problem as a matching problem, but now the resulting matching must be modified in a non-trivial way to obtain a solution to the original problem.

Scheduling [Fujii, Kasami & Ninomiya, 1969]. Suppose, you and your partner want to restore this lovely old house you just bought. You have divided the whole project into a number of one-week jobs that must be carried out in accordance with certain precedence relations. For example, it is difficult to paper a new wall before putting it up. As the project puts already enough pressure on the relationship, you have agreed not to work together on any job. You both have about the same skills, however, so either of you can do each job equally well. The question is: How do you allocate the jobs between you so you can finish the project as early as possible?

We have a list of one-week jobs $J_1, \ldots, J_k$, together with a partial order $\prec$ on them. The relation $J_i \prec J_j$ means that J_i should be carried out before J_j and two jobs are *incomparable* if neither must precede the other. Minimizing the project duration amounts to scheduling the jobs, in accordance with the precedence constraints, so that in as many weeks as possible you and your partner are both assigned jobs. Clearly, if you and your partner are both working in a given week, your jobs must be incomparable. Therefore, as a first attempt at finding an optimal schedule, we look for a largest set of disjoint, incomparable pairs: a maximum cardinality matching problem.

If a largest matching $\mathcal{P}$ consists of ℓ disjoint, incomparable pairs, then we know that the whole project will take at least $k - \ell$ weeks. In fact, we can complete the project within that period. The idea is to schedule the jobs from each pair in $\mathcal{P}$ in a single week. Each of the remaining jobs, called the *singleton jobs*, is assigned to a week by itself. We might, however, have to change $\mathcal{P}$ to make this possible.

The proof that we can complete the project in $k - \ell$ weeks is by induction on k, the number of jobs. Let $\mathcal{E}$ denote the jobs that are minimal with respect to $\prec$. If there is a pair, say $(J_1, J_2) \in \mathcal{P}$ with both J_1 and J_2 in $\mathcal{E}$, we schedule these two jobs in the first week. $\mathcal{P} \setminus (J_1, J_2)$ is still a largest collection of incomparable pairs among the remaining jobs. Hence the induction hypothesis assures us we can schedule the jobs in $k - \ell$ weeks. A similar argument applies if $\mathcal{E}$ contains a singleton job.

Now suppose $\mathcal{E}$ contains neither a singleton job nor a pair in $\mathcal{P}$. Among all pairs $(J_1, J_2) \in \mathcal{P}$ with $J_1 \in \mathcal{E}$ choose one with J_2 minimal with respect to $\prec$. As $J_2 \notin \mathcal{E}$ there exists a $J_3 \in \mathcal{E}$ with $J_3 \prec J_2$. Moreover, as J_3 is not a singleton job, there exists a job J_4 with $(J_3, J_4) \in \mathcal{P}$. As $J_3 \prec J_2$, J_2 does not precede J_4. Moreover, by the choice of J_2, J_4 does not precede J_2. So jobs J_2 and J_4 are incomparable. Hence, $(\mathcal{P} \setminus \{(J_1, J_2), (J_3, J_4)\}) \cup \{(J_1, J_3), (J_2, J_4)\}$ is also a largest collection of disjoint, incomparable pairs. As this collection does have pair in $\mathcal{E}$, we can proceed as before, schedule that pair in the first week, and by the induction hypothesis the other jobs in the remaining $k - \ell - 1$ weeks.

For other applications of matching see Ball, Bodin & Dial [1983] and Murota [1993].

1.2. Generalizations of the matching problem

There are two obvious directions in which the matching problem can be generalized.

Degree-constrained optimization. Matchings are subgraphs in which each edge appears at most once and each node has degree at most one. This suggests extensions allowing more general degree constraints and allowing an edge to appear more than once in a feasible solution. These generalizations lead to so-called 'b-factors' and 'b-matchings', and – if we limit the number of times an edge can appear – to 'capacitated' b-matchings. We can also impose lower bounds on the degrees of nodes and on the number of times edges appear. Thus we can extend the matching problem to a quite general degree-constrained (multi-)graph optimization problem. Surprisingly, many results for ordinary matchings extend to these generalizations. This phenomenon, which we explain in Section 7, is part of the self-refining nature of matching theory previously mentioned.

This self-refining nature persists even when we impose parity conditions on the degrees of nodes, e.g. demanding that feasible solutions have an odd number of edges incident to particular nodes (see Section 7.4).

Set-packing problems. As matching is finding disjoint pairs, a natural generalization of the matching problem is trying to find, in some sense optimal, collections of disjoint triples, quadruples or otherwise structured sets. This gives rise to the very general *set-packing* problem, which includes many combinatorial optimization problems as special cases. Unfortunately the set-packing problem is too general, it is NP-hard (even if we only consider packing triples [Karp, 1972]). Except for a few lines on stable sets in perfect graphs (in Section 3.2), we do not study them here but confine ourselves to matchings.

1.3. Other sources on matching

The number of publications concerning matching problems is enormous, the references in this chapter constitute only a very limited part of them. There are many good books on matching. We mentioned already *Matching Theory* by Lovász & Plummer [1986], which really is the most complete source on matchings available at this moment. Other highly recommendable books that deal (partly) with matchings are: *Graphs* by Berge [1985], *Combinatorial Optimization: Networks and Matroids* by Lawler [1976], *Graphs and Algorithms* by Gondran & Minoux [1984] and *Programming in Networks and Graphs* by Derigs [1988a]. Sources on general integer programming (including matching) are: *Theory of Linear and Integer Programming* by Schrijver [1986] and *Integer and Combinatorial Optimization* by Nemhauser & Wolsey [1988].

Excellent introductions to matchings are: Schrijver's [1983a] survey paper on min–max relations in combinatorial optimization, with special emphasis on the self-refining properties of matching and the paper by Pulleyblank [1995], with more emphasis on structural results then one will find in this chapter. A very nice historical overview of matching theory – from its birth to today's state of the art – is the paper by Plummer [1992]. Recently, the same author gave an overview on how 'hard' or 'easy' various matching and vertex-packing problems are [Plummer, 1993].

As mentioned, bipartite matchings are really network flows. Network flows are also discussed in most of the just mentioned publications. For extensive treatments of network flows we refer to the surveys by Ahuja, Magnanti & Orlin [1989], by Goldberg, Tardos & Tarjan [1990] and by Helgason & Kennington [1995, this volume].

1.4. Outline

The first part of this chapter considers algorithms for finding maximum cardinality as well as maximum weight matchings (Sections 2 and 6). We start with the Hungarian method for finding a maximum cardinality matching in a bipartite graph (Section 2.1). We then extend the method in two directions: to Edmonds' blossom algorithm for finding a maximum cardinality matching in a general graph (Section 2.2) and to the Hungarian method for finding a maximum weight matching in a bipartite graph (Section 6.1). Finally, the ideas of these two methods are combined in Edmonds' algorithm for the weighted matching problem in gen-

eral graphs (Section 6.2). From the insight provided by the Hungarian method for maximum cardinality matching in bipartite graphs the classical theorems of Fröbenius and König on bipartite matchings easily follow (Section 3). Some of the self-refining properties of matchings can be found in that section. Edmonds' blossom algorithm for cardinality matching yields the theorems of Tutte and Berge on matchings in general, non-bipartite, graphs and the Edmonds–Gallai structure theorem (Section 4). This structure theorem facilitates the description and analysis of Edmonds' blossom algorithm for weighted matching. Because algorithms for the weighted matching problem are closely related to the formulation of this problem as a linear program, we discuss the matching polytope in Section 5. In that section we also briefly mention stable matchings (Section 5.2). Together these sections contain the basic algorithmic and structural aspects of matchings.

The second part of this chapter consists of four sections. In Section 7 we consider general degree constraints and discuss some of the self-refining aspects of matching. In Section 8 we present other algorithms for matching problems, including randomized algorithms for maximum matching and for counting matchings. In Section 9, we discuss applications of matchings to other combinatorial optimization problems like the traveling salesman problem. This chapter concludes with a (short) section on the computer implementation of matching algorithms and on heuristics for matching problems.

We conclude this section with some notation and conventions.

1.5. Notation and conventions

1.5.1. Graphs

Typically we denote the edge set of an undirected graph G by E and its node set by V. When ambiguity might arise, we write $E(G)$ and $V(G)$. We write $uv \in E$ to mean that uv is an edge with endpoints u and v. In case of parallel edges this might seem a bit ambiguous, but generally it is not. Parallel edges are not particularly relevant for matching problems and we can always assume that there are none. Yet, parallel edges hardly complicate the problem. In fact, in solving matching problems we construct graphs with parallel edges. So we do not explicitly forbid them.

We assume the reader is familiar with the basic notions of graph theory [see Bondy & Murty, 1976] and only discuss those most important for this chapter. For each subset $U \subseteq V(G)$, we define: $\delta(U) := \{uv \in E(G) \mid u \in U, v \notin U\}$ and $\langle U \rangle := \{uv \in E(G) \mid u \in U, v \in U\}$. We let $G|U$ denote the subgraph of G *induced* by U, i.e., $V(G|U) = U$, $E(G|U) = \langle U \rangle$, and we let $G \setminus U := G|(V(G) \setminus U)$. For each node $u \in V(G)$, $G \setminus u := G \setminus \{u\}$, $\delta(u) := \delta(\{u\})$ and $\deg(u) := |\delta(u)|$, the *degree* of u. For each edge $e \in E(G)$, $G \setminus e$ is the graph with node set $V(G)$ and edge set $E(G) \setminus \{e\}$. We use $G \cup e$ to denote the graph obtained by adding the new edge $e \notin E(G)$ to G, i.e., $V(G \cup e) = V(G)$ and $E(G \cup e) = E(G) \cup \{e\}$. The notions of connected graphs, components, paths, trees and forests are so standard that we omit their definitions here. Circuits and Eulerian graphs are standard notions too, but there is a rather wide-spread

babel as far as the terminology is concerned. We refer to a *circuit* (of length k) as a graph C with k distinct nodes $V(C) := \{v_1, \ldots, v_k\}$ and the k distinct edges $E(C) := \{v_1v_2, v_2v_3, \ldots, v_{k-1}v_k, v_kv_1\}$. A *cycle* is a graph with all degrees even. So a cycle is an edge-disjoint union of circuits (in other writings one might find 'cycle' where we use 'circuit'). A connected cycle is called an *Eulerian* graph. We often identify a circuit C with its edge set $E(C)$, and so write $e \in C$, meaning $e \in E(C)$. Similarly, we identify other subgraphs like paths or trees with their edge sets.

We denote a bipartite graph by $G = (V_1 \cup V_2, E)$ where V_1 and V_2 are the color classes (so each edge has one endpoint in V_1 and one in V_2).

Given a directed graph D, $V(D)$ denotes the node set, $A(D)$ denotes the arc set, and $\overrightarrow{uv}$ denotes an arc from u to v. For each subset $U \subseteq V$, $\delta^-(U) := \{\overrightarrow{uv} \in A(G) \mid u \notin U, v \in U\}$ and $\delta^+(U) := \{\overrightarrow{uv} \in A(G) \mid u \in U, v \notin U\}$. Again, for each node $u \in V(D)$, we abbreviate $\delta^-(\{u\})$ as $\delta^-(u)$ and $\delta^+(\{u\})$ as $\delta^+(u)$.

1.5.2. Numbers, vectors, polytopes and polyhedra

For $\alpha \in \mathbb{R}$, $\lfloor \alpha \rfloor$ denotes the largest integer not greater than α. Similarly $\lceil \alpha \rceil$ denotes the smallest integer not smaller than α.

Given a set R and a finite set S, R^S denotes the collection of vectors indexed over S with components in R. So, for example, $\mathbb{R}^S$ is the collection of real vectors and $\mathbb{Z}^S$ is the collection of integral vectors with components indexed by S. We use $\mathbb{R}_+$ to denote the set of non-negative reals and $\mathbb{Z}_+$ to denote the set of non-negative integers. For each subset $T \subseteq S$, $\chi^T \in \{0, 1\}^S$ denotes the *characteristic vector* of T as a subset of S, i.e., $(\chi^T)_t = 1$ if $t \in T$ and $(\chi^T)_t = 0$ if $t \notin T$. Given $x \in \mathbb{R}^S$ and $T \subseteq S$, we frequently use $x(T)$ to denote $\sum_{t \in T} x_t$.

The *node-edge incidence matrix* $N = (N_{u,e})$ of an undirected graph G is the $V(G) \times E(G)$ matrix with $N_{u,e} = 1$ if u is an endpoint of edge e and $N_{u,e} = 0$ otherwise.

Let $x^1, \ldots, x^k \in \mathbb{R}^S$. Any vector of the form $\sum_{i=1}^k \lambda_i x^i$ with $\sum_{i=1}^k \lambda_i = 1$ and $\lambda_i \geq 0$ for each $i \in \{1, \ldots, k\}$, is called a *convex combination* of $x^1, \ldots, x^k$. The *convex hull* of a set X is the collection of all convex combinations of finite subsets of X. A *polytope* is the convex hull of a finite set. A *polyhedron* is the solution set of a finite system of linear inequalities, i.e., a set of the form $\{x \in \mathbb{R}^n \mid Ax \leq b\}$ for some matrix A and vector b.

1.5.3. Algorithms

One measure of the running time of an algorithm is the number of arithmetic steps it requires. Here, an arithmetic step is the addition, multiplication, division or comparison of two numbers. We report the running time of an algorithm by giving its asymptotic behavior as a function of the size of the input. So, we say for instance that the running time is $O(n)$ meaning that there exists a constant γ so that given input of size n the algorithm takes no more than γn steps before it produces the output. An algorithm is *polynomial* if its running time is $O(n^p)$ for some $p \in \mathbb{Z}_+$.

The input for matching problems is in the form of graphs and rational numbers. We assume that the graph G is represented by its *adjacency lists*, i.e., for each

node v we have a list of the nodes adjacent to v. Thus, the size of the input of a graph is proportional to $|V(G)| + |E(G)|$. (Note that every edge is represented twice in this way.)

Because we measure the running time of algorithms with respect to the number of arithmetic steps, we can consider the input size of a rational number to be 1. However, when implementing the algorithms on a computer, rationals take more space to encode. A rational number p/q, where p and q are integers, can be represented with $\log(|p| + 1) + \log(|q| + 1)$ binary bits. On the other hand, an arithmetic operation on two rationals can be carried out in a number of binary operations that is polynomial in the number of binary bits required to encode the two rationals. So, if the number of arithmetic operations an algorithm requires is polynomial in the number of input rationals, the number of binary operations it requires will be polynomial in the number of binary digits needed to encode those rationals; that is, provided the numbers that are calculated in the process do not become too big! Though not true in general, in this chapter the numbers computed do not become too large and so, it is enough to argue that the number of arithmetic operations an algorithm requires is polynomial in the size of the graph considered.

2. Finding a matching of maximum cardinality

The maximum cardinality of a matching in an undirected graph $G = (V, E)$ is denoted by $\nu(G)$. The main question of this section is: How do we find – in polynomial time – a *maximum matching*, that is, a matching of maximum cardinality? We consider this problem separately from the more general weighted problem because it is easier and it contains the main ideas and notions of the weighted problem: 'alternating paths' and 'shrinking'.

2.1. Alternating paths and forests

A path P in a graph $G = (V, E)$ is said to be *alternating* with respect to a matching M, or *M-alternating*, if the edges of P are alternately in M and not in

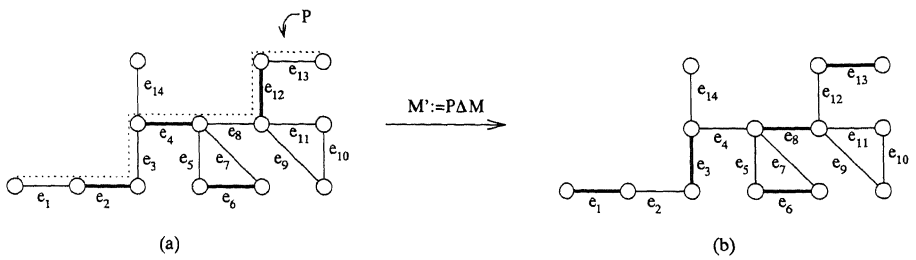

Fig. 1. The bold edges are in the matchings M, respectively M'

M (see Figure 1a; for instance, the paths $\{e_4, e_7, e_6\}$ and $\{e_1, e_2, e_3, e_4, e_5, e_6\}$ are M-alternating). So, every node in an M-alternating path P, except possibly its end nodes, is incident to an edge in $M \cap E(P)$.

If an edge uv is in a matching M, we say that the node u is *matched by* M and that the two nodes u and v are *matched*. We also write u_M to denote v. Nodes not matched by M are called *exposed* and we denote the set of exposed nodes by $\exp(M)$. We define the *deficiency* of G as $\operatorname{def}(G) := |V(G)| - 2\nu(G)$. So the deficiency is minimum cardinality of $\exp(M)$. An alternating path P is *augmenting*, or more precisely *M-augmenting*, if both its end nodes are exposed (see Figure 1a; the dotted line indicates an M-augmenting path). Augmenting paths obviously yield larger matchings (we refer to the operation described in (3) and illustrated in Figure 1, as AUGMENT):

> *If P is an augmenting path with respect to a matching M, then the symmetric difference $M' := P \triangle M$ is a matching too. Moreover,* $|M'| = |M| + 1$. (3)

In fact, the converse is also true: if there is no augmenting path, there is no larger matching.

Theorem 1 [Berge, 1957; Norman & Rabin, 1959]. *A matching M in a graph G is a maximum matching if and only if there is no M-augmenting path in G.*

Proof. Let M' be a matching in G with $|M'| > |M|$. The graph consisting of the edges in $M' \triangle M$ has maximum degree 2. Hence, each of its components is either a path or a circuit in which the edges are alternately in M' and in M. Clearly one of these components must contain more edges from M' than from M and that component must be an augmenting path with respect to M. The converse is (3). □

Mulder [1992] pointed out that this result was in fact already known by Julius Petersen [1891], probably the first to study matchings in graphs.

So, searching for maximum matchings amounts to searching for augmenting paths. We search for augmenting paths by 'growing alternating forests'.

Let M be a matching in a graph $G = (V, E)$. A tree T in G is called *alternating* if the following hold (see Figure 2):

- T contains exactly one exposed node, denoted by r_T,

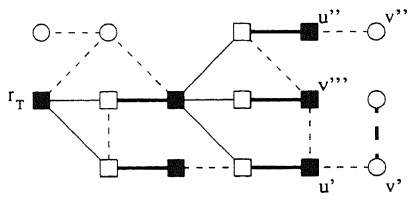

Fig. 2. The solid edges form an alternating tree. The bold edges, dashed or not, are in the matching. Open square nodes are odd; filled square nodes are even.

- for each node $v \in V(T)$, the path from r_T to v in T is alternating, and
- for each node v of degree one in T, other than r_T, the matching-edge vv_M is in T.

An *alternating forest* is a node-disjoint union of alternating trees such that each exposed node is in one of the trees. So, the forest consisting only of the exposed nodes without any edges is alternating. For each node v in an alternating forest F, F_v denotes the alternating tree in F containing v and $r_{v,F}$ denotes the unique exposed node in F_v. We call a node v in F *even* (*odd*) if the unique path in F from $r_{v,F}$ to v contains an even (odd) number of edges. We denote the set of even nodes of an alternating forest F by even(F) and the set of odd nodes by odd(F).

The following procedure uses alternating forests to search for augmenting paths.

GROW: Either (1) or (2) below applies:
(1) If there is a node $u \in \text{even}(F)$ adjacent to a node $v \notin \text{odd}(F)$, then exactly one of the following occurs:
- $v \notin \text{even}(F)$. In this case, we extend F to a larger alternating forest by adding the edges uv and vv_M (v is matched). We refer to this as 'GROWING the alternating forest'. (See Figure 2; $u = u'$, $v = v'$.)
- $v \in \text{even}(F)$ **and** $F_u \neq F_v$. In this case, we have FOUND an AUGMENTING PATH, namely the union of: the path in F_u from $r_{u,F}$ to u, the edge uv, and the path in F_v from v to $r_{v,F}$. (See Figure 2; $u = u''$, $v = v''$.)
- $v \in \text{even}(F)$ **and** $F_u = F_v$. In this case, the procedure HALTS. (See Figure 2; $u = u'$, $v = v'''$.)

(2) If no node $u \in \text{even}(F)$ is adjacent to a node $v \notin \text{odd}(F)$, the procedure TERMINATES. In this case, the forest F is called *Hungarian*.

Lemma 2. *Let M be a matching in a graph $G = (V, E)$ and let F be an alternating forest with respect to M. If F is Hungarian, then M is a maximum matching.*

Proof. Since F is Hungarian, nodes in even(F) are adjacent only to nodes in odd(F). So, each matching in G has at least $|\text{even}(F)| - |\text{odd}(F)|$ exposed nodes. On the other hand, from the definition of alternating forest it follows that $|\exp(M)| = |\text{even}(F)| - |\text{odd}(F)|$. So M is a maximum matching. □

Lemma 2 implies that GROW either finds an augmenting path, terminates with a maximum matching, or HALTS. When G is bipartite, each even node v is in the same color class of G as $r_{v,F}$. So, no two adjacent even nodes can be in the same component of F and GROW cannot HALT. Thus, we can find a maximum matching in a bipartite graph by iteratively applying GROW and AUGMENT. This algorithm has been introduced by Kuhn [1955], in the context of matchings in bipartite graphs, and Hall [1956], in the context of 'systems of distinct representatives' (see Section 3). Kuhn called it the *Hungarian method* in recognition of König and Egerváry's contributions to the theory of matchings. The Hungarian method is easy to implement.

Theorem 3. *The Hungarian method finds a matching of maximum cardinality in a bipartite graph in* $O(|E| \min(|V_1|, |V_2|))$ *time.*

Proof. It takes $O(|E|)$ time for GROW to find an augmenting path or construct a Hungarian alternating forest. Each augmentation takes $O(|E|)$ time as well. Since we AUGMENT $\nu(G) \leq \min(|V_1|, |V_2|)$ times, the theorem follows. (Note that we apply GROW $\nu(G) + 1$ times.) □

Hopcroft & Karp [1971, 1973] improved on this running time by searching for a collection of disjoint, shortest augmenting paths and then augmenting along all the paths simultaneously.

Given a matching M, we define $\ell(M)$ to be the number of edges in a shortest M-augmenting path. A collection of node-disjoint, shortest augmenting paths $P_1, \ldots, P_t$ is called *maximal* if there is no shortest augmenting path in G node-disjoint from each of the paths $P_1, \ldots, P_t$. Hopcroft and Karp's algorithm is based on the following two observations.

> *We can find a maximal collection of node-disjoint shortest augmenting paths in* $O(|E|)$ *time.* (4)

Indeed, breadth-first search starting from $\exp(M) \cap V_1$ accomplishes this.

> *If* $P_1, \ldots, P_t$ *is a maximal collection of shortest* M*-augmenting paths, then* $\ell(M \triangle P_1 \triangle \cdots \triangle P_t) > \ell(M)$. (5)

To see this, let Q be an augmenting path with respect to $M' := M \triangle P_1 \triangle \cdots \triangle P_t$. Now, observe that in proving Theorem 1 we actually proved:

> *Let* M_1 *and* M_2 *be two matchings with* $k := |M_2| - |M_1| > 0$. *Then there exists a collection of* k *mutually node-disjoint* M_1*-augmenting paths in* $M_1 \triangle M_2$. (6)

Applying (6) to M and $M' \triangle Q$, we get $t + 1$ node-disjoint M-augmenting paths $Q_1, \ldots, Q_{t+1}$ in $M \triangle M' \triangle Q$. Now one easily verifies:

$$\begin{aligned} \ell(M)(t+1) \leq |Q_1| + \cdots + |Q_{t+1}| &\leq |M \triangle M' \triangle Q| = \\ &= |P_1 \triangle \cdots \triangle P_t \triangle Q| \\ &= |(P_1 \cup \cdots \cup P_t) \triangle Q| = |P_1 \cup \cdots \cup P_t| + |Q| - \\ &\quad - 2|(P_1 \cup \cdots \cup P_t) \cap Q| \\ &= \ell(M)t + |Q| - 2|(P_1 \cup \cdots \cup P_t) \cap Q|. \end{aligned} \tag{7}$$

Hence

$$|Q| \geq \ell(M) + 2|(P_1 \cup \cdots \cup P_t) \cap Q|. \tag{8}$$

So $|Q| \geq \ell(M)$. Suppose $|Q| = \ell(M)$. Then Q has no edge in common with any of $P_1, \ldots, P_t$ Hence, since Q is augmenting with respect to $M \triangle P_1 \triangle \cdots \triangle P_t$, it must also have no node in common with any of $P_1, \ldots, P_t$. This contradicts the

maximality of the collection $P_1, \ldots, P_t$. So we conclude that $|Q| > \ell(M)$, which proves (5).

This is the algorithm of Hopcroft and Karp. In each *phase* we are given an (initially empty) matching M. We find a maximal collection of node-disjoint shortest M-augmenting paths $P_1, \ldots, P_t$ and augment along all of them to obtain the larger matching $M' := M \triangle P_1 \triangle \cdots \triangle P_t$. We iteratively repeat this procedure using as input to each successive phase the matching M' constructed in the previous phase.

The algorithm of Hopcroft and Karp finds a maximum cardinality matching in a bipartite graph in $O(|E|\sqrt{|V|})$ time. (9)

Since each phase can be carried out in $O(|E|)$ time, it suffices to prove that we need only $O(\sqrt{|V|})$ phases. Let M_1 be the matching found after $\sqrt{|V|}$ phases. By (5), $\ell(M_1) \geq \sqrt{|V|}$. So there can be at most $|V|/\sqrt{|V|} = \sqrt{|V|}$ mutually edge-disjoint M_1-augmenting paths. Applying (6) to M_1 and some maximum matching M_2, we see that $|M_1| \geq \nu(G) - \sqrt{|V|}$. Hence, after at most $\sqrt{|V|}$ further phases we obtain a maximum cardinality matching.

Improvements on (9) are the $O(|V|^{1.5}\sqrt{|E|/\log|V|})$ algorithm by Alt, Blum, Mehlhorn & Paul [1991], which is faster for dense graphs, and the $O(|E|\sqrt{|V|}\log_{|V|}(|V|^2/E))$ algorithm by Feder & Motwani [1991].

In Section 3.1 we show how to find a maximum cardinality matching by solving a max-flow problem. In fact, Even & Tarjan [1975] observed that we can interpret Hopcroft and Karp's algorithm as Dinic's max-flow algorithm [Dinic, 1970] applied to matchings in bipartite graphs. Recently, Balinski & Gonzalez [1991] developed an $O(|E||V|)$ algorithm for finding maximum matchings in bipartite graphs that is not based on augmenting path methods.

2.2. *Non-bipartite graphs – shrinking blossoms*

In non-bipartite graphs the procedure GROW may HALT even when there are augmenting paths. Indeed, consider the example in Figure 3. Nodes u and v are even, adjacent and belong to the same alternating tree. Clearly, we cannot grow the tree any further. On the other hand, there is an augmenting path (in this case it is unique) and it contains edge uv. So, we must modify our procedure for finding augmenting paths.

2.2.1. *Alternating circuits and blossoms*

A circuit C is said to be *alternating* with respect to a matching M if $M \cap E(C)$ is a maximum matching in C. So, when C is an alternating odd circuit with respect to a matching M, exactly one node in C is exposed with respect to $M \cap E(C)$. We call this node the *tip* of the alternating odd circuit C. If the tip t of an alternating odd circuit C is connected to an exposed node by an even alternating path P with $V(P) \cap V(C) = \{t\}$, then C is called a *blossom* and P is called a *stem* of C.

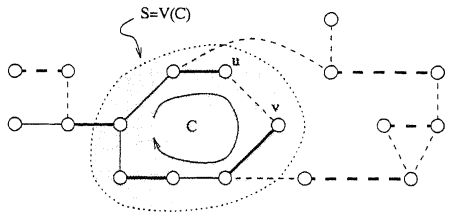

Fig. 3. Solid edges are in the alternating forest; bold edges, dashed or not, are in the matching. The shaded region indicates a blossom.

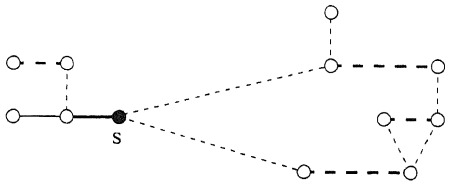

Fig. 4.

When the procedure GROW HALTS, *G contains a blossom.* (10)

Indeed, suppose we have two adjacent even nodes u and v in an alternating forest F, both belonging to the same alternating tree T of F (see Figure 3). Consider the paths P_u from r_T to u and P_v from r_T to v in F. Then $E(P_u) \triangle E(P_v)$ together with uv forms a blossom and the intersection of P_u and P_v is one of its stems.

Having detected a blossom C, we 'shrink' it. That is, we apply the procedure SHRINK to $V(C)$. Figure 4 illustrates the effect of shrinking $V(C)$ in Figure 3.

SHRINK: The graph $G \times S$ obtained from G by *shrinking* $S \subseteq V$ is constructed as follows. Remove S from V and add the new node s, called a *pseudo-node*. Remove $\langle S \rangle$ from E and replace each edge uv with one endpoint, v, in S with an edge us. We denote by $M \times S$ the edges of M in $G \times S$, i.e., $M \times S = (M \setminus \langle S \rangle) \cup \{us \mid uv \in M \cap \delta(S)$ and $v \in S\}$. Similarly, $F \times S$ denotes the edges of F in $G \times S$. If no confusion is likely, we write M and F in place of the more cumbersome $M \times S$ and $F \times S$.

When we apply SHRINK to a blossom C with node set S, $M \times S$ is a matching and $F \times S$ is an $M \times S$-alternating forest in $G \times S$. In fact, we can continue our search for an augmenting path in the shrunken graph.

Each augmenting path Q in $G \times S$ can be extended to an augmenting path Q' in G. (11)

Indeed, if $s \notin V(Q)$ then take $Q' = Q$. Otherwise, there is a unique even path P in C with the tip as one of the endpoints, such that adding P to Q yields a path Q' in G. It is easy to see that Q' is an augmenting path in G.

So, finding an augmenting path in $G \times S$, amounts to finding one in G. We can EXPAND the blossom to extend the alternating path Q in $G \times S$ to an alternating path Q' in G, and augment the matching in G. Therefor, when GROW HALTS we SHRINK.

The next theorem shows that alternately applying GROW and SHRINK finds an augmenting path, if one exists.

Theorem 4. *Let S be a blossom with respect to the matching M in the graph G. Then M is a maximum cardinality matching in G if and only if $M \times S$ is a maximum cardinality matching in $G \times S$.*

Proof. By (11), M is a maximum cardinality matching only if $M \times S$ is. To prove the converse, we assume that M is not maximum in G.

We may assume that the tip t of S and the pseudo-node s corresponding to S are exposed nodes. Indeed, if this is not the case, we simply take the stem P of S and replace M and $M \times S$ with the matchings $M \triangle P$ and $(M \times S) \triangle P$ having the same cardinalities.

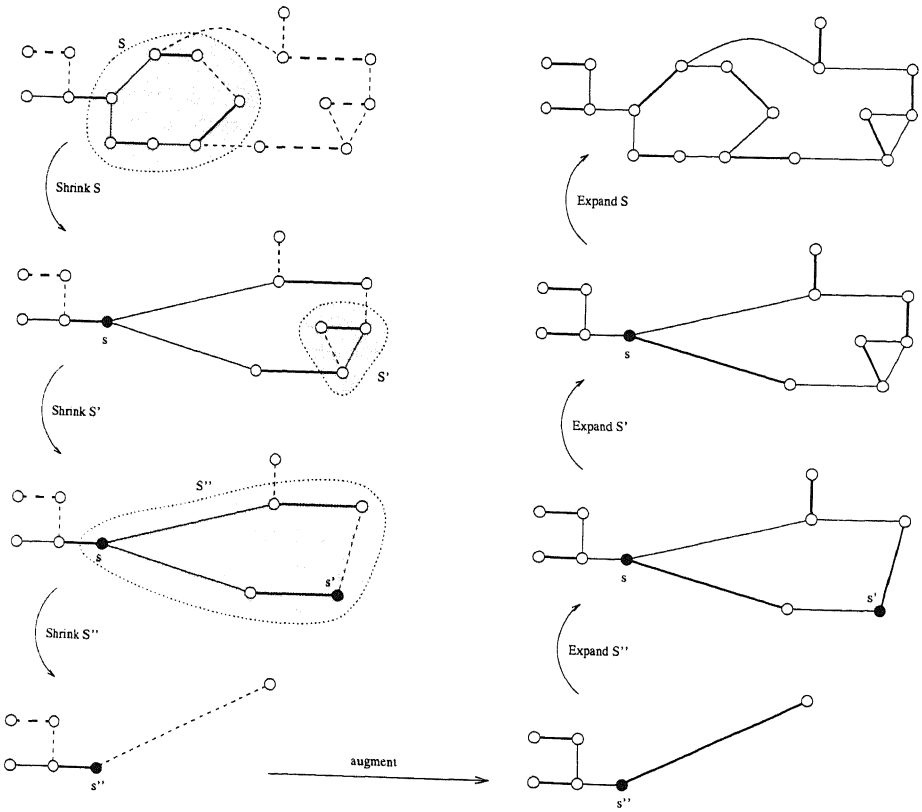

Fig. 5.

Let Q be an augmenting path in G with endpoints u and v, where $u \neq t$. If Q is disjoint from S, it is augmenting in $G \times S$. Otherwise, $Q \times S$ contains a unique us-path P. Clearly, P is augmenting in $G \times S$, so $M \times S$ is not maximum in $G \times S$. □

Thus we have an algorithm for finding a largest matching in a non-bipartite graph. This algorithm, called the *blossom algorithm* has been developed by Edmonds [1965c] (Figure 5 continues the examples in Figures 3 and 4). Witzgall & Zahn [1965] developed an $O(|V|^3)$ algorithm for general non-bipartite matching that does not rely on shrinking.

Clearly, the blossom algorithm can be implemented to run in polynomial time. Edmonds' implementation runs in $O(|V|^4)$. Balinski [1969], Gabow [1973, 1976], and Lawler [1976] developed $O(|V|^3)$ versions (but only the latter two authors explicitly state the running time). Developing an $O(|V|^3)$ version requires careful implementation of SHRINK.

2.2.2. Implementation of the blossom algorithm

Assume that during the search for an augmenting path, the blossom algorithm successively generates the graphs $G = G_0, G_1, \ldots, G_k$, i.e., GROW identifies the blossom S_i in G_i and SHRINK produces the graph $G_{i+1} = G_i \times S_i$ by shrinking S_i to the pseudo-node s_i $(i = 0, \ldots, k-1)$. We identify each node v in G_i, not in S_i, with the corresponding node in G_{i+1}. Pseudo-nodes are considered to be new elements. (So, $V(G_{i+1}) = (V(G_i) \setminus S_i) \cup \{s_i\}$ for $i = 0, \ldots, k-1$.)

We use the following notation to represent the relations between the pseudo-node s_i and the set S_i of nodes and pseudo-nodes 'contained' in it. For each node $s \in \bigcup_{i=0}^{k} V(G_i)$ $(= V(G) \cup \{s_0, \ldots, s_{k-1}\})$ we define:

$$\text{SHALLOW}[s] := \begin{cases} \emptyset & \text{if } s \in V(G) \\ [t_0, \ldots, t_\ell] & \text{if } s = s_i \ (i = 0, \ldots, k-1),\ t_0t_1, t_1t_2, \ldots, t_\ell t_0 \\ & \text{is the alternating odd circuit defining the} \\ & \text{blossom } S_i, \text{ and } t_0 \text{ is the tip of } S_i, \end{cases}$$

(so SHALLOW$[s]$ is an ordered list) and

$$\text{BLOSSOM}[s] := \begin{cases} t & \text{if } s \in \text{SHALLOW}[t],\ t \in \{s_0, \ldots, s_{k-1}\} \\ \emptyset & \text{otherwise.} \end{cases}$$

We define, recursively, the relations:

$$\text{DEEP}[s] := \begin{cases} s & \text{if } s \in V(G) \\ \bigcup_{t \in \text{SHALLOW}[s]} \text{DEEP}[t] & \text{otherwise} \end{cases}$$

and

$$\text{OUTER}_i[s] := t \text{ if } s \in \text{DEEP}[t],\ t \in V(G_i).$$

During the execution of the algorithm, we are mainly interested in the most recently constructed graph G_k and so denote OUTER$_k$ by OUTER. Together with the adjacency lists representing G, OUTER represents the current shrunken graph G_k.

Implementation 1 – explicitly updating OUTER: We maintain the functions SHALLOW, DEEP and OUTER as data structures. Each time we detect a blossom S, we introduce a new pseudo-node s. We can determine SHALLOW$[s]$ and the nodes in S in $O(|S|)$ time and we can determine DEEP$[s]$ and update the array OUTER in $O(\sum_{t \in \text{SHALLOW}[s]} |\text{DEEP}[t]|) = O(|V(G)|)$ time. So, since we shrink at most $\frac{1}{2}|V(G)|$ times between successive augmentations, we spend $O(|V(G)|^2)$ time updating SHALLOW, DEEP and OUTER between successive augmentations.

It is easy to implement GROW with these data structures. We grow an alternating forest F' in the current shrunken graph G' represented by OUTER by scanning the edges uv in G such that OUTER$[u] \in \text{even}(F')$. Note that we can determine OUTER$[u]$ and OUTER$[v]$ in constant time. We keep track of the forest by creating a label FOREST[OUTER$[v]$]$= u$ for the node OUTER$[v]$ each time we decide to add the edge uv to F'. When we shrink a blossom S with tip t into a pseudo-node s, we easily create $F' \times S$ in the new graph $G' \times S$ by setting FOREST$[s]$:= FOREST$[t]$. Implemented in this way, GROW takes $O(|E(G)|)$ time (disregarding the time spent on updating SHALLOW, DEEP and OUTER), just as in the Hungarian method.

When we detect an augmenting path in G', we can construct it using FOREST and expand it to an augmenting path in G using OUTER and the ordered list SHALLOW. This, and carrying out the augmentation, takes $O(|E(G)|)$ time plus $O(|V(G)|^2)$ time for updating OUTER (which is necessary each time we expand a blossom).

As there are at most $|V(G)|$ augmentations, we obtain the following result:

> *The blossom algorithm can be implemented to run in* $O(|V(G)|^3)$ *time.* (12)

In sparse graphs, when $|E(G)|$ is significantly smaller than $\binom{|V(G)|}{2}$, the running time is dominated by the time required to update DEEP and OUTER. The other operations take $O(|V(G)||E(G)|)$ time. So, to improve the running time one could economize on the implementation of the blossoms. This can be done by employing a more implicit method of updating OUTER.

Implementation 2 – implicitly updating OUTER: To represent the blossoms, we maintain a single (dynamic) function IN, with the property that IN$[u]$ = OUTER$_i[u]$ for some (unspecified) i. Initially, IN$[u] = u$. When we shrink a blossom S to a pseudo-node s we update IN by resetting IN$[u]$:= s for $u \in S$ and setting IN$[s]$:= s. This takes $O(|S|)$ time. So, between two successive augmentations we spend, overall, $O(|E(G)|)$ time updating IN. To determine OUTER$[u_0]$, we iterate u_i := IN$[u_{i-1}]$ until, at some point u_k = IN$[u_k]$, then OUTER$[u_0] = u_k$. This can take as many as $\frac{1}{2}|V(G)|$ iterations, but – in the hope that the next time we need OUTER$[u_0]$, we get it almost for free – we reset IN$[u_i]$:= OUTER$[u_0](= u_k)$ for each $i = 0, \ldots, k$.

The disadvantage of this approach is that we no longer have the data structure SHALLOW to help expand augmenting paths. We can, however, overcome this by extending the labels in FOREST to include labels on the nodes in a blossom

indicating how to trace an augmenting path through the blossom. It is quite straightforward to find such labeling, but some care is needed to handle the tips of the blossoms properly. The labeling can be implemented so that finding the labels and using them to find an augmenting path can be carried out in $O(|E(G)|)$ time per augmentation [see Lawler, 1976; Lovász & Plummer, 1986; or Tarjan, 1983].

Implementing the blossoms with IN, *the blossom algorithm uses 'almost a constant' times* $|E(G)||V(G)|$ *steps.* (13)

Gabow [1973, 1976] demonstrated this result. To make the statement (13) precise, 'almost a constant' refers to a function $\alpha(|E(G)|, |V(G)|)$ (the inverse of the Ackerman function) that grows very slowly. The procedure is in fact a standard implementation of the 'set union' problem: blossoms are sets, which are united into new sets every time we shrink. For a precise definition of α and a proof of (13), see Aho, Hopcroft & Ullman [1974] or Tarjan [1983]. Gabow & Tarjan [1983] developed a linear time algorithm for set union problems with special structure. As blossoms have that structure [cf. Gabow & Tarjan, 1983], this implies:

The blossom algorithm can be implemented in $O(|E(G)||V(G)|)$ *time.* (14)

The same time bound has been achieved by Kameda & Munro [1974], but in a different manner. Instead of fine tuning the implementation of the set union problem, they obtain the $O(|V||E|)$ time bound by growing the alternating forest in a depth-first manner.

The result (14) and the way it is achieved reflect the following perspective on Edmond's blossom algorithm: The algorithm searches for augmenting paths as though the graph were bipartite and, as soon as it encounters an odd circuit, it 'shrinks the trouble away'. So, we might hope that by implementing shrinking efficiently, we could achieve the same time bound for non-bipartite matching as for the bipartite matching algorithm used as a subroutine. We have just seen that this is indeed the case when the bipartite matching subroutine is the Hungarian method. Generally, it appears that the hope is idle. For instance, we cannot simply apply this idea to Hopcroft and Karp's $O(\sqrt{|V|}|E|)$ algorithm for bipartite matching because shrinking changes the length of augmenting paths. On the other hand, the $O(\sqrt{|V|}|E|)$ bound can be achieved for non-bipartite graphs. Indeed, Hopcroft and Karp's algorithm can be generalized to achieve this bound, but the generalization is far from trivial. Even and Kariv developed an $O(|V|^{5/2})$ algorithm (as did Bartnik [1978]) and an $O(\sqrt{|V|}|E|\log|V|)$ algorithm [Even & Kariv, 1975; Kariv, 1976]. The gap was finally bridged by Micali & Vazirani [1980], who developed an $O(\sqrt{|V|}|E|)$ algorithm for general cardinality matching (see Vazirani [1994] for a proof of correctness). Blum [1990a, b] and Gabow & Tarjan [1991] achieved the same time bound in different manners.

3. Bipartite matching duality

A subset $N \subseteq V(G)$ is called a *node cover* if each edge has at least one endpoint in N. $\tau(G)$ denotes the minimum cardinality of a node cover in G. Because a node cover is always at least as large as a matching we have that, for any graph G, bipartite or not:

$$\nu(G) \leq \tau(G). \tag{15}$$

In the complete graph K_3 on 3 nodes: $\nu(K_3) = 1 \neq 2 = \tau(K_3)$. So, ν and τ need not be equal. But, when G is bipartite we have equality in (15):

Theorem 5 [König, 1931, 1933]. *For each bipartite graph G, $\nu(G) = \tau(G)$.*

Proof. Let F be a Hungarian forest with respect to a maximum matching M. Then $N := (V_1 \setminus \text{even}(F)) \cup (V_2 \cap \text{odd}(F))$ is a node cover with $|N| = |M|$. □

Many equivalent versions of *König's theorem* (Theorem 5) appeared in the first half of this century. The oldest of these is probably the following result due to Frobenius [1917]. (For a short 'linear algebra'-proof see Edmonds [1967].)

> *The determinant of a square matrix A viewed as a polynomial in its non-zero coefficients is identically zero, i.e., is zero for all values of its non-zero coefficients, if and only if there exists, for some p with $0 < p \leq n$, a $p \times (n - p + 1)$ submatrix of A having only zero coefficients.* (16)

On the other hand, the best known version addresses the existence of a system of distinct representatives. A *system of distinct representatives* for a finite collection of finite sets $S_1, \ldots, S_n$ is a collection of distinct elements $s_1, \ldots, s_n$ with $s_i \in S_i$ for each $i \in \{1, 2, \ldots, n\}$.

> *There exists a system of distinct representatives for $S_1, \ldots, S_n$ if and only if $|\cup_{i \in I} S_i| \geq |I|$ for each $I \subseteq \{1, 2, \ldots, n\}$.* (17)

Formulated in terms of matchings in bipartite graphs, (16) and (17) become:

Theorem 6 [Frobenius, 1917; Hall, 1935]. *In each bipartite graph $G = (V_1 \cup V_2, E)$, exactly one of the following holds:*

- *$\nu(G) = |V_1|$, i.e. there exists a matching 'of V_1 into V_2';*
- *there exists a set $U \subseteq V_1$, such that $|\Gamma(U)| < |U|$.*

Here, $\Gamma(U) := \{v \in V_2 \mid \text{there is a node } u \in U \text{ with } uv \in E\}$. (See Ore [1955] for a version of König's theorem in terms of $\Gamma(U)$.) In fact, (16) is the special case of Theorem 6 in which $|V_1| = |V_2|$. Clearly, Theorem 6 follows from König's theorem. If $\nu(G) < |V_1|$, there is a node cover N with $|N| = \nu(G) < |V_1|$. Since N is a node

cover, $U := V_1 \setminus N$ satisfies $\Gamma(U) \subseteq V_2 \cap N$, i.e., $|\Gamma(U)| \leq |V_2 \cap N| < |V_1 \setminus N| = |U|$. On the other hand, even though Theorem 6 seems a rather special case of König's theorem, the latter can easily be proved from the former. (Add $|V_1| - \tau(G)$ new nodes to V_2, each adjacent to every node in V_1.) This is an example of the self-refining nature of matching theory.

Frobenius used (16) to simplify the proof of one of his earlier results [Frobenius, 1912] describing when the determinant of a matrix viewed as a polynomial in its non-zero coefficients can be factored into polynomials of lower degree. König [1915] also gave a simpler proof of Frobenius' factoring result in which he pointed out the relation to matchings in graphs. Frobenius did not appreciate this relation: *'Die Theorie der Graphen, mittels deren Hr. König den obigen Satz abgeleitet hat, is nach meiner Ansicht ein wenig geeignetes Hilfsmittel für die Entwicklung der Determinantentheorie. In diesem Falle führt sie zu einem ganz speziellen Satz von geringem Werte* [essentially statement (18) below]. *Was von seinem Inhalt Wert hat, ist in dem Satze II* [statement (16)] *ausgesprochen'* [Frobenius, 1917]. Moreover he did not acknowledge König's proof, though König had sent it to him. Apparently, this did not please König [see König, 1933, 1936]. Like so many fields, matching theory also has its controversies. (Schneider [1977], in trying to reconstruct the issue, speculates that Frobenius had not penned this criticism. He hypothesized that because Frobenius was already very ill at the time – he died in August 1917 – someone else finished the paper and wrote the criticism. See Schneider [1977] for Mirsky's refutation of this hypothesis.)

König also proved the following consequence of Theorem 6 for *regular* graphs, i.e., graphs in which all nodes have the same degree.

> *Each regular bipartite graph admits a perfect matching* [König, 1916a, b]. (18)

An immediate consequence of (18) deals with edge colorings in bipartite graphs. An *edge coloring* is an assignment of colors to the edges so that if two edges share an end node they get different colors. The minimum number of colors needed to color the edges of G in this way is denoted by $\chi_e(G)$. Clearly, $\chi_e(G)$ is at least the maximum degree of a node in G, denoted $\Delta(G)$. In finding an edge coloring we may assume that the graph is regular (just add edges and nodes to G until this is the case; of course without changing the maximum degree). Since an edge coloring is just a partition of the edge set into matchings, (18) implies the following.

> *For each bipartite graph G, $\chi_e(G) = \Delta(G)$* [König, 1916a, b]. (19)

We digress briefly from our discussion of bipartite graphs to point out that although $\chi_e(G)$ is either $\Delta(G)$ or $\Delta(G) + 1$ for each simple (non-bipartite) graph G [Vizing, 1964, 1965], determining whether or not $\chi_e(G) = \Delta(G)$ is $\mathcal{NP}$-hard, even when G is regular of degree 3 [Holyer, 1981].

Another form of edge coloring asks for a coloring of the edges of G so that there is at least one edge of each color incident to each node. A coloring of this form corresponds to a partition of $E(G)$ into edge covers. An *edge cover* in a

graph G is a collection of edges F such that each node in V is an endpoint of at least one of the edges in F. Clearly, when the minimum degree of a node in G is $\delta(G)$, a coloring of this form cannot use more than $\delta(G)$ colors. König [1916a, b] showed that when G is bipartite this upper bound is achievable, i.e., there is a coloring of the edges using $\delta(G)$ colors such that each node is incident to each color.

König's theorem establishes a strong relationship between matchings and node covers in bipartite graphs. There is also a strong relationship in bipartite graphs between edge covers and matchings and between node covers and stable sets. A *stable set* in G is a collection of mutually non-adjacent nodes in G. We use $\alpha(G)$ to denote the maximum cardinality of a stable set in G and $\rho(G)$ to denote the minimum cardinality of an edge cover in G. Clearly $\alpha(G) \leq \rho(G)$. The following relationship among these problems is true for all graphs, bipartite or not.

Theorem 7 [Gallai, 1959]. *For each graph $G = (V, E)$ without isolated nodes, $\alpha(G) + \tau(G) = |V| = \nu(G) + \rho(G)$.*

Proof. The first equality is trivial: stable sets are exactly the complements of node covers. To prove the second equality we use edge/node covers instead of edge covers. An *edge/node cover* of G is a covering of the nodes of G by edges and nodes. It is easy to see that the minimum cardinality of an edge/node cover is also $\rho(G)$: each edge cover is an edge/node cover, and conversely, each edge/node cover can be turned into an edge cover of the same cardinality by replacing each node by an incident edge (G has no isolated nodes). Now observe that the edges a minimum cardinality edge/node cover may assumed to form a matching – a maximum matching in fact. So: $\rho(G) = \text{def}(G) + \nu(G) = |V| - \nu(G)$. □

An immediate consequence of this result is:

For each bipartite graph $G = (V_1 \cup V_2, E)$ without isolated nodes, $\alpha(G) = \rho(G)$ [König, 1931]. (20)

Bipartite graphs are not the only graphs with $\nu(G) = \tau(G)$ and, equivalently, $\alpha(G) = \rho(G)$. The class of graphs with this property, called the *König property*, has been characterized by Deming [1979] and Sterboul [1979] (see also Bourjolly & Pulleyblank [1989], Lovász [1983] and Lovász & Plummer [1986]).

Intermezzo: min–max relations and good characterizations

Both (20) and Theorem 7 are characterizations for the maximum size of a stable set. Even though (20) applies only to bipartite graphs we consider it to be a 'better' characterization than Theorem 7. The reason is that (20) assures that whatever the answer to the question: Given a bipartite graph G, is $\alpha(G) \leq k$? is, we can always give a polynomial length certificate of it. Namely, either a stable set with k nodes or an edge-cover with less than k edges. This is not the case with Theorem 7. If $\alpha(G) < k$ it only guarantees that **all** node-covers of G are larger than $|V| - k$, and it is not clear how to verify that.

A problem is called *well-characterized* if whatever the answer is, there exists a polynomial length certificate for the correctness of that answer. Note that the existence of the certificate might not guarantee us that we can find it in polynomial time. A theorem asserting that a problem is well-characterized is called a *good characterization* for the problem. (In NP-language: a decision problem is well-characterized if it belongs to NP ∩ co-NP.)

So '$\alpha(G) \leq k$?' is well-characterized for bipartite graphs and (20) is a good characterization for it. Theorem 7 is not a good characterization for '$\alpha(G) \leq k$?'. For non-bipartite graphs, '$\alpha(G) \leq k$?' is not known to be well-characterized. (If it would be, then NP = co-NP, which is generally believed not to be true.) It was Edmonds who first explicitly made the distinction between characterizations that are good and those that are not.

Good characterizations for optimization problems often come in the form of a min–max relation, like Theorem 5 or (20). However, they can have other forms as well. For instance a polynomial time algorithm is a good-characterization (implying that P ⊆ NP ∩ co-NP). On the other hand, many polynomial time algorithms for optimization problems use a good characterization, mostly a min–max relation, as stopping criterion – also in this chapter. The theorems in this section (except for Theorem 7) are good characterizations, and there are more to come in this chapter.

3.1. Network flows

There is a strong relation between bipartite matching problems and flow problems; essentially they are identical.

Given a directed graph $D = (V(D), A(D))$ and two nodes s and t. An *s,t-flow* is a function f from $A(D)$ to $\mathbb{R}_+$ such that for each $v \in V(D) \setminus \{s,t\}$: $f(\delta^-(v)) = f(\delta^+(v))$. The *value* of a flow f is defined by $f(\delta^+(s)) - f(\delta^-(s))$ $(= f(\delta^-(t)) - f(\delta^+(t)))$. The *max-flow problem* is to find a flow f of maximum value subject to the *capacity constraints*: $f(a) \leq c(a)$ for each $a \in A$. Here, c is a given *capacity function* from $A(D)$ to $\mathbb{R}_+ \cup \{\infty\}$. For each $U \subseteq V(D)$ with $s \in U, t \notin V$, the *capacity* of the *s,t-cut* $\delta^+(U)$ is defined to be $c(\delta^+(U))$. Flow values and cut capacities satisfy the following min–max relation known as the *Max-flow min-cut theorem*.

Theorem 8 [Ford & Fulkerson, 1956; Elias, Feinstein & Shannon, 1956]. *The maximum value of an s,t-flow with respect to a given capacity function is equal to the minimum capacity of an s,t-cut. Moreover, if the capacity function is integral, then there is an integral maximum flow.*

The following construction demonstrates the relation between flow problems and bipartite matching problems. Given a bipartite graph $G = (V_1 \cup V_2, E)$; construct a directed graph $D = (V(D), A(D))$ as follows. Add a node s to V and directed edges from s to each node of V_1. Orient all edges in G from V_1 to V_2. Add a node t and a directed edge from each node of V_2 to t. Assign the

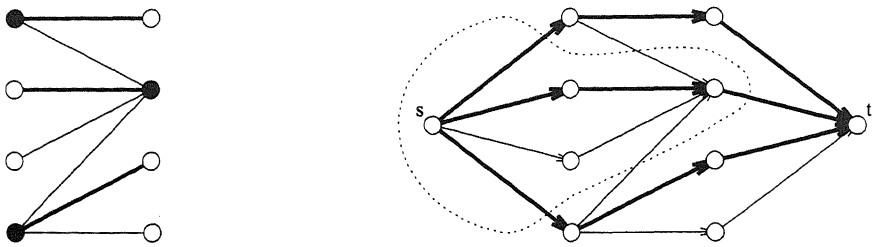

Fig. 6.

following capacities to the directed edges. Each arc corresponding to an edge in G has infinite capacity. The arcs out of s or into t each have capacity 1. There is a one-to-one correspondence between matchings in G and integral flows in D and between node covers in G and cuts with finite capacity in D. Figure 6 illustrates these relations: bold edges form a matching and bold arcs indicate a corresponding flow (bold arcs carry flow 1; the other arcs carry flow 0). The black nodes indicate a node cover and the dotted balloon identifies the set U such that $\delta^+(U)$ is the corresponding cut. Thus, König's theorem follows from the Max-flow min-cut theorem. Similarly, the algorithm for finding a maximum matching in a bipartite graph is essentially Ford and Fulkerson's maximum flow algorithm.

Conversely, it is possible to derive the Max-flow min-cut theorem from König's theorem. Instead, we use König's theorem to prove another closely related result on directed graphs, namely *Menger's theorem* (Theorem 9).

Let D be a directed graph and let $S, T \subseteq V(D)$. An S, T-*path* in D is a directed path from a node in S to a node in T. A set $U \subseteq V(D)$ is called an S, T-*separator* if it intersects each S, T-path.

Theorem 9 [Menger, 1927]. *Let D be a directed graph and let $S, T \subseteq V(D)$. Then the maximum number of mutually node-disjoint S, T-paths is equal to the minimum cardinality of an S, T-separator.*

Proof. Clearly, we may assume that S and T are disjoint. Let $W := V(D) \setminus (S \cup T)$ and construct a bipartite graph G as follows. For each node $u \in W$ we have two nodes u^+ and u^- in G. For each node $u \in S$ we have only one node u^+ in G. Similarly, for each node $u \in T$ we have one node u^- in G. (We refer to the set $\{u^+ \mid u \in S\}$ of nodes in G as S and to the set $\{u^- \mid u \in T\}$ of nodes in G as T.) There are two types of edges in G: for each arc $\overrightarrow{uv}$ in D, u^+v^- is an edge in G and for each $u \in W$, u^+u^- is an edge in G.

Let M be a maximum matching in G and assume, without loss of generality, that $\exp(M) \subseteq S \cup T$. Let $k := |S \setminus \exp(M)| = |T \setminus \exp(M)|$. It is easy to see that the collection $\{\overrightarrow{uv} \in A(D) \mid u^+v^- \in M\}$ forms the node-disjoint union of k directed S, T-paths and a collection of directed circuits in D.

To prove the theorem it suffices to show the existence of an S, T- separator with cardinality k. Let N be a minimum node cover. By König's theorem, $|N| = \nu(G) =$

$\frac{1}{2}(|V(G)|-|\exp(M)|) = \frac{1}{2}(2|W|+|S|+|T|-|\exp(M)|) = \frac{1}{2}(2|W|+2k) = |W|+k$. For each node u in W, to cover the edge u^+u^- in G either u^+ or u^- must be in N. Hence the set $U := (S \cap N) \cup (T \cap N) \cup \{u \in W \mid u^+ \text{ and } u^- \in N\}$ has cardinality k. It remains to prove that U is an S, T-separator. Let $u_0, \ldots, u_t$ be a directed S, T-path. Since $|N \cap \{u_0^+, u_1^-, u_1^+, \ldots, u_t^-\}| \geq t$, either u_0^+ is in N in which case u_0 is in U, u_t^- is in N in which case u_t is in U, or $\{u_i^-, u_i^+\} \subseteq N$ for some $0 < i < t$ in which case u_i is in U. □

There are many equivalent versions of Menger's theorem. One can, for instance, consider internally node-disjoint directed s, t-paths (shrink S and T to the single nodes s and t). Further, similar min–max relations hold for arc-disjoint s, t-paths as well as node- or edge-disjoint s, t-paths in undirected graphs. All these results are equivalent in the sense that we can easily derive one from another. In fact, all these results can be seen as versions of König's theorem (or conversely).

Menger's theorem for the number of arc-disjoint s, t-paths is a special case of the Max-flow min-cut theorem in which all capacities are one. And conversely, one can derive the Max-flow min-cut theorem from this version of Menger's theorem. So there is a close relationship between matchings in bipartite graphs and flows. It extends to almost every problem on bipartite graphs discussed in this chapter. For instance, the minimum weight matching problem in bipartite graphs corresponds to the *min-cost flow problem* in which we are given the unit costs of flow through each arc and are asked to find a maximum flow of minimum total cost.

For further consequences of König's theorem, e.g., Dilworth's theorems on chains and anti-chains in partially ordered sets [Dilworth, 1950], see Lovász & Plummer [1986] or Mirsky [1971]. For a tour along all the above mentioned equivalences, see Reichmeider [1984].

3.2. Intermezzo: perfect graphs and matroids

We conclude this section with a short discussion on extensions of the results on matchings in bipartite graphs described in this section. This brings us outside the context of bipartite graphs. We go here in two directions: first, node coloring and stable sets in general graphs, and second, matroids.

3.2.1. Node coloring and stable sets

The *chromatic number* $\chi(G)$ of G is the minimum number of colors needed to color the nodes of G such that adjacent nodes receive different colors. $\chi(G)$ is at least the maximum size $\omega(G)$ of a collection of mutually adjacent nodes in G. A graph is called *perfect* if and only if $\chi(H) = \omega(H)$ for each induced subgraph H in G. Odd circuits are not perfect. On the other hand, bipartite graphs are, trivially, perfect. König's results stated in the beginning of this section yield three other, less trivial, classes of perfect graphs.

We need a few definitions. The *complement* $\overline{G}$ of a graph G has the same nodes as G; nodes are adjacent in $\overline{G}$ when they are non-adjacent in G. The *line graph*

of G has the edges of G as nodes; two edges in G are adjacent in the line graph when they share an end node in G. Clearly, $\chi_e(G)$ is the chromatic number of the line graph of G. Similarly, matchings in G are stable sets in the line graph of G.

With these definitions we get the following results. By (20), the complement of bipartite graph is perfect. By (19), the line graph of a bipartite graph is perfect. And, by Theorem 5, the complement of the line graph of a bipartite graph is perfect.

So perfect graphs not only generalize bipartite graphs but also their line graphs. Moreover, also the complements of all these graphs are perfect. The latter is not so much of a coincidence: *The complement of a perfect graph is perfect*. This is the famous *Perfect graph theorem* proved by Lóvasz [1972b]. Although, many classes of perfect graphs have been discovered over the last decennia, the main conjecture on perfect graphs is still open: *If a graph is not perfect, then it or its complement contains an odd circuit with 5 or more edges as an induced subgraph* [Berge, 1962].

The *stable set problem* is: Given a graph G and a weight function on $V(G)$, find a stable set of maximum weight. In general this problem is NP-hard [Karp, 1972]. On the other hand, by the Perfect graph theorem, we have a min–max relation for the maximum cardinality stable set in a perfect graph: $\alpha(G) = \chi(\overline{G})$. In fact, the stable set problem can be solved in polynomial time when G is perfect [Grötschel, Lovász & Schrijver, 1981, 1984, 1988].

There is another class of graphs for which the stable set problem is polynomially solvable. In fact, it is very strongly related to matchings. A graph is *claw-free* if it has no node with three mutually non-adjacent neighbors. Line graphs are claw-free. Sbihi [1980] and Minty [1980] proved that one can find a maximum cardinality stable set in a claw-free graph in polynomial time. Minty [1980] did this by reducing the problem to a matching problem [see also Lovász & Plummer, 1986]. His algorithm also solves the weighted case.

3.2.2. Matroid intersection

In this section we will see an extension of König's theorem.

A matroid $\mathcal{M} = (E, \mathcal{I})$ consists of a finite set E, the *ground set*, and a collection $\mathcal{I}$ of subsets of E satisfying the following three axioms: $\emptyset \in \mathcal{I}$; $I \in \mathcal{I}$ and $J \subseteq I$ implies that $J \in \mathcal{I}$; and, finally, $I, J \in \mathcal{I}$ and $|I| < |J|$ implies that there exist an $e \in J \setminus I$ such that $I \cup \{e\} \in \mathcal{I}$. The members of $\mathcal{I}$ are called *the independent sets of* $\mathcal{M}$. The *rank-function* $r_{\mathcal{M}}$ of a matroid $\mathcal{M}$ is defined by $r_{\mathcal{M}}(F) := \max\{|I| \mid I \in \mathcal{I}; I \subseteq F\}$. Examples of matroids are: the edge sets of forests in a graph and the linearly independent collections of columns of a matrix.

There is a vast theory on matroids [see Welsh, 1976; Recski, 1989; Truemper, 1992; Oxley, 1992] and many of the results there are inspired by results on matchings [see Lovász & Plummer, 1986]. Here we just mention one of these results.

The *matroid intersection* problem is: given two matroids on the same ground set find the largest set that is independent in both matroids. The maximum

matching problem in a bipartite graph is a matroid intersection problem: given $G = (V_1 \cup V_2, E)$; define $\mathcal{M}_1$ and $\mathcal{M}_2$ with ground set $E(G)$, and with $\mathcal{I}_i := \{I \subseteq E \mid |I \cap \delta(v) \leq 1 \text{ for each } v \in V_i\}$ for $i = 1, 2$. It is easy to check that these are matroids, and that a collection of edges is independent in both $\mathcal{M}_1$ and $\mathcal{M}_2$ if and only if it is a matching.

Edmonds [1970] derived the following min–max relation for the matroid intersection problem.

Let $\mathcal{M}_1 := (E, \mathcal{I}_1)$ and $\mathcal{M}_2 := (E, \mathcal{I}_2)$ be two matroids on the same ground set E. Then $\max\{|I| \mid I \in \mathcal{I}_1 \cup \mathcal{I}_2\} = \min\{r_{\mathcal{M}_1}(F) + r_{\mathcal{M}_2}(E \setminus F) \mid F \subseteq E\}$. (21)

Königs theorem is a special case of (21). Indeed, let $G = (V_1 \cup V_2, E)$ be a bipartite graph and let $\mathcal{M}_1$ and $\mathcal{M}_2$ be the two matroids defined above. If F is a set of edges in G then $r_{\mathcal{M}_i}(F)$ is the cardinality of set of nodes in V_i incident to at least one of the edges in F. From that the relation between sizes of node covers and $r_{\mathcal{M}_1}(F) + r_{\mathcal{M}_2}(E \setminus F)$ is easy.

A similar extension of non-bipartite matching to a problem on matroids is the 'matroid matching' problem [see Lovász & Plummer, 1986].

4. Non-bipartite matching duality

We begin our discussion of non-bipartite matching by developing a min–max relation for the size of a maximum matching. Although $\tau(G)$ does not, as we have seen, provide a good characterization for the size of a maximum matching in a non-bipartite graph G, there is a min–max relation for $\nu(G)$. An *odd component* of a graph G is a connected component of G with an odd number of nodes. We let $c_o(G)$ denote the number of odd components of $G = (V, E)$.

For each matching M and subset $B \subseteq V$, $|\exp(M)| \geq c_o(G \setminus B) - |B|$. (22)

To see this, let M_1 be the set of the edges in M with both endpoints in $G \setminus B$. As M_1 leaves at least one node exposed in each odd component of $G \setminus B$, $|\exp(M_1)| \geq c_o(G \setminus B) + |B|$. Since, $|\exp(M)| = |\exp(M_1)| - 2|M \setminus M_1| \geq |\exp(M_1)| - 2|B|$, (22) follows.

Theorem 10 [Berge, 1958]. *For each graph $G = (V, E)$:*

$$\operatorname{def}(G) = \max_{B \subseteq V} c_o(G \setminus B) - |B|,$$

$$\nu(G) = \min_{B \subseteq V} \tfrac{1}{2}(|V| - c_o(G \setminus B) + |B|).$$

Proof. Clearly, it suffices to prove the formula for $\operatorname{def}(G)$. Let M be a maximum matching in G. Apply the procedures GROW and SHRINK to G and M until

we get a shrunken graph G' with a Hungarian forest F'. Each odd node in F' is a node of the original graph G and so is not contained in a pseudo-node. Each odd component of $G \setminus \text{odd}(F')$ has been shrunk into a different even node of F' (or is equal to an even node). Moreover each even node arises in this way. Hence, $c_o(G \setminus \text{odd}(F')) = |\text{even}(F')|$. So, $c_o(G \setminus \text{odd}(F')) - |\text{odd}(F')| = |\text{even}(F')| - |\text{odd}(F')| = \text{def}(G') = \text{def}(G)$. Combining this with (22), the theorem follows. □

Theorem 10 generalizes Tutte's characterization of those graphs that contain a perfect matching.

Theorem 11 [Tutte, 1947, 1954]. *The graph $G = (V, E)$ has a perfect matching if and only if*

$$c_o(G \setminus B) \leq |B| \text{ for each } B \subseteq V.$$

Tutte's used matrix-techniques ('Pfaffians') to prove this. The first proof of his theorem that uses alternating paths has been given by Gallai [1950].

Edmonds' *Odd set cover theorem* is a version of Berge's theorem that more explicitly extends König's theorem to non-bipartite graphs. An *odd set cover* is a collection B of nodes together with a collection $\{S_1, \ldots, S_k\}$ of subsets of V, each with odd cardinality, such that for each edge uv either $\{u, v\} \cap B \neq \emptyset$, or $\{u, v\} \subseteq S_i$ for some $i = 1, \ldots, k$.

Theorem 12 [Edmonds, 1965c]. *For each graph G,*

$$\nu(G) = \min \left\{ |B| + \sum_{i=1}^{k} \tfrac{1}{2}(|S_i| - 1) \;\middle|\; B \text{ and } S_1, \ldots, S_k \text{ form an odd set cover of } G \right\}.$$

Proof. Among all $B \subseteq V$ with $\frac{1}{2}(|V| - c_o(G \setminus B) + |B|) = \nu(G)$, choose one with $|B|$ maximum, and let $S_1, \ldots, S_k$ denote the components of $G \setminus B$. Then all S_i are odd (if $|S_i|$ would be even, then $B \cup \{v\}$ with $v \in S_i$ would contradict the choice of B). Hence B and $S_1, \ldots, S_k$ form an odd set cover. It satisfies $|B| + \sum_{i=1}^{k} \frac{1}{2}(|S_i| - 1) = |B| + \frac{1}{2}|V \setminus B| - \frac{1}{2}k = \frac{1}{2}(|V| - c_o(G \setminus B) + |B|) = \nu(G)$. As obviously the minimum is at least $\nu(G)$ the theorem follows. □

The following special case of Tutte's theorem is over a hundred years old. It clearly demonstrates the long and careful attention paid to perfect matchings. A *cubic graph* is one in which each node has degree three. An *isthmus* is an edge that, when deleted from G, disconnects the graph.

Theorem 13 [Petersen, 1891]. *Every connected, cubic graph with at most two isthmi contains a perfect matching.*

Proof. Let $B \subseteq V$ and let $S_1, \ldots, S_k$ denote the odd components of $G \setminus B$. As G is cubic, $|\delta(S_i)|$ is odd for all $i = 1, \ldots, k$. Hence, $3|B| \geq |\delta(B)| \geq \sum_{i=1}^{k} |\delta(S_i)| \geq 3(k-2) + 2 = 3c_o(G \setminus B) - 4$. So, $|B| \geq c_o(G \setminus B) - \frac{4}{3}$. Which implies that $|B| \geq c_o(G \setminus B)$ (as $|B| - c_o(G \setminus B)$ is even). So, by Theorem 11, we may conclude that G has a perfect matching. □

4.1. *The Edmonds–Gallai structure theorem*

A graph G can have many different maximum matchings and applying GROW and SHRINK to one of them can lead to many different Hungarian forests. The ultimate shrunken graph, however, is independent of the choice of matching or the order in which we apply GROW and SHRINK. This observation is one aspect of the Edmonds–Gallai structure theorem [Gallai, 1963, 1964; Edmonds, 1965c]. In this section we discuss the main features of the Edmonds–Gallai structure, which plays a role in the development of algorithms for finding maximum weight matchings. In fact, every polynomial time maximum matching algorithm that calculates the Edmonds–Gallai structure can be embedded in a 'primal-dual framework' for solving the weighted matching problem in polynomial time (see Section 6.2).

Suppose we have applied GROW and SHRINK to a graph G and a maximum matching M, yielding a Hungarian forest F in a shrunken graph G^*. We use the notation OUTER$[u]$ (for nodes u in $V(G)$) and DEEP$[u]$ (for nodes u in $V(G^*)$) introduced in Section 2.2. The *Edmonds–Gallai structure* of a graph G is the partition of $V(G)$ into three sets, $D(G)$, $A(G)$ and $C(G)$, defined by $D(G) := \{u \in V \mid \nu(G \setminus u) = \nu(G)\}$, $A(G) := \{u \in V(G) \setminus D(G) \mid u \text{ is adjacent to a node in } D(G)\}$ and $C(G) := V(G) \setminus (D(G) \cup A(G))$.

> *The set $D(G)$ is the union of the sets* DEEP$[u]$ *with* $u \in \text{even}(F)$. *In fact, the components of $G|D(G)$ are exactly the sets* DEEP$[u]$ *with* $u \in \text{even}(F)$. *Moreover,* $A(G) = \text{odd}(F)$. (23)

So, G^* is obtained from G by shrinking the components of $G|D(G)$. We call the shrunken graph G^* the *Edmonds–Gallai* graph of G. The result (23) follows from the definitions of GROW and SHRINK. All statements of (23) follow easily from the first one: if F is a Hungarian forest in G^*, $D(G)$ is the union of the sets DEEP$[u]$ with $u \in \text{even}(F)$. To see this, consider a node u in G. Construct a new graph H by adding a new node v to G adjacent to u and construct a new graph H^* by adding a new node v^* to G^* adjacent to OUTER$[u]$. Now, we have the following equivalences:

$$u \in D(G) \iff \nu(H) = \nu(G) + 1 \iff \nu(H^*) = \nu(G^*) + 1 \tag{24}$$

$$\text{OUTER}[u] \in \text{even}(F) \iff u \in \text{DEEP}[w] \text{ for some } w \text{ in } \text{even}(F). \tag{25}$$

So we only need to establish the equivalence of the third statement in (24) with the first one in (25). If OUTER$[u] \in \text{even}(F)$, consider the alternating forest F^* in H^* obtained by adding v^* as a component to F. The nodes v^* and OUTER$[u]$ are both in even(F^*) and in different components of F^*. So, in that case GROW will

find an augmenting path using the edge connecting these two nodes, implying that $\nu(H^*) = \nu(G^*) + 1$. On the other hand, if OUTER$[u] \notin$ even(F), then

$$\begin{aligned} \text{def}(H^*) &\geq c_o(H^* \setminus \text{odd}(F)) - |\text{odd}(F)| = \\ &= |\text{even}(F)| + 1 - |\text{odd}(F)| = \text{def}(G^*) + 1. \end{aligned} \tag{26}$$

So in that case, $\nu(H^*) = \nu(G^*)$. Thus (23) follows.

The relation between the Edmonds–Gallai structure and the Hungarian forest in the Edmonds–Gallai graph provides insight into the structure of all maximum cardinality matchings in G. We need a few definitions. A graph G is *factor-critical* if $\nu(G \setminus v) = \nu(G) = \frac{1}{2}(|V(G)| - 1)$ for each $v \in V(G)$. A matching is *near-perfect* if it has exactly one exposed node. We let $\kappa(G)$ denote the number of components of $G|D(G)$.

Each component of $G|D(G)$ is factor-critical and $\text{def}(G) = \kappa(G) - |A(G)|$. *Moreover, a matching M in G is maximum if and only if it consists of:*
- *a perfect matching in $C(G)$,*
- *a near-perfect matching in each component of $G|D(G)$,*
- *a matching of each node $u \in A(G)$ to a node in a distinct component of $G|D(G)$.* (27)

This is the *Edmonds–Gallai structure theorem*. Note that it implies all the results on non-bipartite matching duality stated earlier in this section. Every statement in (27) follows easily from the first one: each component of $G|D(G)$ is factor-critical. This follows inductively from (23) together with the fact that each graph spanned by an odd circuit – like a blossom – is factor-critical, and the following:

Let S be a subset of nodes in a graph G such that $G|S$ is factor-critical. If $G \times S$ is factor-critical then so is G. (28)

And, (28) is in turn immediate from:

Let S be a subset of nodes in a graph G such that $G|S$ is factor-critical. Let s be the pseudo-node in $G \times S$ obtained by shrinking S. Then, for each matching M in $G \times S$, there exists a near-perfect matching M_S in S such that $M \cup M_S$ is a matching in G with
- $\exp(M \cup M_S) = \exp(M)$ *if* $s \notin \exp(M)$,
- $\exp(M \cup M_S) = (\exp(M) \setminus \{s\}) \cup \{s_1\}$ *for some* $s_1 \in S$, *if* $s \in \exp(M)$. (29)

Dulmage & Mendelsohn [1958, 1959, 1967] derived the Edmonds–Gallai structure for bipartite graphs. The components of $G|D(G)$ in a bipartite graph G each consist of a single node (because, GROW can never HALT in a bipartite graph, or equivalently, because the only factor-critical bipartite graph consists of a single node and no edges).

The Edmonds–Gallai structure describes the maximum matchings of a graph as unions of (near-)perfect matchings in certain subgraphs and a matching in a bipartite subgraph (between $A(G)$ and $D(G)$). So, more detailed information on the structure of maximum matchings requires structural knowledge about perfect and near-perfect matchings. Indeed, such structural results exist. In addition to the 'Dulmage–Mendelsohn decomposition' for bipartite graphs [Dulmage & Mendelsohn, 1958], we have the 'ear-decomposition' of bipartite graphs developed by Brualdi & Gibson [1977]. Hetyei [1964], Lovász & Plummer [1975], Lovász [1983] extended the 'ear-decomposition' to, not necessarily bipartite, graphs with perfect matchings and Lovász [1972c] developed the 'ear-decomposition' of factor-critical graphs. Finally, Kotzig [1959a, b, 1960], Lovász [1972d] and Lovász & Plummer [1975] developed the 'brick decomposition' of graphs with perfect matchings. See Lovász [1987] for an overview of some of these results and Chapters 3, 4, and 5 of the book by Lovász & Plummer [1986] for an exhaustive treatment of the subject.

We conclude this section with some observations on how the Edmonds–Gallai structure behaves under certain changes of the graph. These observations facilitate our analysis of an algorithm for weighted matching in Section 6.2.

> *For each edge* $e = uv \in E(G)$, *where* $u \in A(G)$ *and* $v \in A(G) \cup C(G)$, G *and* $G \setminus e$ *have the same Edmonds–Gallai structure.* (30)

It is easy to see that this is true. A bit more complicated is:

> *For each pair of nodes* $u \in D(G)$ *and* $v \in C(G) \cup D(G)$ *not both in the same component of* $G|D(G)$, $\operatorname{def}(G \cup e) \leq \operatorname{def}(G)$, *where* $e = uv$. *Moreover, if* $\operatorname{def}(G \cup e) = \operatorname{def}(G)$, *then* $D(G \cup e) \supsetneq D(G)$. (31)

To see this, first observe that $\operatorname{def}(G \cup e) \leq \operatorname{def}(G)$. Further, if $\operatorname{def}(G \cup e) = \operatorname{def}(G)$, then $D(G \cup e) \supseteq D(G)$. Now, assume that $\operatorname{def}(G \cup e) = \operatorname{def}(G)$ and $D(G \cup e) = D(G)$. Obviously, this implies that $\kappa(G \cup e) \leq \kappa(G)$ and $|A(G \cup e)| \geq |A(G)|$. By (27), this yields $\kappa(G \cup e) = \kappa(G)$ and $|A(G \cup e)| = |A(G)|$ (otherwise $\operatorname{def}(G \cup e) < \operatorname{def}(G)$). But, this contradicts the definition of edge e.

> *Let* $S \subseteq V(G)$ *such that* $G|S$ *is factor-critical and such that the pseudo-node* s *in* $G \times S$, *obtained by shrinking* S, *is contained in* $A(G \times S)$. *Then* $\operatorname{def}(G) \leq \operatorname{def}(G \times S)$. *Moreover, if* $\operatorname{def}(G) = \operatorname{def}(G \times S)$ *then* $D(G) \supseteq D(G \times S)$. *Finally, if* $\operatorname{def}(G) = \operatorname{def}(G \times S)$ *and* $D(G) = D(G \times S)$, *then* $C(G) \supsetneq C(G \times S)$. (32)

The proof of (31) is similar to the proof of (31). By (29), $\operatorname{def}(G) \leq \operatorname{def}(G \times S)$. Further, if $\operatorname{def}(G) = \operatorname{def}(G \times S)$, then $D(G) \supseteq D(G \times S)$. Now assume that $\operatorname{def}(G) = \operatorname{def}(G \times S)$ and $D(G) = D(G \times S)$. Then $C(G) \supseteq C(G \times S)$ and $\kappa(G \times S) = \kappa(G)$. By (27), $|A(G)| = |A(G \times S)|$. Combining all this with $|V(G)| > |V(G \times S)|$ yields $|C(G)| > |C(G \times S)|$.

5. Matching and integer and linear programming

In the next section we discuss the problem of finding a maximum weight matching. In this section we show that the problem is a linear programming problem. We first observe that, like many combinatorial optimization problems, the maximum weight matching problem is an integer linear programming problem:

$$\begin{array}{lllll} \text{maximize} & \sum_{e\in E} w_e x_e & & & \\ \text{subject to} & x(\delta(v)) & \leq & 1 & (v \in V) \\ & x_e & \geq & 0 & (e \in E) \\ & x_e & \in & \mathbb{Z} & (e \in E). \end{array} \tag{33}$$

In general, integer linear programming problems are hard to solve; they are NP-hard [Cook, 1971]. On the other hand, linear programming problems are easy to solve. There are not only practically efficient (but non-polynomial) procedures like the simplex method [Dantzig, 1951], but also polynomial time algorithms [Khachiyan, 1979; Karmarkar, 1984] for solving linear programs. Moreover, we have a min–max relation for linear programming, namely the famous *LP-duality theorem* [Von Neumann, 1947; Gale, Kuhn & Tucker, 1951]:

$$\max\{w^\top x \mid Ax \leq b\} = \min\{y^\top b \mid y^\top A = w^\top;\ y^\top \geq 0\}. \tag{34}$$

This min–max relation provides a good characterization for linear programming. In this chapter one of problems in such a pair of linear programming problems will typically be a maximum or minimum weight matching problem. In that case we will refer to the other problem as the *dual problem*. Its feasible (optimal) solutions will be called the *dual feasible (optimal) solutions*.

One consequence of the LP-duality theorem is that a pair of solutions, a feasible solution x to the maximization problem and a feasible solution y to the minimization problem, are both optimal if and only if they satisfy the *complementary slackness conditions*:

$$y^\top(b - Ax) = 0. \tag{35}$$

The complementary slackness conditions, more than the linear programming algorithms, guide the algorithms for maximum weight matching in Sections 6 and 8.1. An obvious first attempt at a linear programming formulation of the weighted matching problem is the *LP-relaxation*:

$$\begin{array}{lllll} \text{maximize} & \sum_{e\in E} w_e x_e & & & \\ \text{subject to} & x(\delta(v)) & \leq & 1 & (v \in V) \\ & x_e & \geq & 0 & (e \in E). \end{array} \tag{36}$$

If (36) admits an integral optimum solution, that solution also solves (33). Hence the question arises: When does (36) admit an integral optimum solution for every weight function w? This question is equivalent to: When is the polyhedron

$$\text{Fract}(G) := \{x \in \mathbb{R}_+^E \mid x(\delta(v)) \leq 1 \quad (v \in V)\} \tag{37}$$

equal to the *matching polytope*:

$$\begin{aligned} \text{Match}(G) &:= \text{convex hull } \{x \in \mathbb{Z}_+^E | x(\delta(v)) \leq 1 \quad (v \in V)\} \\ &= \text{convex hull } \{\chi^M | M \text{ is a matching in } G\} \text{ ?} \end{aligned} \tag{38}$$

If $\text{Fract}(G) \neq \text{Match}(G)$, can we find another system of inequalities $Ax \leq b$ such that $\text{Match}(G) := \{x \in \mathbb{R}^{E(G)} | Ax \leq b\}$ (and thus, write (33) as $\max\{w^T x | Ax \leq b\}$)? In this section we answer these questions.

5.1. Bipartite graphs – the assignment polytope

Theorem 14. *Let G be an undirected graph. Then* $\text{Match}(G) = \text{Fract}(G)$ *if and only if G is bipartite.*

Proof. First, consider a bipartite graph G and a vector $x \in \text{Fract}(G)$. Let $F := \{e \in E | 0 < x_e < 1\} \neq \emptyset$ and select $K \subseteq F$ as follows. If F is a forest, let K be a path in F between two nodes of degree 1 in F. If F is not a forest, let K be a circuit in F, which – since G is bipartite – is even. In either case K is the disjoint union of two matchings, M_1 and M_2. It is easy to see that for some sufficiently small, positive ϵ both $x + \epsilon(\chi^{M_1} - \chi^{M_2})$ and $x - \epsilon(\chi^{M_1} - \chi^{M_2})$ are in $\text{Fract}(G)$. Moreover x is a convex combination of these two vectors. Thus, each extreme point of $\text{Fract}(G)$ is an integral vector. (A vector x in a set P is an *extreme point* of P if it cannot be expressed as a convex combination of other vectors in P.) In other words, each extreme point of $\text{Fract}(G)$ is the characteristic vector of a matching and so $\text{Fract}(G) \subseteq \text{Match}(G)$. The reverse inclusion holds trivially.

Next, consider a non-bipartite graph G. Since G is not bipartite, it contains an odd circuit C. Clearly, $x := \frac{1}{2}\chi^C$ is in $\text{Fract}(G)$. If x is also in $\text{Match}(G)$, then there are matchings M_i $(i = 1, \ldots, k)$ and non-negative numbers λ_i $(i = 1, \ldots, k)$, such that: $x = \sum_{i=1}^k \lambda_i \chi^{M_i}$ and $\sum_{i=1}^k \lambda_i = 1$. This implies that: $\frac{1}{2}|C| = x(C) = \sum_{i=1}^k \lambda_i \chi^{M_i}(C) \leq \sum_{i=1}^k \lambda_i \frac{1}{2}(|C| - 1) = \frac{1}{2}(|C| - 1)$; a contradiction. So, $x \notin \text{Match}(G)$ and $\text{Fract}(G) \neq \text{Match}(G)$. □

So, when G is bipartite, (36) has an integral optimum solution (the characteristic vector of a maximum weight matching).

Egerváry proved the following strengthening of Theorem 14.

Theorem 15 [Egerváry, 1931]. *Let $G = (V_1 \cup V_2, E)$ be a bipartite graph and $w \in \mathbb{Z}^E$. Then both of the following linear programming problems have integral optimum solutions.*

$$\begin{array}{llll} \text{maximum} & \sum_{e \in E} w_e x_e & = \text{minimum} & \pi(V_1 \cup V_2) \\ \text{subject to} & x(\delta(v)) \leq 1 \ (v \in V_1 \cup V_2) & \text{subject to} & \pi_u + \pi_v \geq w_{uv} \ (uv \in E) \\ & x_e \geq 0 \ (e \in E) & & \pi_v \geq 0 \quad (v \in V_1 \cup V_2). \end{array}$$

Proof. That the maximization problem admits an integral optimum solution is Theorem 14. So, we consider only the dual problem (i.e. the minimization problem).

Let π' be a, not necessarily integral, dual optimal solution. We show that $\tilde{\pi} \in \mathbb{Z}_+^{V_1 \cup V_2}$ defined by:

$$\tilde{\pi}_v := \begin{cases} \lfloor \pi'_v \rfloor & \text{if} \quad v \in V_1 \\ \lceil \pi'_v \rceil & \text{if} \quad v \in V_2 \end{cases} . \tag{39}$$

is an integral dual optimal solution.

To see that $\tilde{\pi}$ is feasible, observe that for each $uv \in E$

$$\tilde{\pi}_u + \tilde{\pi}_v = \lfloor \pi'_u \rfloor + \lceil \pi'_v \rceil = \lfloor \pi'_u + \lceil \pi'_v \rceil \rfloor \geq \lfloor \pi'_u + \pi'_v \rfloor \geq \lfloor w_{uv} \rfloor = w_{uv}. \tag{40}$$

Define for $\alpha \in \mathbb{R}$: $V_1^\alpha := \{u \in V_1 | \pi'_u - \lfloor \pi'_u \rfloor = \alpha\}$; $V_2^\alpha := \{u \in V_2 | \lceil \pi'_u \rceil - \pi'_u = \alpha\}$ and $V^\alpha := V_1^\alpha \cup V_1^\alpha$. For each $\alpha > 0$, $|V_2^\alpha| - |V_1^\alpha| \leq 0$. Indeed, for some sufficiently small $\epsilon > 0$, $\pi^\epsilon := \pi' - \epsilon(\chi^{V_2^\alpha} - \chi^{V_1^\alpha})$ is a dual feasible solution. So $\pi'(V_1 \cup V_2) \leq \pi^\epsilon(V_1 \cup V_2) = \pi'(V_1 \cup V_2) - \epsilon(|V_2^\alpha| - |V_1^\alpha|)$. And thus we get:

$$\tilde{\pi}(V_1 \cup V_2) = \pi'(V_1 \cup V_2) + \sum_{\alpha > 0, V^\alpha \neq \emptyset} \alpha(|V_2^\alpha| - |V_1^\alpha|) \leq \pi'(V_1 \cup V_2). \tag{41}$$

So $\tilde{\pi}$ is an integral dual optimal solution. □

So, when the weight function w is integral, the linear programming dual of (36) has an integral optimum solution.

A system of linear inequalities $Ax \leq b$ with the property that – like (36) – $\min\{y^\top b | y^\top A = w; y \geq 0\}$ is attained by an integral y for each integral objective function w for which the minimum exists, is called *totally dual integral*. Edmonds & Giles [1977] (and Hoffman [1974] for the special case of *pointed polyhedra*, i.e., polyhedra with extreme points) proved that a totally dual integral system $Ax \leq b$, with A and b integral, describes an integral polyhedron. (A polyhedron in $\mathbb{R}^n$is *integral* if it is the convex hull of vectors in $\mathbb{Z}^n$.) Thus, when G is bipartite, the fact that the minimization problem in Theorem 15 admits an integral optimal solution implies that $\text{Fract}(G) = \text{Match}(G)$.

The *perfect matching polytope* of a graph G is the convex hull $\text{Perfect}(G)$ of the characteristic vectors of perfect matchings in G. In the next section, when studying the weighted matching problem, we concentrate on perfect matchings. In the context of bipartite graphs the perfect matching polytope is often referred to as the *assignment polytope*. The following characterization of the assignment polytope follows easily from Theorem 14.

Corollary 16. *Let G be a bipartite graph. Then*

$$\text{Perfect}(G) = \{x \in \mathbb{R}_+^E \mid x(\delta(v)) = 1 \ (v \in V)\}. \tag{42}$$

Note that, unlike Theorem 14, there exist non-bipartite graphs G for which (42) holds.

Theorem 16 is probably best known in terms of doubly stochastic matrices and by the names of its re-inventors. A matrix $A = (a_{ij})$ is *doubly stochastic* if all its entries are non-negative and all its row and column sums are equal to one, i.e., $\sum_j a_{ij} = 1$ for each row i and $\sum_i a_{ij} = 1$ for each column j.

Theorem 17 [Birkhoff, 1946; Von Neumann, 1953]. *Each doubly stochastic matrix is a convex combination of permutation matrices.*

5.2. *Intermezzo: stable matchings*

Shapley & Shubik [1972] give the following economic interpretation of Theorem 15. Consider the vertices in V_1 and V_2 as disjoint sets of players and each edge uv in G as a possible coalition between u and v. If u and v form a coalition they may share out the *worth* w_{uv} of uv as payoffs π_u and π_v among themselves. Suppose that M is a matching of G (i.e. a collection of coalitions) and that $\pi_u (u \in V)$ is a corresponding collection of payoffs, i.e. $\pi_u + \pi_v = w_{uv}$ if $uv \in M$. If there exists an edge $uv \notin M$ such that $\pi_u + \pi_v < w_{uv}$, then the payoffs are *not stable for* M: u and v could increase their profits by breaking their respective coalitions and joining together. A matching is *payoff-stable* if there exists a collection of payoffs for M without such instability. By Theorem 15 and the complementary slackness conditions (35) payoff-stable matchings exist, they are exactly the maximum weight matchings. The optimal dual solutions compress all the possible payoffs without any instability.

Gale & Shapley [1962] introduced another notion of stability for matchings, the *stable marriage problem*. Suppose we have a marriage-market with n man and n women. Each person u has linear order $\prec_u$ on the persons of opposite sex, where $v \prec_u w$ means that u prefers to be married with w rather than with v. Modeling this on a bipartite graph, a collection of monogamous marriages is a matching. A perfect matching M between the men and women is *stable*, if for each $uv \notin M$, with $uu', vv' \in M$, either $v \prec_u u'$ or $u \prec_v v'$. In other words, a perfect matching is stable if no unmarried couple would prefer to get divorced and marry each other.

Gale & Shapley [1962] showed that a stable matching always exists. To see this, consider the following procedure in which each man starts proposing to his most preferred woman. If she rejects, he proposes to the next woman on his preference list and so on. Each woman keeps her best received proposal under consideration, rejecting all the other ones. When all women hold a proposal, the procedure stops, a stable matching has been obtained. Indeed, suppose there is unmarried couple u and v. If u did not propose to v he prefers his partner above v. If he did proposed to v, she rejected him, which means that she got a better proposal.

Instead of letting the above 'propose and reject' game decide on how the couples are made, there also could be a match-maker that arranges the marriages. His arrangement has to be stable, but additionally to the preferences he has a weight function on the possible pairs and he wishes to arrange a stable

matching with maximum weight. The following polyhedral characterization of stable matchings, due to Vande Vate [1989], shows that this maximal weight stable matching problem is a linear programming problem.

The convex hull of the characteristic vectors of stable matchings in a complete bipartite graph $G = (V_1 \cup V_2, E)$ is:

$$\Big\{x \in \text{Perfect}(G) \mid \sum_{v \prec_u w} x_{uw} + \sum_{u \prec_v w} x_{wv} + x_{uv} \geq 1 (u \in V_1, v \in V_2)\Big\}. \quad (43)$$

In fact, it turned out that many of the properties of stable matchings can be derived form this polyhedral result [see Roth, Rothblum and Vande Vate, 1993].

The stable matching problem can also be formulated for non-bipartite graphs. However, in that case, no stable matching might exist; a 4-node example is easily constructed. On the other hand, Irving [1985] derived a polynomial time algorithm that finds a stable matching if it exists. For further reading on stable matchings we refer to the book of Gusfield and Irving [1989].

5.3. *Non-bipartite graphs – Edmonds' matching polytope*

So, when G is not bipartite, $\text{Fract}(G) \neq \text{Match}(G)$. In trying to formulate the weighted matching problem in a non-bipartite graph G as a linear programming problem, we begin with the inequalities defining $\text{Fract}(G)$. Then, we search for inequalities that 'cut off' the fractional extreme points of $\text{Fract}(G)$. The following lemma characterizes the fractional extreme points of $\text{Fract}(G)$. Its proof is similar to that of Theorem 14.

Lemma 18 [Balinski, 1965]. *A vector $x \in \mathbb{R}^E$ is an extreme point of* $\text{Fract}(G)$ *if and only if there exists a matching M and a collection of odd circuits $C_1, \ldots, C_k$ in the graph G, such that the matching and the odd circuits are pairwise node-disjoint and*

$$x = \chi^M + \tfrac{1}{2}(\chi^{C_1} + \cdots + \chi^{C_k}). \quad (44)$$

Let $U \subseteq V(G)$, with $|U| \geq 3$ and odd. Add up all the inequalities $\sum_{e \in \delta(v)} x_e \leq 1$ with $v \in U$, and all inequalities $-x_e \leq 0$ with $e \in \delta(U)$. Dividing the resulting inequality by 2 yields

$$x(\langle U \rangle) = \tfrac{1}{2}\Big(\sum_{v \in U} x(\delta(v)) - \sum_{e \in \delta(U)} x_e\Big) \leq \tfrac{1}{2}|U|. \quad (45)$$

Obviously, each $x \in \text{Fract}(G)$ satisfies (45). Rounding down the right hand side we get the following *blossom constraint*

$$x(\langle U \rangle) \leq \tfrac{1}{2}(|U| - 1). \quad (46)$$

The characteristic vectors of matchings in G satisfy all the blossom constraints. However, the fractional extreme points of $\text{Fract}(G)$ do not. Indeed, the fractional extreme point x of $\text{Fract}(G)$ violates the blossom constraint obtained when U is chosen to be the node set of one of the odd circuits defining x. So, if we add all

the blossom constraints to the constraints defining Fract(G), we get a polyhedron Blossom(G), which contains Match(G), but is, for non-bipartite graphs, properly contained in Fract(G). In particular, the blossom constraints 'cut off' all the fractional extreme points of Fract(G). In the process, however, we might have introduced new fractional extreme points. Edmonds showed that adding the blossom constraints does not introduce any new fractional extreme points.

Theorem 19 [Edmonds, 1965b]. *For each graph G,* Match(G) = Blossom(G).

Proof. Edmonds [1965b] originally proved this result via his weighted matching algorithm (cf. Section 6.2). Since then, others including Balinski [1972], Hoffman & Oppenheim [1978], Lovász [1979a], Seymour [1979], Aráoz, Cunningham, Edmonds & Green-Krótki [1983] and Schrijver [1983b] have offered different proofs. We essentially follow the proof by Aráoz and coworkers and Schrijver.

Suppose that for some graph G, Match(G) $\neq$ Blossom(G). Among all such graphs, suppose $G = (V, E)$ has $|V| + |E|$ as small as possible. So G is connected and non-bipartite. Consider a fractional extreme point x of Blossom(G). Since no fractional extreme point of Fract(G) is in Blossom(G), x is not an extreme point of Fract(G). Hence, there exists a subset S of V with $|S| \geq 3$ and odd, such that

$$x(\langle S\rangle) = \tfrac{1}{2}(|S| - 1). \tag{47}$$

Among all such subsets, choose S so that $|S|$ is as small as possible.

Claim 1. $|S| < |V|$.

Proof of Claim 1. If not, $x(E) = \frac{1}{2}(|V| - 1)$ must be the only blossom constraint x satisfies with equality. Further, since $|V| + |E|$ is as small as possible, $x_e > 0$ for each $e \in E$. Otherwise, if $x_e = 0$ for some $e \in E$, $G \setminus e$ would be a smaller counterexample. Finally, since x is an extreme point of Blossom(G), x satisfies with equality at least $|E|$ constraints from the defining system. Hence there are at least $|E| - 1$ nodes u in $V(G)$ with $x(\delta(u)) = 1$. On the other hand, since $x(\delta(u)) > 0$ for each node u and

$$\sum_{v \in V}(1 - x(\delta(v))) = |V| - 2x(E) = 1, \tag{48}$$

there are at least two nodes u such that $x(\delta(u)) < 1$. Hence, $|V| - 1 \geq |E|$. But connected non-bipartite graphs have at least as many edges as nodes – contradiction! *End of proof of Claim 1.*

Partition x into $x = [x^1, x^2]$, where x^1 is the restriction of x to edges in $\langle S\rangle$. Consider the graph $G \times S$ obtained by shrinking S to the pseudo-node s.

Claim 2. $x^1 \in$ Blossom($G|S$) *and* $x^2 \in$ Blossom($G \times S$).

Proof of Claim 2. Since x is in Blossom(G), x^1 satisfies $x^1(\delta(v)) \leq 1$ for each $v \in S$ and the blossom constraints for $G|S$. So, $x^1 \in$ Blossom($G|S$). Likewise,

x^2 satisfies $x^2(\delta(v)) \leq 1$ for each $v \in V(G) \setminus S$ and the blossom constraint $x^2(\langle U\rangle) \leq \frac{1}{2}(|U| - 1)$ for each subset $U \subseteq V \setminus S$ with $|U| \geq 3$ and odd. Further, since $|S| \geq \sum_{v \in S} x(\delta(v)) = 2x(\langle S\rangle) + x(\delta(S))$ and $2x(\langle S\rangle) = |S| - 1$, $x^2(\delta(s)) = x(\delta(S)) \leq 1$. Finally, for each $U \subseteq V(G \times S)$ containing s,

$$x^2(\langle U\rangle) = x(\langle (U \setminus \{s\}) \cup S\rangle) - x(\langle S\rangle) \leq$$
$$\leq \tfrac{1}{2}(|(U \setminus \{s\}) \cup S| - 1) - \tfrac{1}{2}(|S| - 1) = \tfrac{1}{2}(|U| - 1)$$

and so $x^2 \in \text{Blossom}(G \times S)$. *End of proof of Claim 2.*

Since G is a smallest counterexample,

$$\text{Match}(G|S) = \text{Blossom}(G|S)$$
$$\text{and} \quad \text{Match}(G \times S) = \text{Blossom}(G \times S). \tag{49}$$

Hence, x^1 can be expressed as a convex combination of the characteristic vectors of matchings in $G|S$ and x^2 can be expressed as a convex combination of the characteristic vectors of matchings in $G \times S$. This implies that there is a non-negative integer k, matchings $M_1, \ldots, M_k$ in $G|S$, and matchings $N_1, \ldots, N_k$ in $G \times S$, such that

$$x_1 = \frac{1}{k}(\chi^{M_1} + \cdots + \chi^{M_k}) \quad \text{and} \quad x_2 = \frac{1}{k}(\chi^{N_1} + \cdots + \chi^{N_k}). \tag{50}$$

Note that the matchings M_i, and similarly the matchings N_i, need not all be distinct.

Claim 3. *We can renumber the matchings N_i $(i = 1, \ldots, k)$, so that $M_i \cup N_i$ is a matching in G for each $(i = 1, \ldots, k)$.*

Proof of Claim 3. By (47) each M_i has exactly one exposed node in $G|S$. Thus, we need only prove that for each $u \in S$: $|\{i \mid |M_i \cap \delta(u)| = 0\}| \geq |\{i \mid |N_i \cap \delta(u)| = 1\}|$. To see this, observe that:

$$|\{i \mid |M_i \cap \delta(u)| = 0\}| - |\{i \mid |N_i \cap \delta(u)| = 1\}|$$
$$= k - |\{i \mid |M_i \cap \delta(u)| = 1\}| - |\{i \mid |N_i \cap \delta(u)| = 1\}|$$
$$= k - \Big(\sum_{i=1}^{k} |M_i \cap \delta(u) \cap \langle S\rangle| + \sum_{i=1}^{k} |N_i \cap \delta(u) \cap \delta(S)|\Big)$$
$$= k - k\left(x^1(\delta(u) \cap \langle S\rangle) + x^2(\delta(u) \cap \delta(S))\right)$$
$$= k - kx(\delta(u)) \geq 0.$$

End of proof of Claim 3.

So, as $x = 1/k(\chi^{M_1 \cup N_1} + \cdots + \chi^{M_k \cup N_k})$, it is the convex combination of characteristic vectors of matchings; contradicting the assumption that it is a fractional extreme point of $\text{Blossom}(G)$. □

We can sharpen this result in the sense that we can specify which of the non-negativity, degree and blossom constraints are necessary to describe the

matching polytope of a given graph. Indeed, although we need all the non-negativity constraints, we do not need the *degree constraint* $x(\delta(v)) \leq 1$ for a node $v \in V(G)$ if there is another node u with $\delta(v) \subsetneq \delta(u)$ or there is an edge uw with $\delta(v) \subseteq \delta(u) \cup \delta(w)$. Moreover, we only need the blossom constraint $x(\langle S \rangle) \leq \frac{1}{2}(|S| - 1)$ for $S \subseteq V(G)$ such that $|S| \geq 3$ and odd, $G|S$ is factor-critical and $G|S$ has no cut node. (A node u is called a *cut node* of a connected graph G if $G \setminus u$ is not connected.) Pulleyblank & Edmonds [1974] showed that these are exactly the constraints we need in order to have a minimum system of linear inequalities defining the matching polytope. In geometric terms these constraints correspond to the 'facets' of the matching polytope [see Pulleyblank, 1989].

The fact that the dual problem in Theorem 15 has integral optimum solutions extends to non-bipartite graphs: the non-negativity, degree and blossom constraints form a totally dual integral system [Cunningham & Marsh, 1978; Hoffman & Oppenheim, 1978; Schrijver & Seymour, 1977; Schrijver, 1983a, b]. In fact, this remains true if we restrict ourselves to Pulleyblank and Edmonds' minimal description of the matching polytope [Cunningham & Marsh, 1978].

The following characterization of the perfect matching polytope follows easily from Theorem 19.

Corollary 20. *For each graph* $G = (V, E)$, Perfect(G) *is the solution set of the system:*

$$\begin{array}{lcll} x(\delta(v)) & = & 1 & (v \in V) \\ x(\langle U \rangle) & \leq & \frac{1}{2}(|U| - 1) & (U \subseteq V,\ |U| \text{ odd and at least } 3) \\ x_e & \geq & 0 & (e \in E), \end{array} \tag{51}$$

which is equivalent to the system

$$\begin{array}{lcll} x(\delta(v)) & = & 1 & (v \in V) \\ x(\delta(U)) & \geq & 1 & (U \subseteq V,\ |U| \text{ odd and at least } 3) \\ x_e & \geq & 0 & (e \in E). \end{array} \tag{52}$$

Proof. That (51) describes the perfect matching polytope is trivial. We need only prove that (51) and (52) are equivalent.

Consider $U \subseteq V$, with $|U|$ odd, and let $x \in \mathbb{R}^E$ be such that $x(\delta(v)) = 1$ for each $v \in V$. Then

$$\begin{aligned} x(\langle U \rangle) \leq \tfrac{1}{2}(|U| - 1) &\iff x(\langle U \rangle) - \tfrac{1}{2} \sum_{v \in U} x(\delta(v)) \leq \tfrac{1}{2}(|U| - 1) - \tfrac{1}{2}|U| \\ &\iff -\tfrac{1}{2} x(\delta(U)) \leq -\tfrac{1}{2} \\ &\iff x(\delta(U)) \geq 1. \end{aligned}$$

□

So, we have two descriptions of the perfect matching polytope. We call (51) the *blossom description* of the perfect matching polytope and (52) the *odd cut description*. The inequalities $x(\delta(U)) \geq 1$ in (52) are called the *odd cut constraints*.

Like the blossom description of the matching polytope, the blossom description of the perfect matching polytope is totally dual integral (the 'perfect matching case' follows directly from the 'matching case'). This is not the case for the odd cut description – the complete graph on 4 nodes, K_4, provides a counterexample. However, from the proof of Corollary 20 and the fact that (51) is totally dual integral, it can be shown that when the weight function is integral, the odd cut description admits half-integral dual optimal solutions.

The (perfect) matching polytope is a geometric object: namely the convex hull of points in a Euclidean space. There are many other interesting geometric objects related to matchings, e.g., the cone generated by the characteristic vectors of perfect matchings, the linear hull of the characteristic vectors of matchings, etc. Edmonds, Lovász & Pulleyblank [1982], Naddef [1982], Naddef & Pulleyblank [1982] have obtained results in this vein. Structural results on matchings like those mentioned in Section 4.1 ('ear-decomposition' and 'brick-decomposition') often play a crucial role in characterizing these geometric objects. An especially noteworthy example is Lovász's beautiful characterization of the matching lattice, i.e., the set $\{\sum_{M\in\mathcal{M}} \lambda_M \chi^M \mid \lambda_M \in \mathbb{Z} (M \in \mathcal{M})\}$ where $\mathcal{M}$ denotes the set of perfect matchings [Lovász, 1987; cf. Murty, 1994].

Karzanov [1992] derived a polynomial time algorithm for calculating the Euclidean distance from the origin to the perfect matching polytope of a bipartite graph.

In this section we have formulated matching problems as linear programming problems. Making combinatorial problems accessible to linear programming techniques in this way is one of the main goals of 'polyhedral combinatorics'. In general, this area could be described as the study of methods for solving combinatorial problems using the theory of linear inequalities. Over the years, the results in this section have been among the principal paradigms of this polyhedral approach. (However, even for matchings not all polyhedral questions have been resolved, see Cunningham & Green-Krotki [1986].) For surveys on polyhedral combinatorics, see Pulleyblank [1983, 1989] and Schrijver [1983a, 1995]. The standard reference for the theory of integer and linear programming – the toolbox for polyhedral combinatorics – is Schrijver's book [1986].

6. Finding maximum and minimum weight matchings

In this section we give polynomial time algorithms for finding a (perfect) matching of maximum or minimum weight. Actually we consider only the problem of finding a minimum weight perfect matching, but this is not really a restriction. Indeed, suppose we are given a non-negative weight function $w \in \mathbb{R}^E$ and want to find a maximum weight matching in a graph G. By adding nodes and edges with zero weight, we can transform the problem into one of finding a maximum weight matching in a complete graph with an even number of vertices, or if G is bipartite, in a complete bipartite graph in which the two color classes have the same number of nodes. Since each matching in these graphs is contained in a perfect matching, we may find a maximum weight matching in the original graph by finding a

maximum weight perfect matching in the complete (bipartite) graph. Replacing each weight w_e by $-w_e$, we turn the problem into a minimization problem and, if we prefer non-negative weights, we may add a suitable constant to each weight.

A weighted matching problem is a linear programming problem, so the duality theorem of linear programming (34) provides a stopping criterion. In fact, using the complementary slackness conditions, we reduce the weighted matching problem to a series of cardinality matching problems.

6.1. Bipartite graphs

Throughout this section $G = (V_1 \cup V_2, E)$ is a bipartite graph and $w \in \mathbb{R}_+^E$. For convenience, we assume that G contains a perfect matching. From Corollary 16 and linear programming duality (34) the minimum weight of a perfect matching is equal to the maximum in

$$\begin{array}{ll} \text{maximize} & \pi(V_1 \cup V_2) \\ \text{subject to} & \pi_u + \pi_v \leq w_{uv} \quad (uv \in E). \end{array} \tag{53}$$

We refer to (53) as the *dual* problem and to each feasible solution π to (53) as *dual feasible*. For each dual feasible solution π we define the *reduced cost function* $w^\pi \in \mathbb{R}^E$ by: $w^\pi_{uv} := w_{uv} - \pi_u - \pi_v$ for each $uv \in E$, and the graph G_π by $V(G_\pi) := V(G)$ and $E(G_\pi) := \{uv \in E(G) \mid w^\pi_{uv} = 0\}$. Thus, G_π is the subgraph on the nodes of G that includes only those edges, called *admissible*, with zero reduced cost under π. The edges not in $E(G_\pi)$ are called *inadmissible*.

The complementary slackness conditions (35) imply that:

A dual feasible solution π is optimal if and only if G_π admits a perfect matching. Moreover, if G_π admits a perfect matching, then the perfect matchings in G_π are exactly the minimum weight perfect matchings in G. (54)

So, given a dual feasible π, we check whether G_π has a perfect matching M. If it does, M is a minimum weight perfect matching in G and we are done. If it does not, we change π as follows.

DUAL CHANGE (in bipartite graphs): Let M be a maximum matching and F a Hungarian forest in G_π. Define π' by

$$\pi'_u := \begin{cases} \pi_u + \epsilon & \text{if } u \in V_1 \cap \text{even}(F) \\ \pi_u - \epsilon & \text{if } u \in V_2 \cap \text{odd}(F) \\ \pi_u & \text{otherwise,} \end{cases} \tag{55}$$

where

$$\epsilon := \min\{w^\pi_{uv} \mid u \in \text{even}(F) \cap V_1, v \in V_2 \setminus \text{odd}(F), uv \in E\}. \tag{56}$$

By (56) and because F is Hungarian, π' is dual feasible. Moreover,

M and F are contained in $G_{\pi'}$. (57)

So, we can apply GROW, starting with the matching M and the alternating forest F, to search for a maximum matching in $G_{\pi'}$.

Note that, since $uv \in E(G_{\pi'})$ for each $u \in V_1 \cap \text{even}(F)$ and $v \in V_2 \setminus \text{odd}(F)$ with $w^{\pi}_{uv} = \epsilon$,

$$F \text{ is not Hungarian in } G_{\pi'}. \tag{58}$$

Thus, we have the following algorithm for finding a minimum weight perfect matching in a bipartite graph. We begin with the dual feasible solution π with $\pi_v = 0$ for each $v \in V_1 \cup V_2$.

We apply GROW and AUGMENT until we find either a perfect matching M or a Hungarian alternating forest F in G_π. If we find a perfect matching M in G_π, we are done: M is a minimum weight perfect matching in G. Otherwise, we apply DUAL CHANGE. Clearly this algorithm, called the *weighted Hungarian method*, runs in polynomial time. It was originally introduced by Kuhn [1955, 1956]. For variants of his method, see Flood [1956], Ford & Fulkerson [1957], Motzkin [1956] and Munkres [1957]. Bertsekas [1979] proposed a so-called auction method [cf. Bertsekas, 1990].

Kuhn's algorithm is a 'dual' algorithm in the sense that at any stage it keeps a dual feasible solution and a primal infeasible solution (namely a non-perfect matching) satisfying complementary slackness. Primal feasibility, i.e. the matching being perfect, acts as the stopping criterion. Another possible approach is the 'primal' algorithm of Balinski & Gomory [1964]. It keeps at any stage a perfect matching which is changed until it becomes optimal. The stopping criterion in their algorithm is dual feasibility.

Note that although the weighted Hungarian method was motivated by Theorem 14, its correctness does not rely on that result. In fact, the algorithm provides a separate proof of Theorem 14 as well as of Theorem 15 (the initial dual feasible solution is integral and, when the weights are integral, each dual change maintains integrality).

6.1.1. Implementation of the weighted Hungarian method

After $|V_1|$ augmentations, the weighted Hungarian method finds a perfect matching M in G_π and, by the complementary slackness conditions (35), M is a minimum weight perfect matching in G. Thus, the running time of the algorithm depends on the computations required between consecutive augmentations, called a *phase*.

Note that if an admissible edge becomes inadmissible after a dual change, it cannot be come admissible again until after the next augmentation. This means that if we ignore the effort required to make the dual changes, each phase of the algorithm is essentially an application of GROW. One could say that the 'dual changer' confounds the 'grower': any time the alternating forest becomes Hungarian, the 'dual changer' adjusts the graph so that the forest is no longer Hungarian. Consequently, if we disregard the effort spent on dual changes, each phase can be carried out in $O(|E|)$ time. We sketch two implementations of the dual changes; one for dense graphs and one for sparse graphs.

Dual changes in dense graphs: In dense graphs, where $|V|^2 = O(|E|)$, we maintain an array CLOSE so that for each $v \in V_2$, $\text{CLOSE}[v] = u$, where $u \in V_1 \cap \text{even}(F)$ and $w^\pi_{uv} = \min\{w^\pi_{u'v} \mid u' \in V_1 \cap \text{even}(F)\}$. Using CLOSE, we can make each dual change in $O(|V|)$ time and, each time we add a node to $V_1 \cap \text{even}(F)$, we can update CLOSE in $O(|V|)$ time. During a phase we make at most $|V_1|$ dual changes and add at most $|V_1|$ nodes to $V_1 \cap \text{even}(F)$. So, with this implementation we can carry out the dual changes for each phase in $O(|V|^2)$ time.

The weighted Hungarian method can be implemented to run in $O(|V|^3)$ time. (59)

Dual changes in sparse graphs: In sparse graphs, where $|E|$ is significantly smaller than $|V_1|^2$, we can improve the running time by implementing the dual changes more efficiently. First, we concentrate on finding the value of ϵ. For each node v in $V_2 \setminus \text{odd}(F)$, we maintain the value $\text{SLACK}[v] := w^\pi_{\text{CLOSE}[v]v}$. Each time we add a node u to $V_1 \cap \text{even}(F)$, we scan each of its $\deg(u)$ neighbors v to update CLOSE and SLACK. So, during an entire phase we require only $O(|E|)$ time to maintain these two arrays.

Standard data structures such as 'd-heaps' or 'priority queues' [see Tarjan, 1983, or Aho, Hopcroft & Ullman, 1974] for storing $V_2 \setminus \text{odd}(F)$ according to the entries in SLACK facilitate quick determination of ϵ. Using such a data structure we can not only find ϵ, but also update the data structure itself in $O(\log |V|)$ time whenever SLACK changes or a node leaves $V_2 \setminus \text{odd}(F)$. So, it is possible to determine the values of ϵ in $O(|E| \log |V|)$ time per phase.

Instead of making dual changes explicitly, which can take up to $|V|$ steps each, we make them implicitly. We maintain a variable ϵ_{total} and, for each node $u \in V$, we keep two variables: π^{old}_u and π^{cor}_u. Together these represent the dual variable π_u according to the following rule:

$$\pi_u := \begin{cases} \pi^{old}_u + (\epsilon_{total} - \pi^{cor}_u) & \text{if } u \in V_1 \cap \text{even}(F) \\ \pi^{old}_u - (\epsilon_{total} - \pi^{cor}_u) & \text{if } u \in V_2 \cap \text{odd}(F) \\ \pi^{old}_u & \text{otherwise} \end{cases} \tag{60}$$

To make a dual change implicitly, we replace ϵ_{total} by $\epsilon_{total} + \epsilon$. When a node u enters the alternating forest, we set $\pi^{cor}_u := \epsilon_{total}$; and when an augmentation leads us to remove the node u from the alternating forest we set

$$\pi^{old}_u := \begin{cases} \pi^{old}_u + (\epsilon_{total} - \pi^{cor}_u) & \text{if } u \in \text{even}(F) \\ \pi^{old}_u - (\epsilon_{total} - \pi^{cor}_u) & \text{if } u \in \text{odd}(F). \end{cases}$$

Clearly, this 'delayed' revision can be carried out within GROW and AUGMENT.

Thus, we have the following result:

The weighted Hungarian method can be implemented to run in $O(|E||V| \log |V|)$ time. (61)

Fredman & Tarjan [1987] improved this time bound to $O(|V|(|E| + |V| \log |V|))$ using 'Fibonacci-heaps'. Brezovec, Cornuéjols, & Glover [1988] obtained the same time bound based on algorithm for a special case of matroid intersection.

6.2. Non-bipartite graphs

In this section we consider the problem of finding a minimum weight perfect matching in a non-bipartite graph G. For convenience, we assume that G admits a perfect matching and that the weights $w \in \mathbb{R}^E$ are non-negative.

By Corollary 20, a minimum weight perfect matching solves the linear programming problem

$$\begin{array}{lllll} \text{maximize} & \sum_{e \in E} w_e x_e & & & \\ \text{subject to} & x(\delta(v)) & = & 1 & (v \in V) \\ & x(\delta(S)) & \geq & 1 & (S \in \Omega(G);\ |S| \neq 1) \\ & x_e & \geq & 0 & (e \in E), \end{array} \tag{62}$$

where $\Omega(G) := \{S \subseteq V(G) \mid |S| \text{ is odd}\}$. Thus, by linear programming duality (34), the minimum weight of a perfect matching is equal to the maximum in

$$\begin{array}{lllll} \text{maximize} & \sum_{S \in \Omega(G)} \pi_S & & & \\ \text{subject to} & \sum_{S \in \Omega(G);\delta(S) \ni e} \pi_S & \leq & w_e & (e \in E) \\ & \pi_S & \geq & 0 & (S \in \Omega(G);\ |S| \neq 1). \end{array} \tag{63}$$

We refer to (63) as the *dual* problem and to each feasible solution π to (63) as *dual feasible*. For each dual feasible solution π the *reduced cost function* $w^\pi \in \mathbb{R}^E$ is defined by: $w^\pi_e := w_e - \sum_{S \in \Omega(G);\delta(S) \ni e} \pi_S$ for each $e \in E$, and the graph G_π on the node set $V(G_\pi) := V(G)$ has edge set defined by $E(G_\pi) := \{uv \in E(G) \mid w^\pi_{uv} = 0\}$. So, again, G_π is the subgraph on the nodes of G that includes only those edges, called *admissible*, with zero reduced cost under π. Finally, we define $\Omega_\pi := \Omega_\pi(G) := \{S \in \Omega(G) \mid \pi_S > 0 \text{ and } |S| \neq 1\}$.

The complementary slackness conditions (35) imply the following characterization of minimum weight perfect matchings:

> *A dual feasible solution π is optimal if and only if G_π admits a perfect matching M such that $M \cap \delta(S) = 1$ for each $S \in \Omega_\pi(G)$. If π is optimal, the collection of all such perfect matchings in G_π is exactly the collection of minimum weight perfect matchings in G.* (64)

So minimum weight perfect matchings are perfect matchings in G_π, for some optimal π, that satisfy additional conditions. These additional conditions prevent us from finding a minimum weight perfect matching in G by simply searching for a maximum cardinality matching in G_π as we did in the case of bipartite graphs. To overcome this, we restrict attention to those dual feasible solutions π, called *structured*, that satisfy the following two conditions:

$$\Omega_\pi \text{ is } \textit{nested}, \text{i.e., if } S, T \in \Omega_\pi, \text{ then } S \subseteq T,\ T \subseteq S \text{ or } S \cap T = \emptyset. \tag{65}$$

If $S \in \Omega_\pi$, and $S_1, \ldots, S_k$ are the inclusion-wise maximal members of Ω_π properly contained in S, then $(G_\pi \times S_1 \times \cdots \times S_k)|S$ is factor-critical. (66)

Note that by (28), (66) implies that $G_\pi|S$ and $G|S$ are also factor-critical.

For each structured dual feasible π we define $\widetilde{G_\pi}$ to be the graph obtained from G_π by shrinking the members of Ω_π. The motivation for considering only structured dual solutions is apparent from the following consequence of (64) and (29):

A structured dual feasible solution π is optimal if and only if $\widetilde{G_\pi}$ admits a perfect matching. (67)

This suggests the following algorithm, developed by Edmonds [1965b], for finding a minimum weight perfect matching in a non-bipartite graph G.

EDMONDS' ALGORITHM: Given a structured dual feasible π – initially identical to 0 – construct $\widetilde{G_\pi}$. If $\widetilde{G_\pi}$ admits a perfect matching M, then the dual feasible solution π is optimal; extend M to a minimum weight perfect matching in G_π. Otherwise, determine the Edmonds–Gallai structure of $\widetilde{G_\pi}$ and revise the dual solution according to (68) below.

We rely on notation similar to that in Section 2.2 to describe the relation between G and $\widetilde{G_\pi}$. If $S \in \Omega_\pi(G)$ is shrunk into pseudo-node $s \in \widetilde{G_\pi}$, we define $\text{DEEP}_\pi[s] = S$ and $\text{OUTER}_\pi[u] = s$ for each $u \in S$. For each node u in G that is also a node in $\widetilde{G_\pi}$, we define $\text{DEEP}_\pi[u] = \{u\}$ and $\text{OUTER}_\pi[u] = u$. For each $T \subseteq V(\widetilde{G_\pi})$, $\text{DEEP}_\pi[T] := \cup_{s \in T} \text{DEEP}_\pi[s]$.

DUAL CHANGE (in non-bipartite graphs): Given a structured dual feasible solution π, define

$$\pi'_S := \begin{cases} \pi_S + \epsilon & \text{if } S = \text{DEEP}_\pi[D] \text{ for some component } D \text{ of } D(\widetilde{G_\pi}) \\ \pi_S - \epsilon & \text{if } S = \text{DEEP}_\pi[s] \text{ for some } s \in A(\widetilde{G_\pi}) \\ \pi_S & \text{otherwise,} \end{cases} \tag{68}$$

where

$$\begin{aligned} \epsilon &:= \min\{\epsilon_1, \tfrac{1}{2}\epsilon_2, \epsilon_3\}, \text{ and} \\ \epsilon_1 &:= \min\{w^\pi_{uv} \mid \text{OUTER}_\pi[u] \in D(\widetilde{G_\pi}); \text{OUTER}_\pi[v] \in C(\widetilde{G_\pi})\}; \\ \epsilon_2 &:= \min\{w^\pi_{uv} \mid \text{OUTER}_\pi[u] \text{ and } \text{OUTER}_\pi[v] \\ &\qquad \text{in different components of } D(\widetilde{G_\pi})\}; \\ \epsilon_3 &:= \min\{\pi_S \mid S = \text{DEEP}_\pi[s] \text{ for some } s \in A(\widetilde{G_\pi}), \\ &\qquad |\text{DEEP}_\pi[s]| \neq 1\}. \end{aligned} \tag{69}$$

Determining whether or not $\widetilde{G_\pi}$ has a perfect matching, computing the Edmonds–Gallai structure of $\widetilde{G_\pi}$, and extending a perfect matching in $\widetilde{G_\pi}$ to a perfect matching in G_π can all be accomplished via any maximum cardinality algorithm that determines the Edmonds–Gallai structure, like the blossom algorithm or

the algorithm in Section 8.4. A perfect matching in G_π obtained by extending a perfect matching in $\widetilde{G_\pi}$ satisfies the complementary slackness conditions (64) and hence is a minimum weight perfect matching in G. If $\widetilde{G_\pi}$ does not admit a perfect matching, DUAL CHANGE increases the dual objective function value by $\epsilon(\kappa(\widetilde{G_\pi}) - |A(\widetilde{G_\pi})|) = \epsilon \operatorname{def}(\widetilde{G_\pi}) > 0$. So Edmonds' algorithm only stops, when a minimum weight perfect matching has been obtained. That the algorithm does stop follows from the following lemma.

Lemma 21. *Given a structured dual feasible solution π,* DUAL CHANGE *yields a structured dual feasible solution π', such that:*

- $\operatorname{def}(\widetilde{G_{\pi'}}) \leq \operatorname{def}(\widetilde{G_\pi})$;
- *if* $\operatorname{def}(\widetilde{G_{\pi'}}) = \operatorname{def}(\widetilde{G_\pi})$, *then* $\text{DEEP}_{\pi'}(D(\widetilde{G_{\pi'}})) \supseteq \text{DEEP}_\pi(D(\widetilde{G_\pi}))$;
- *if* $\operatorname{def}(\widetilde{G_{\pi'}}) = \operatorname{def}(\widetilde{G_\pi})$ *and* $\text{DEEP}_{\pi'}(D(\widetilde{G_{\pi'}})) = \text{DEEP}_\pi(D(\widetilde{G_\pi}))$, *then* $\text{DEEP}_{\pi'}(C(\widetilde{G_{\pi'}})) \supsetneq \text{DEEP}_\pi(C(\widetilde{G_\pi}))$.

Proof. For each component D of $D(\widetilde{G_\pi})$, the sets in Ω_π are either disjoint from $\text{DEEP}_\pi[D]$ or contained in $\text{DEEP}_\pi[D]$. So $\Omega_{\pi'}$ is nested. Moreover, by (27) $\widetilde{G_\pi}|D$ is factor-critical, so π' satisfies (66) and hence is structured.

To prove the remainder of the lemma, observe that $\widetilde{G_{\pi'}}$ can be obtained from $\widetilde{G_\pi}$ in two steps. First, shrink the node-sets S that are not in Ω_π but are in $\Omega_{\pi'}$, this yields $\widetilde{G_\pi}^*$ (i.e. the Edmonds–Gallai graph of $\widetilde{G_\pi}$). The nodes in $D(\widetilde{G_\pi}^*)$ and in $C(\widetilde{G_\pi}^*)$ are contained in $V(\widetilde{G_{\pi'}})$, Hence:

$$\begin{aligned}
&- \operatorname{def}(\widetilde{G_\pi}^*) = \operatorname{def}(\widetilde{G_\pi});\\
&- \text{DEEP}_{\pi'}(D(\widetilde{G_\pi}^*)) = \text{DEEP}_\pi(D(\widetilde{G_\pi}));\\
&- \text{DEEP}_{\pi'}(C(\widetilde{G_\pi}^*)) = \text{DEEP}_\pi(C(\widetilde{G_\pi}))
\end{aligned} \tag{70}$$

$\widetilde{G_{\pi'}}$ can be obtained from $\widetilde{G_{\pi'}}^*$ by applying the operation (31) if $\epsilon = \epsilon_1$ or $\frac{1}{2}\epsilon_2$, the operation in (32) if $\epsilon = \epsilon_3$ and the operation (30). So, by (30), (31) and (32):

$$\begin{aligned}
&- \operatorname{def}(\widetilde{G_{\pi'}}) \leq \operatorname{def}(\widetilde{G_\pi}^*);\\
&- \textit{if } \operatorname{def}(\widetilde{G_{\pi'}}) = \operatorname{def}(\widetilde{G_\pi}^*) \textit{ then } D(\widetilde{G_{\pi'}}) \supseteq D(\widetilde{G_\pi}^*);\\
&- \textit{if both } \operatorname{def}(\widetilde{G_{\pi'}}) = \operatorname{def}(\widetilde{G_\pi}^*) \textit{ and } D(\widetilde{G_{\pi'}}) = D(\widetilde{G_\pi}^*),\\
&\quad \textit{then } C(\widetilde{G_{\pi'}}) \supsetneq C(\widetilde{G_\pi}^*).
\end{aligned} \tag{71}$$

From (71) and (70), the lemma follows. □

As a consequence, there are at most $O(|V(G)|^3)$ dual changes. Since we can find a maximum cardinality matching and the Edmonds–Gallai structure of $\widetilde{G_\pi}$ in polynomial time,

Edmonds' algorithm finds a minimum weight perfect matching in polynomial time. (72)

Note that the algorithm provides a constructive proof of Corollary 20.

6.2.1. Implementing Edmonds' algorithm

In implementing Edmonds' algorithm for finding a minimum weight perfect matching in a non-bipartite graph, we can exploit the efficient algorithms discussed in Section 2.2 for finding a maximum cardinality matching. Note, however, that unlike the cardinality problem, in solving the weighted problem we must be able to expand a pseudo-node without expanding the pseudo-nodes contained in it. We can similarly exploit the efficient methods discussed in Section 6.1 for revising the dual solution but the continual shrinking and expanding of blossoms gives rise to certain complications [see Galil, Micali & Gabow, 1986].

Lawler [1976] developed an $O(|V|^3)$ implementation of Edmonds' algorithm. Galil, Micali & Gabow [1986] derived an $O(|E||V|\log|V|)$ algorithm. Gabow, Galil & Spencer [1989] derived an implementation that runs in $O(|V|(|E|\log_2\log_2 \log_{\max\{|E|/|V|,2\}}|V| + |V|\log|V|))$ time. This, in turn, has been improved by Gabow's $O(|V|(|E| + |V|\log|V|))$ bound [Gabow, 1990]. Nice reviews of these implementations are Ball & Derigs [1983] and Galil [1986a].

Gabow & Tarjan [1991] and Gabow [1985] have developed algorithms whose running times depend on the edge weights. These algorithms require $O(\sqrt{|V|}\alpha(|V|,|E|)\log|V||E|\ \log(|V|N))$ and $O(|V|^{3/4}|E|\log N)$ time, respectively, where N is an upper bound on the edge weights.

These running times can be further improved when we confine ourselves to restricted classes of weighted matching problems. Lipton & Tarjan [1980] derived an $O(|V|^{3/2}\log|V|)$ algorithm for matching in planar graphs. This algorithm is based on their *Separator theorem* for planar graphs: If $G = (V, E)$ is planar we can partition V into three sets A, B and C with $|A|, |B| \leq \frac{2}{3}|V|$ and $|C| \leq 2\sqrt{2|V|}$ such that no edge connects A with B [Lipton & Tarjan, 1979]. The *separator* C can be found in linear time and can be used to recursively decompose a matching problem in a planar graph into matching problems in smaller planar graphs.

Vaidya [1989] showed that *Euclidean matching problems*, in which the nodes are given as points in the plane and the weight of an edge between the two points u and v is the distance between the two points in the plane, can be solved in $O(|V|^{5/2}(\log|V|)^4)$ time. When the points lie on a convex polygon, a minimum weight matching can be found in $O(|V|\log|V|)$ time [Marcotte & Suri, 1991].

7. General degree constraints

Matching can be viewed as a 'degree-constrained subgraph problem': find a maximum cardinality, or maximum weight, subgraph in which each node has degree at most one. In this section we consider more general degree constraints.

Let $G = (V, E)$ be an undirected graph, possibly with loops. The collection of loops incident to node v is denoted by $\lambda(v)$. The *general matching problem* is: Given edge weights $w \in \mathbb{R}^E$, edge capacities $c \in (\mathbb{R} \cup \{\infty\})^E$ and degree bounds $a, b \in (\mathbb{R} \cup \{\infty\})^V$ find a minimum or maximum weight integral vector x

satisfying:

$$\begin{array}{rcccl} a_v & \leq & x(\delta(v)) + 2x(\lambda(v)) & \leq & b_v \quad (v \in V) \\ 0 & \leq & x_e & \leq & c_e \quad (e \in E). \end{array} \tag{73}$$

We call an integral vector x satisfying (73) a *general matching*. We call a the *degree lower bounds*, b the *degree upper bounds* and c the *capacities*. The corresponding constraints are called the *lower* and *upper degree constraints* and the *capacity constraints*. We did not impose more general lower bounds on the values of x_e since this does not yield a more general problem. (Given a lower bound $d \neq 0$ on the edges, replace each degree lower bound a_v by $a_v - d(\delta(v)) - 2d(\lambda(v))$, each degree upper bound b_v by $b_v - d(\delta(v)) - 2d(\lambda(v))$, each capacity c_e by $c_e - d_e$, and each edge variable x_e by $x_e - d_e$.)

In addition to matching and perfect matching, the general matching problem includes: the *simple b-matching problem* in which $a = 0$, b is arbitrary and $c = 1$; the *b-matching problem* in which $a = 0$, b is arbitrary and $c = \infty$; the *capacitated b-matching problem* in which $a = 0$ and b and c are arbitrary; and the *edge cover problem* in which $a = 1$, $b = \infty$, and $c = 1$. The general matching problem also includes the *perfect* versions of the (capacitated) b-matching problems in which $a = b$ and the simple perfect b-matching problem, also called the *b-factor problem*.

We can use loops to express the degree constraints as parity conditions. For instance, to force the degree of a node v to be an odd number between 3 and 11 we add a loop ℓ to v and impose the constraints: $x(\delta(v)) + 2x_\ell = 11$; $0 \leq x_\ell \leq 4$. We discuss parity conditions in Section 7.4.

An even more general degree-constrained subgraph problem is the *D-matching problem*: Given a set $D_v \subseteq \mathbb{Z}_+^E$ for each $v \in V$, find a subgraph G' of G with $\deg_{G'}(v) \in D_v$ for each $v \in V$. Lovász [1972a] proved that finding a D-matching is NP-complete, even when D_v is restricted to be either $\{1\}$ or $\{0, 3\}$. When for each $v \in V$, $\mathbb{Z}_+ \setminus D_v$ contains no consecutive integers, the D-matching problem is polynomially solvable [Lovász, 1973; Cornuéjols, 1988; and Sebö, 1993].

Related to the degree-constrained subgraph problem is the question: For which $d \in \mathbb{Z}_+^{V(G)}$ has (73) an integral solution x with $x(\delta(v)) = d_v$ for all $v \in V(G)$? A polyhedral answer to this question has been given by Cunningham & Green-Krótki [1991], generalizing results by Balas & Pulleyblank [1983, 1989], who solved cases $a = 0$, $b = 1$ and $c = 1$, and by Koren [1973]. Koren considered the special instances of the question that G is the complete graph, $a = 0$, $b = \infty$ and $c = 1$, in other words he derived a system of inequalities for the convex hull of all the degree-sequences of simple graphs on $V(G)$ (see also Peled & Srinivasan [1989] and Cunningham & Zhang [1992]). The inequalities in this system are exactly the well-known necessary and sufficient conditions derived by Erdös & Gallai [1960] for a sequence of integers to be the sequence of degrees of a simple graph. For a separation algorithm (cf. Section 8.3) for the polyhedron given by Balas & Pulleyblank [1989] see Cunningham & Green-Krótki [1994].

For other generalizations of matchings see: Cornuéjols & Hartvigsen [1986], Cornuéjols, Hartvigsen & Pulleyblank [1982], Giles [1982a, b, c], and Lovász [1970b]. In this chapter we restrict attention to general matchings as defined in (73).

7.1. Reducing the general matching problem

One aspect of the self-refining nature of matching theory is that the general matching problem not only includes matching as a special case, but can also be reduced to the matching problem. There are two main steps in the reduction of general matching to matching. First, one reduces the general matching problem to a perfect b-matching problem in a new graph with no loops and no capacity constraints. Second, one further reduces the perfect b-matching problem to a perfect matching problem. The reductions are due to Tutte [1954].

In view of these reductions it is reasonable to expect that results analogous to those discussed earlier in this chapter extend to general matching. Indeed, this is the case. Tutte [1952, 1974] generalized his perfect matching theorem (Theorem 11) to give necessary and sufficient conditions for the existence of an f-factor (see also Tutte [1981] and, for an algorithmic proof, Anstee [1985]). Lovász [1970a] further generalized this result to *(f, g)-factors* or general matchings with $a_v = \deg(v) - g_v$ $(v \in V)$, $b = f$ and $c = 1$. Lovász also generalized the Edmonds–Gallai structure theorem to a structure theorem for (f, g)-factors [Lovász, 1970a, c, 1972a] and f-factors [Lovász, 1972e]. For a discussion of these latter results, see Lovász & Plummer [1986].

The polyhedral results and polynomial-time solvability of matching also extend to the general matching problem. After explaining the reductions from general matching problems to the (perfect) matching problem, we first consider polyhedral results for general matchings and next deal with algorithmic issues. It is possible to derive general matching results via the reductions [see Aráoz, Cunningham, Edmonds & Green-Krótki, 1983], but direct proofs are typically simpler.

In explaining the reductions we only show how the new graphs should be constructed and what the new bounds on the degrees should be. We leave it to the reader to determine appropriate weights on the edges and to prove that indeed the original problem can be solved by solving the newly constructed problem.

7.1.1. Reduction to perfect b-matching

We first show how to transform the general matching problem in $G = (V, E)$ to a perfect b-matching problem in a graph with no loops and no capacity constraints.

Reduction to $c_{uv} < \infty$ for each edge uv and $b_v < \infty$ for each node v: First, replace for each edge uv the capacity c_{uv} with $\min\{c_{uv}, b_u, b_v\}$ or, if this minimum is infinite, with $\max\{a_u, a_v\}$. With these new, finite, capacities, replace for each node v the degree upper bound b_v with $\min\{b_v, c(\delta(v)) + 2c(\lambda(v))\}$.

Reduction to $a_v = b_v$ for each node v: Next, construct a new graph G' as follows. Make two copies G_1 and G_2 of G. For each node v in G add an edge v_1v_2 with capacity $c_{v_1v_2} := b_v - a_v$ between the copies v_1 in G_1 and v_2 in G_2 of v. A copy (in G_1 or G_2) of an edge e in G gets the same capacity as e. For each node v in G the degree bounds of its two copies v_1 and v_2 in G' are: $a_{v_1} := b_{v_1} := a_{v_2} := b_{v_2} := b_v$.

Reduction to a loopless graph with $c \equiv \infty$: Finally, replace each edge $e = uv$ in G' with two new nodes, u_e and v_e, and three new edges, uu_e, u_ev_e, and v_ev. The capacities of these new edges are infinite and the degree bounds of the new nodes are: $a_{u_e} := a_{v_e} := b_{u_e} := b_{v_e} := c_{uv}$.

7.1.2. Reduction to perfect matching

Now, we further reduce the perfect b-matching problem in the loopless graph G' to a perfect matching problem in a graph G''. Replace each node v in G' with b_v copies $v_1, v_2, \ldots, v_{b_v}$. Replace each edge uv in G' by a collection of edges in G'', namely one between each copy u_i of u and each copy v_j of v. The b-matching problem in G' is now a perfect matching problem in the new graph G''.

7.2. General matching polyhedra

As polyhedral results for bipartite graphs are easier, we consider them first.

7.2.1. Bipartite graphs

Theorem 22. *The polyhedron described by (73) is integral for all integral vectors a, b and c if and only if G is bipartite. (Note that bipartiteness excludes loops.)*

Proof. Rather than derive this result by combining Corollary 16 with the above reductions, we present a proof based on the well-known result of Hoffman & Kruskal [1956] on totally unimodular matrices. An $m \times n$-matrix A is called *totally unimodular* if each square submatrix of A has determinant equal to 0 or ± 1. The following is easy to prove:

> *The node-edge incidence matrix of a graph G is totally unimodular if and only if G is bipartite.* (74)

Hence, the theorem follows from:

> *Given an $m \times n$-matrix A, the polyhedron $\{x \in \mathbb{R}^n \mid a \leq Ax \leq b;\ 0 \leq x \leq c\}$ is integral for each $a, b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^n$ if and only if A is totally unimodular* [Hoffman & Kruskal, 1956]. (75)

□

7.2.2. Network flows and bidirected graphs

Given a directed graph $D = (V(D), A(D))$, $a, b \in \mathbb{Z}^{V(D)}$ and $c \in \mathbb{Z}^{A(D)}$ a *general network flow* is a vector x satisfying:

$$\begin{array}{rcccll} a_v & \leq & x(\delta^+(v)) - x(\delta^-(v)) & \leq & b_v & (v \in V(D)) \\ 0 & \leq & x_a & \leq & c_a & (a \in A(D)). \end{array} \tag{76}$$

That (76) defines an integral polyhedron follows from network flow theory as well as from Hoffman and Kruskal's theorem (75). By a construction similar

to that used in Section 3.1 to reduce bipartite matching problems to max-flow problems, this in turn implies Theorem 22.

The general network flow problem and the general matching problem are similar. Both are constraint by a system of the form $a \leq Ax \leq b; 0 \leq x \leq c$, where each column of A has at most two non-zero coefficients. In the matching case both non-zero coefficients are 1, whereas in the network flow case one is 1 and the other is -1. The other difference is that in the matching case we also allow columns with a single coefficient of 2 as the only non-zero entry. Edmonds & Johnson [1970] proposed a common generalization of these two models: *the general matching problem for bidirected graphs*. A *bidirected graph* is a matrix A in which each column either contains two non-zero entries both ± 1 or contains a single non-zero entry equal to ± 1 or ± 2. A general matching in a bidirected graph A is an integral vector x satisfying: $a \leq Ax \leq b; 0 \leq x \leq c$. The results in this section also hold for these more general objects [Edmonds & Johnson, 1970].

7.2.3. Non-bipartite graphs

We derived the matching polytope (45) and (46) of a non-bipartite graph from the degree constraints by adding constraints obtained in the following manner. Add up degree and non-negativity constraints so that the coefficients in the left hand side of the resulting inequality are even, divide the resulting inequality by 2 and round the right hand side down to the nearest integer. Applying the same construction to (73) yields the inequalities:

$$x(\langle V_1\rangle) - x(\langle V_2\rangle) + x(F_1) - x(F_2^*) \leq \left\lfloor \tfrac{1}{2}(b(V_1) - a(V_2) + c(F_1 \cup F_2)) \right\rfloor$$
for each pair V_1, V_2 of disjoint subsets of V, each $F_1 \subseteq \delta(V_1) \setminus \delta(V_2)$, and each partition F_2, F_2^* of $\delta(V_2) \setminus \delta(V_1)$. (77)

In fact these inequalities, also called *blossom constraints*, describe the convex hull of general matchings. This can be derived from Theorem 19 or Corollary 20 via the above reductions [see Cook, 1983b; Schrijver, 1983a].

Theorem 23 [Edmonds, 1965b; Edmonds & Johnson, 1970]. *For each graph $G = (V, E)$, $a, b \in \mathbb{Z}^V$ and $c \in \mathbb{Z}^E$, the convex hull of all integral solutions to (73) is the solution set of the system of inequalities defined by (73) and (77). Moreover, this system is totally dual integral.*

Note that many of the inequalities (77) are redundant, e.g., when $b(V_1) - a(V_2) + c(F_1 \cup F_2)$ is even (though this is not the only case!). Although restricting the formulation to the inequalities (77) with $b(V_1) - a(V_2) + c(F_1 \cup F_2)$ odd gives a description of the convex hull of general matchings, we can no longer be assured that the system is totally dual integral. So, unlike ordinary matchings, the systems that are non-redundant and those that are minimally totally dual integral are distinct [see Cook & Pulleyblank, 1987; Cook, 1983a, b; Pulleyblank, 1980, 1981].

Below we list consequences of Theorem 23 for some of the more prominent special cases of general matching.

The b-matching polytope: The convex hull of all b-matchings is given by:

$$\begin{array}{lcll} x(\delta(v)) & \leq & b_v & (v \in V) \\ x(\langle U \rangle) & \leq & \lfloor \frac{1}{2} b(U) \rfloor & (U \subseteq V) \\ x_e & \geq & 0 & (e \in E) \end{array} \tag{78}$$

(Edmonds [1965b], see Hoffman & Oppenheim [1978] and Schrijver & Seymour [1977] for alternative proofs).

If we replace the constraints $x(\delta(v)) \leq b_v$ with the constraints $x(\delta(v)) = b_v$, we get the convex hull of perfect b-matchings. In the case of perfect b-matchings, as with perfect matchings, we can replace the blossom constraints with the odd cut constraints to get the following description of the convex hull of perfect b-matchings:

$$\begin{array}{lcll} x(\delta(v)) & = & b_v & (v \in V) \\ x(\delta(U)) & \geq & 1 & (U \subseteq V \text{ with } b(U) \text{ odd}) \\ x_e & \geq & 0 & (e \in E). \end{array} \tag{79}$$

When all components of b are even, the b-matching polytope is described by the degree and the non-negativity constraints, regardless of whether the graph is bipartite or not. In fact, when b has only even components, we can reduce the b-matching problem in a non-bipartite graph to one on a bipartite graph, or equivalently, to a general network flow problem. Consider the perfect b-matching problem on a graph G where all the components of b are even. Construct a directed graph $D := (V(D), A(D))$ as follows: For each node v in G there are two nodes, v^- and v^+, in $V(D)$, and for each edge uv in G there are two directed edges, one from u^- to v^+ and one from v^- to u^+, in D. Now, solving the perfect b-matching problem in G is equivalent to solving a general network flow problem in D, subject to the following constraints (note that, $\delta^-(v^+) = \delta^+(v^-) = \emptyset$ for each $v \in V$):

$$\begin{array}{lcll} x(\delta^+(v^+)) - x(\delta^-(v^+)) & = & \frac{1}{2} b_v & (v \in V) \\ x(\delta^+(v^-)) - x(\delta^-(v^-)) & = & -\frac{1}{2} b_v & (v \in V) \\ x_a & \geq & 0 & (a \in A(D)). \end{array} \tag{80}$$

Since the right-hand-side in the general network flow problem is integral, it admits an integral optimum solution.

The 2-factor polytope: A 2-factor, or simple perfect 2-matching, in $G = (V, E)$ is a collection of node-disjoint circuits covering V. Note the difference with perfect 2-matchings, in which we may use edges twice. The perfect 2-matching problem is a special case of the perfect b-matching problem with b even, the 2-factor problem is not. Theorem 23 implies that the convex hull of 2-factors is described by:

$$\begin{array}{lll} x(\delta(v)) & = 2 & (v \in V) \\ 0 \leq x_e & \leq 1 & (e \in E) \\ x(\langle U \rangle) + x(F) & \leq |U| + \frac{1}{2}(|F| - 1) & (U \subseteq V,\ F \subseteq \delta(U),\ |F| \text{ odd}). \end{array} \tag{81}$$

Again, we may replace the blossom constraints with the odd cut constraints:

$$x(\delta(U) \setminus F) - x(F) \leq 1 - |F| \ (U \subseteq V, F \subseteq \delta(U), \ |F| \text{ odd}). \tag{82}$$

Using (81), and (82), one can derive a characterization of those graphs with simple perfect 2-matchings [see Belck, 1950; Tutte, 1952].

The edge-cover polyhedron: The convex hull of edge covers is described by:

$$\begin{array}{ccccll} & x(\delta(v)) & \geq & 1 & (v \in V) \\ 0 \leq & x_e & \leq & 1 & (e \in E) \\ & x(\delta(U) \cup \langle U \rangle) & \geq & \frac{1}{2}(|U| + 1) & (U \subseteq V, \ |U| \text{ odd}). \end{array} \tag{83}$$

7.3. Algorithms for general matchings

In this section we consider the polynomiality of the algorithms for general matching problems.

The reduction of a general matching problem to a perfect b-matching problem requires $O(|V| + |E|)$ time and results in a graph with $O(|V| + |E|)$ nodes and edges. So, given a polynomial time algorithm for the perfect b-matching problem, we may solve the general matching problem in polynomial time. The reduction from the perfect b-matching problem to the perfect matching problem, on the other hand, requires $O(b(V))$ steps and results in a perfect matching problem in a graph with $O(b(V))$ nodes. Hence we get:

> *There exists an algorithm for the b-matching problem with running time bounded by a polynomial in $|E(G)|$ and $b(V)$* (84)

[Edmonds, 1965b, see also Edmonds, Johnson & Lockhart, 1969, and Pulleyblank, 1973]. This time bound, however, is not very good. It grows polynomially with the values of b_v's and hence exponentially with the space required to encode them. (Recall that the integer b_v can be encoded in $\log(|b_v| + 1) + 1$ binary digits.) So, the given time bound is exponential in the size of the input of the problem. (An algorithm like this, whose running time is polynomial in the values of the numbers involved in the input, is called *pseudo-polynomial.*) If we restrict attention to those instances of the general matching problem in which the degree bounds and capacities are bounded by some fixed constant (or by a polynomial in $|V(G)|$), (84) yields polynomial algorithms:

> *The (simple) (perfect) 2-matching problem and the edge cover problem can be solved in time bounded by a polynomial in $|V(G)|$.* (85)

A different approach is required to solve the general b-matching problem in polynomial time.

7.3.1. A strongly polynomial algorithm for perfect b-matching

We describe an algorithm, due to Edmonds, that solves a b-matching problem by first solving a single general network flow problem and then a single perfect matching problem. It is based on the following 'sensitivity' result.

Theorem 24. *Let $G = (V, E)$ be an undirected graph and $b, b' \in \mathbb{Z}_+^V$. If x' is a minimum weight perfect b'-matching with respect to a given weight function $w \in \mathbb{Z}_+^E$, then there exists a minimum weight perfect b-matching x (with respect to w) such that*

$$|x_e - x'_e| \leq \sum_{v \in V} |b_v - b'_v| \quad \text{for each } e \in E.$$

Proof. Let $d := b - b'$. Clearly, it suffices to prove the theorem for the special case that $\sum_{v \in V} |d_v| = 2$. In fact, we will additionally assume that $b \in \{0, 1\}^V$. (In applying this theorem, we only need that case anyway. Moreover, the other cases are proved similarly.) So there exist $u_1, u_2 \in V$ such that $d_{u_1} = d_{u_2} = 1$ and $d_u = 0$ if $u \notin \{u_1, u_2\}$.

Let x' be a minimum weight perfect b'-matching, and x'' be a minimum weight perfect b-matching. Let B be the collection of all $y \in \mathbb{Z}^E$ such that:

$$\begin{array}{lllll} & y(\delta(v)) & = & d_v & (v \in V) \\ 0 \leq & y_e & \leq & x''_e - x'_e & (e \in E \text{ and } x''_e \geq x'_e) \\ x''_e - x'_e \leq & y_e & \leq & 0 & (e \in E \text{ and } x''_e \leq x'_e). \end{array} \tag{86}$$

Note that if $y \in B$, $x'' - y$ is a perfect b'-matching and $x' + y$ is a perfect b-matching. Hence $w^\top(x'' - y) \geq w^\top x'$, and thus $w^\top(x' + y) \leq w^\top x''$. Which implies that for each $y \in B$, $x' + y$ is a minimum weight perfect b-matching. So it suffices to prove that B contains a vector y with $|y_e| \leq 2$ for each $e \in E$.

Take a sequence $v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k$ of edges and nodes such that the following conditions are satisfied: $v_0 = u_1$ and $v_k = u_2$; $e_i = v_{i-1}v_i$ for $i = 1, \ldots, k$; if i is odd then $x''_{e_i} > x'_{e_i}$; if i is even then $x''_{e_i} < x'_{e_i}$; and, for each edge e at most $|x''_e - x'_e|$ edges e_i are equal to e. It is not difficult to see that, since $x'' - x' \in B$, such a sequence exists. Assume that the sequence is as short as possible. This implies that we do not use an edge more than twice in the sequence. Let $y \in \mathbb{Z}^E$ be defined by $y_e := \sum_{i=1, e_i = e}^{k} (-1)^{i+1}$. Then $y \in B$ and $|y_e| \leq 2$, so the theorem follows. □

We can apply this theorem in solving perfect b-matching problems as follows: Let x' be a minimum weight perfect b'-matching, where $b'_v := 2\lfloor \frac{1}{2} b_v \rfloor$ for each $v \in V$. Next define $d := b - b'$ ($\in \{0, 1\}^V$) and search for a minimum weight general matching $\bar{x}$ subject to the constraints:

$$\begin{array}{lll} x(\delta(v)) = d_v & & (v \in V) \\ x_e \geq \max\{-|V|, -x'_e\} & & (e \in E). \end{array} \tag{87}$$

Then, by Theorem 24, $x' + \bar{x}$ is a minimum weight perfect b-matching.

By the remarks following (73) and the reductions in Section 7.1 we can transform the general matching problem subject to (87) into perfect matching problem on a graph whose size is a polynomial in the size of G. As b' has only even components the perfect b'-matching problem is a general network flow problem. So, we have:

> *A b-matching problem in a graph G can be solved by solving* **one** *polynomially sized general network flow problem and* **one** *polynomially sized perfect matching problem (Edmonds).* (88)

The general network flow problem with constraints (80) is essentially equivalent to the min-cost flow (or circulation) problem. The first polynomial algorithm for the min-cost circulation problem was developed by Edmonds & Karp [1970, 1972] and has running time polynomial in $\sum_{v \in V(D)} \log(|b_v| + 1)$. This algorithm combines the pseudo-polynomial 'out-of-kilter' method [Yakovleva, 1959; Minty, 1960; and Fulkerson, 1961] with a scaling technique. Cunningham and Marsh [see Marsh, 1979] and Gabow [1983] found algorithms for b-matching that are polynomial in $\sum_{v \in V(D)} \log(|b_v| + 1)$, also using a scaling technique. The disadvantage of these algorithms is that the number of arithmetic steps grows with $\sum_{v \in V(D)} \log(|b_v| + 1)$. So, larger numbers in the input not only involve more work for each arithmetic operation, but also require more arithmetic operations. This raised the question of whether there is an algorithm such that the number of arithmetic operations it requires is bounded by a polynomial in $|V(D)|$ and the size of the numbers calculated during its execution is bounded by a polynomial in $\sum_{v \in V(D)} \log(|b_v| + 1)$ (this guarantees that no single arithmetic operation requires exponential time). For a long time this issue remained unsettled, until Tardos [1985] showed that, indeed, there exists such a, *strongly polynomial*, algorithm for the min-cost circulation problem [see also Goldberg & Tarjan, 1989]. Combining this with (88) we get:

Theorem 25. *There exists a strongly polynomial algorithm for the general matching problem.*

For a similar strongly polynomial algorithm for b-matching, see Anstee [1987].

7.4. Parity constraints

7.4.1. The Chinese postman problem [Kwan Mei-Ko, 1962; Edmonds, 1965a]

Given a connected graph $G = (V, E)$ and a length function $\ell \in \mathbb{Z}_+^E$: find a closed walk $e_1, \ldots, e_k$ in the graph using each edge at least once – we call this a *Chinese postman tour* – such that its length $\ell(e_1) + \cdots + \ell(e_k)$ is as small as possible.

If G is *Eulerian*, i.e., the degree of each node is even, then there exists an *Eulerian walk*, that is a closed walk using each edge exactly once. This is Euler's [1736] famous resolution of the Königsberger bridge problem. So, for Eulerian graphs the Chinese postman problem is trivial (actually finding the Eulerian

walk takes $O(|E|)$ time). On the other hand, if G has nodes of odd degree, every Chinese postman tour must use some edges more than once. We call a vector $x \in \mathbb{Z}^E$ such that $x_e \geq 1$ for each edge e and $\sum_{e \in \delta(v)} x_e$ is even for each node v, *Eulerian*. By Euler's theorem it is clear that for each Eulerian vector x there is a Chinese postman tour that uses each edge e exactly x_e times. Thus, solving the Chinese postman problem amounts to finding an Eulerian vector x of minimum length $\ell^\top x$. Clearly, a minimum length Eulerian vector can be assumed to be $\{1, 2\}$-valued. Hence, searching for a shortest Eulerian vector x amounts to searching for a set $F := \{e \in E \mid x_e = 2\}$ with $\ell(F)$ minimum such that duplicating the edges of F leads to an Eulerian graph. Duplicating the edges in F leads to an Eulerian graph exactly when each node v is incident to an odd number of edges in F if and only if the degree of v in G is odd. So, the Chinese postman problem is a 'T-join problem' discussed below.

There are other versions of the Chinese postman problem. In a directed graph, the problem is a general network flow problem. Other versions, including: the rural postman problem in which we need only visit a subset of the edges; the mixed Chinese postman problem in which some edges are directed and others are not; and the windy postman problem in which the cost of traversing an edge depends on the direction, are NP-hard.

7.4.2. The T-join problem

Given a graph $G = (V, E)$ and an even subset T of the node set V, a subset F of edges such that $|\delta_F(v)|$ is odd for each node v in T and even for each v not in T is called a *T-join*. The *T-join problem* is: Given a length function $\ell \in \mathbb{Z}^E$ find a T-join F of minimum length $\ell(F)$.

The T-join problem is the special case of the general matching problem with no upper bound constraints on the edges and no degree constraints other than the parity constraints.

7.4.3. Algorithms for T-joins

We describe two algorithms for finding a shortest T-join with respect to a length function $\ell \in \mathbb{Z}_+^{E(G)}$. The two algorithms rely on matchings in different ways.

The first algorithm is due to Edmonds & Johnson [1973]. Let H be the complete graph with $V(H) = T$. Define the weight function $w \in \mathbb{Z}_+^{E(H)}$ as follows. For each edge $uv \in E(H)$, w_{uv} is the length, with respect to ℓ, of a shortest uv-path P_{uv} in G. Find a minimum weight perfect matching, $u_1u_2, u_3u_4, \ldots, u_{k-1}u_k$ say, in H. The symmetric difference of the edge sets of the shortest paths $P_{u_1u_2}, P_{u_3u_4}, \ldots, P_{u_{k-1}u_k}$ is a shortest T-join.

Since the shortest paths and a minimum weight perfect matching can be found in polynomial time, the algorithm runs in polynomial time. In fact, we can find shortest paths in polynomial time when some of the edges have negative length, as long as G has no negative length circuit (see Section 9.2). But, when we allow negative length circuits, the shortest path problems become NP-hard. Nevertheless, the T-join problem with a general length function can be solved

in polynomial time (which implies that we also can find a T-join of maximum length).

In the second algorithm we construct a graph H as follows. For each node u in G and each edge e incident to u we have a node u_e. For each node u in T with even degree or not in T with odd degree, we have the node $\tilde{u}$ and the edges $\tilde{u}u_e$ for each edge e incident to u. For each node u in G and each pair e, f of edges in $\delta(u)$, we have an edge $u_e u_f$. Finally, for each edge $e = uv$ in G we have an edge $u_e v_e$ in H; we call these the *G-edges* of H. Each collection of G-edges is a matching in H and it corresponds to a T-join in G if and only if it is contained in a perfect matching of H. So, if we give each G-edge $u_e v_e$ weight ℓ_e and all other the edges in H weight 0, we have transformed the minimum length T-join problem in G into a minimum weight perfect matching problem in H.

Clearly, this algorithm allows edges with negative weights. Another way to solve a T-join problem with negative weights is by the following transformation to a T'-join problem with all weights non-negative. Let $N := \{e \in E \mid w_e < 0\}$ and $T_N := \{v \in V \mid \deg_N(v) \text{ is odd}\}$. Moreover, define $w^+ \in \mathbb{R}_+^E$ be defined by $w_e^+ := |w_e|$ for each $e \in E$ and $T' := T \triangle T_N$. Then $\min\{w(F) \mid F \text{ is a } T\text{-join}\} = w(N) + \min\{w^+(F) \mid F \text{ is a } T'\text{-join}\}$. F is a minimum weight T-join with respect to w if and only if then $F \triangle N$ is a minimum weight T'-join with respect to w^+.

Edmonds & Johnson [1973] derived a direct algorithm for the T-join problem, which, like Edmonds' weighted matching algorithm, maintains a dual feasible solution and terminates when there is a feasible primal solution that satisfies the complementary slackness conditions. Barahona [1980] and Barahona, Maynard, Rammal, & Uhry [1982] derived a 'primal' algorithm using dual feasibility as a stopping criterion (similar to the primal matching algorithm of Cunningham and Marsh (see Section 8.1)). Like the matching algorithm, these algorithms can be implemented to run in $O(|V|^3)$ and $O(|E||V| \log |V|)$ time, respectively. In planar graphs the T-join problem can be solved in $O(|V|^{3/2} \log |V|)$ time [Matsumoto, Nishizeki & Saito, 1986; Gabow, 1985; Barahona, 1990].

7.4.4. Min–max relations for T-joins – the T-join polyhedron

For each $U \subseteq V(G)$ with $U \cap T$ odd, we call $\delta(U)$ a *T-cut*. Clearly, the maximum number $\nu(G, T)$ of pairwise edge-disjoint T-cuts cannot exceed the smallest number $\tau(G, T)$ of edges in a T-join. Equality need not hold. For example, $\nu(K_4, V(K_4)) = 1 < 2 = \tau(K_4, V(K_4))$. Seymour proved [Seymour, 1981]:

> *In a bipartite graph G, $\nu(G, T) = \tau(G, T)$ for each even subset T of nodes.* (89)

Frank, Sebö & Tardos [1984] and Sebö [1987] derived short proofs of this result. In a bipartite graph, a maximum collection of pairwise edge-disjoint T-cuts can be found in polynomial time. (Korach [1982] gives an $O(|E||V|^4)$ procedure and Barahona [1990] showed that the above mentioned $O(|V|^3)$ and $O(|E||V| \log |V|)$

T-join algorithms can be modified to produce a maximum collection of disjoint T-cuts when the graph is bipartite.)

When the length function ℓ is non-negative and integral, we have the following min–max relation for shortest T-joins in arbitrary graphs [Lovász, 1975]:

> *The minimum length of a T-join with respect to a length function $\ell \in \mathbb{Z}_+^{E(G)}$ is equal to half the maximum number of T-cuts such that each edge e is in at most $2\ell(e)$ of them.* (90)

This can be proved from (89) or from the algorithm of Edmonds & Johnson [1993]. Let H be the bipartite graph obtained from G by replacing each edge e by a path of length $2\ell(e)$. If $\ell(e)$ is 0, contract e. A minimum length T-join in G corresponds to a minimum cardinality T-join in H. Applying (89) to H yields (90).

As a consequence, we obtain a linear inequality description of the *T-join polyhedron*, i.e., the set of vectors $x \in \mathbb{R}^E$ such that there exists a convex combination y of characteristic vectors of T-joins with $x \geq y$.

Corollary 26 [Edmonds & Johnson, 1973]. *Let T be an even subset of the node set of a graph $G = (V, E)$. Then the T-join polyhedron is the solution set of:*

$$\begin{array}{lll} x(\delta(U)) & \geq & 1 \quad (U \subseteq V;\ |U \cap T| \text{ is odd}) \\ x_e & \geq & 0 \quad (e \in E). \end{array} \tag{91}$$

Note that this result immediately yields Corollary 20. Conversely, Corollary 26 follows from Corollary 20 via the reduction to perfect matchings used in Schrijver's T-join algorithm. Alternatively, we can prove Corollary 26 in a manner analogous to our proof of Theorem 19. For a generalization of Corollary 26, see Burlet & Karzanov [1993].

The system (91) is not totally dual integral. The complete graph on four nodes, K_4, with $T = V(K_4)$ again provides a counterexample. In a sense, this is the only counterexample. One consequence of Seymour's characterization of 'binary clutters with the max-flow min-cut property' [Seymour, 1977] is:

> *If G is connected and T is even, then* (91) *is totally dual integral if and only if $V(G)$ cannot be partitioned into four sets $V_1, \ldots, V_4$ such that $V_i \cap T$ is odd and $G|V_i$ is connected for each $i = 1, \ldots, 4$ and for each pair V_i and V_j among $V_1, \ldots, V_4$, there is an edge uv with $u \in V_i$ and $v \in V_j$.* (92)

An immediate consequence of (92) is that, like bipartite graphs, series parallel graphs are *Seymour graphs*, meaning that $\nu(G, T) = \tau(G, T)$ for each even subset T of nodes. Other classes of Seymour graphs have been derived by Gerards [1992] and Szigeti [1993]. It is unknown whether recognizing Seymour graphs is in NP. Just recently, Ageev, Kostochka & Szigeti [1994] showed that this problem is in co-NP by proving a conjecture of Sebö.

Sebö [1988] derived a (minimal) totally dual integral system for the T-join polyhedron of a general graph. (For a short proof of this result and of (92) see

Frank & Szigeti [1994].) Sebö [1986, 1990] also developed a structure theory for T-joins analogous to the Edmonds–Gallai structure for matchings. The core of this structure theory concerns structural properties of shortest paths in undirected graphs with respect to length functions that may include negative length edges, but admit no negative length circuits. Frank [1993] derived a good characterization for finding a node set T in G that maximizes $\tau(G, T)$.

8. Other matching algorithms

In this section we discuss other algorithms for both cardinality and weighted matchings.

8.1. *A primal algorithm*

Edmonds' algorithm for finding a minimum weight perfect matching maintains a (structured) dual feasible solution and a non-perfect, and so infeasible, matching that together satisfy the complementary slackness conditions. At each iteration it revises the dual solution so that the matching can be augmented. When the matching becomes perfect it is optimal. An alternative approach is to maintain a perfect matching and a (structured) dual solution that satisfy the complementary slackness conditions. At each iteration, revise the matching so that the dual solution approaches feasibility. Cunningham & Marsh [1978] developed such a 'primal' algorithm. In outlining their algorithm we return to the notation of Section 6.2.

Let G be an undirected graph and suppose $w \in \mathbb{R}_+^{E(G)}$. Moreover, let $\pi \in \mathbb{R}^{\Omega(G)}$ be a structured dual solution, i.e., π satisfies (65) and (66). We also assume that $\pi_S \geq 0$ for each $S \in \Omega(G)$ with $|S| \neq 1$ and that M is a perfect matching in $\widetilde{G_\pi}$. If all the edges have non-negative reduced cost $w_e^\pi = w_e - \sum_{S \in \Omega(G); \delta(S) \ni e} \pi_S$, then π is dual feasible and, since M can be extended to a minimum weight perfect matching in G, π is optimal. Otherwise, we 'repair' π and M as follows:

REPAIR: Let $uv = e \in E(G)$ with $w_e^\pi < 0$ and suppose there exists an alternating path P in $\widetilde{G_\pi}$ from $\text{OUTER}_\pi[u]$ to $\text{OUTER}_\pi[v]$ starting and ending with a matching edge. We call such a path a *repairing path*. Carry out the following repairs ($R := \text{DEEP}_\pi[\text{OUTER}_\pi[u]]$):

EXPANDING R: If $\pi_R \leq -w_e^\pi$ and $|R| \neq 1$, revise the dual solution by changing π_R to 0. This means that we must expand R in $\widetilde{G_\pi}$ and extend M accordingly. Moreover, since π satisfies (66), we can extend P to an alternating path from the new node $\text{OUTER}_\pi[u]$ to $\text{OUTER}_\pi[v]$, again starting and ending with a matching edge.

We repeat EXPANDING R until $\pi_R > -w_e^\pi$ or $|R| = 1$. Note that each EXPANSION of R causes a matching edge, namely the starting edge of P, to receive positive reduced cost. Once we have finished EXPANDING, we call REPAIRING e to find a new perfect matching and a revised dual solution that satisfy the complementary slackness conditions.

REPAIRING e: If $|R| = 1$, or $\pi_R > -w_e^\pi$, replace M by $M\Delta(P \cup \{e\})$ and change the dual solution by adding w_e^π to π_R.

So, all that remains is the question of how to find a repairing path. Assume u is a node incident to an edge with negative reduce cost and let $r := \text{OUTER}_\pi[u]$. We create an auxiliary graph H by adding a new node u^* to G and an edge from u^* to u. Similarly, we construct $\widetilde{H}_\pi$ by adding the edge u^*r to $\widetilde{G}_\pi$. Consider the Edmonds–Gallai structure of $\widetilde{H}_\pi$. There are two possibilities:

1. There is an edge between u and $\text{DEEP}_\pi[v]$ with negative reduced cost for some node $v \in D(\widetilde{H}_\pi)$. In this case, let Q be an M-alternating u^*v-path (Q exists because $v \in D(\widetilde{H}_\pi)$ and u^* is the only node in H exposed with respect to M). Clearly $Q \setminus \{u^*r\}$ is a repairing path.

2. If there is no such node v, we change the dual variables according to the definitions in (68) and (69), but with the understanding that in (69) we ignore those edges with negative reduced cost. We also ignore a dual change in u^* (note that u^* is a singleton component of $D(\widetilde{H}_\pi)$). We repeat this operation until 1. applies or until all the edges incident to u receive a non-negative reduced cost.

Needless to say, in implementing the algorithm we do not need to find the Edmonds–Gallai structure explicitly, but instead use GROW and SHRINK. The algorithm can be implemented in $O(|V(G)|^3)$ time.

8.2. Shortest alternating paths and negative alternating circuits

Given a matching M, a weight function $w \in \mathbb{R}^{E(G)}$ and a set of edges F, we define $w_M(F) := w(F \setminus M) - w(F \cap M)$. A *negative* circuit is an even alternating circuit C with $w_M(C) < 0$. A matching M is called *extreme* if it admits no negative circuit. In a manner similar to the proof of Theorem 1, one can prove that a perfect matching is extreme if and only if it is a minimum weight perfect matching. This suggests the following algorithm for finding a minimum weight perfect matching:

NEGATIVE CIRCUIT CANCELLING: Given a perfect matching M, look for a negative circuit. If none exists, M is extreme and hence optimal. If M admits a negative circuit C, replace M with $M\Delta C$ and repeat the procedure.

Given a matching M and an exposed node u, an augmenting path P starting at u is called a *shortest augmenting path from* u if it minimizes $w_M(P)$. It is easy to prove that if M is extreme and P is an augmenting path starting at an exposed node u, then $M\Delta P$ is extreme if and only if P is a shortest augmenting path from u. This also suggests an algorithm:

SHORTEST AUGMENTING PATHS: Given an extreme matching M, (initially $M = \emptyset$), look for a shortest augmenting path. If none exists, M is a minimum weight maximum cardinality matching. If M admits a shortest augmenting path P, replace M by $M\Delta P$ and repeat the procedure.

So the question arises: How to find negative circuits or shortest augmenting paths? The answer is not so obvious. We can hardly check all possible alternating circuits or augmenting paths. In fact, the observations above are weighted analogues of the theorem of Berge and Norman and Rabin (Theorem 1). However, Edmonds' algorithm for minimum weight perfect matching can be viewed as a shortest augmenting path algorithm and Cunningham and Marsh's primal algorithm is a negative circuit cancelling method. Derigs [1981] [see also Derigs, 1988b] developed versions of these algorithms in which shortest augmenting path occur more explicit. Not surprisingly, these algorithms also rely on alternating forests, shrinking and the use of dual variables.

8.3. Matching, separation and linear programming

In Section 5 we formulated the weighted matching problem as a linear programming problem. Can we solve it as a linear program? The main problem is the number of inequalities. There are, in general, an exponential number of blossom constraints (viz. odd cut constraints). A first approach to overcoming this is, in fact, the development of algorithms like Edmonds' algorithm and the primal algorithm by Cunningham and Marsh that can be viewed as special purpose versions of simplex methods in which only the constraints corresponding to non-zero dual variables are explicitly considered. A second approach is to use the ellipsoid method, the first polynomial time algorithm for linear programming [Khachiyan, 1979]. Grötschel, Lovász & Schrijver [1981], Karp & Papadimitriou [1982] and Padberg & Rao [1982] observed that the polynomial time performance of this method is relatively insensitive to the size of the system of linear constraints. The only information the ellipsoid method needs about the constraint system is a polynomial time separation algorithm for the set of feasible solutions. A *separation algorithm* for a polyhedron solves the following problem.

Separation problem: Given a polyhedron $P \subseteq \mathbb{R}^n$ and a vector $\tilde{x} \in \mathbb{R}^n$, decide whether $\tilde{x} \in P$ and, if it is not, give a *violated inequality*, that is an inequality $a^T x \leq \alpha$ satisfied by each $x \in P$, but such that $a^T \tilde{x} > \alpha$.

Padberg & Rao [1982] developed a separation algorithm for the perfect matching polytope. It is easy to check whether a given $x \in \mathbb{R}^{E(G)}$ satisfies the non-negativity and degree constraints. So, the separation problem for the perfect matching polyhedron is essentially: Given a non-negative vector $x \in \mathbb{R}^{E(G)}$, find an odd collection S of nodes such that $x(\delta(S)) < 1$ or decide that no such S exists. This problem can be solved by solving the following problem (with $T = V(G)$).

Minimum capacity T-cut problem: Given an even collection T of nodes and $x \in \mathbb{R}_+^{E(G)}$, find $S \subsetneq V(G)$ with $|S \cap T|$ odd and $x(\delta(S))$ as small as possible.

We call a set $\delta(S)$ with $S \subsetneq V(G)$ a *T-separator* if $S \cap T$ and $S \setminus T$ are not empty. A *minimum T-cut* is a T-cut $\delta(S)$ with $x(\delta(S))$ as small as possible. We define a *minimum T-separator* similarly.

Crucial to the solution of this problem is the following fact.

Let $\delta(W)$ be a minimum T-separator, then there exists a minimum T-cut $\delta(S)$ with $S \subseteq W$ or $S \subseteq V(G) \setminus W$ [Padberg & Rao, 1982]. (93)

To prove this, let $\delta(W)$ be a minimum T-separator and $\delta(Z)$ be a minimum T-cut. If $\delta(W)$ is a T-cut, $Z \subseteq W$ or $Z \subseteq V(G) \setminus W$, we are done. So, suppose none of these is the case. By interchanging W and $V(G) \setminus W$ or Z and $V(G) \setminus Z$ (or both) we may assume that $|Z \cap W \cap T|$ is odd and $V(G) \setminus (W \cup Z)$ contains a node of T. Hence $\delta(W \cap Z)$ is a T-cut and $\delta(W \cup Z)$ is a T-separator. So, $x(\delta(W)) \leq x(\delta(W \cup Z))$. Now, straightforward calculations show that: $x(\delta(Z)) - x(\delta(W \cap Z)) \geq x(\delta(Z)) - x(\delta(W \cap Z)) + x(\delta(W)) - x(\delta(W \cup Z)) = 2 \sum_{u \in Z \setminus W} \sum_{u \in W \setminus Z} x_{uv} \geq 0$, which completes the proof of (93).

This suggests the following recursive algorithm. Determine a minimum T-separator $\delta(W)$. If $|W \cap T|$ is odd we are done, $\delta(W)$ is a minimum T-cut. Otherwise, we search for a minimum $(T \setminus W)$-cut in $G \times W$ and a minimum $(T \cap W)$-cut in $G \times (V(G) \setminus W)$. By (93), one of these two yields a minimum T-cut in G. It is easy to see that this recursive method requires at most $|T| - 1$ searches for a minimum T-separator. Each search for a T-separator can be carried out by solving $|T| - 1$ max-flow problems. (Indeed, fix $s \in T$ and use a max-flow algorithm to find a minimum s, t-cut for each $t \in T \setminus \{s\}$.) So the minimum odd cut problem and the separation problem for the perfect matching polytope can be solved in polynomial time by solving a series of $O(|T|^2)$ max-flow problems. Thus, the ellipsoid method provides a new polynomial time algorithm for the minimum weight perfect matching problem. (In fact, the minimum T-cut algorithm can be improved so that only $|T| - 1$ max-flow problems are required, by calculating a 'Gomory-Hu' tree [see Padberg & Rao, 1982].)

This method is not practical because the ellipsoid method performs poorly in practice. On the other hand, the separation algorithm can be used in a *cutting plane approach* for solving matching problems via linear programming. Start by solving a linear program consisting of only the non-negativity and degree constraints. If the optimal solution x^* to this problem is integral, it corresponds to a perfect matching and we are done. Otherwise, use Padberg and Rao's procedure to find an odd cut constraint violated by x^*. Add this to the list of constraints and resolve the linear programming problem. Grötschel & Holland [1985] built a matching code based on this idea. At that time, their code was competitive with existing combinatorial codes (based on Edmonds' or Cunningham and Marsh's algorithm). This contradicted the general belief that the more fully a method exploits problem structure, the faster it should be. This belief has been reconfirmed, at least for the matching problem, by new and faster combinatorial matching codes.

An entirely different approach to solving matching problems with linear programming is to construct a polynomial size system of linear inequalities $Ax + By \leq c$ such that $\{x \in \mathbb{R}^{E(G)} \mid Ax + By \leq c\}$ is the (perfect) matching polytope. We call such a linear system a *compact system* for the (perfect) match-

ing polytope. Although, perfect matching polytopes of planar graphs [Barahona, 1993a] and, in fact, perfect matching polytopes of graphs embeddable on a fixed surface [Gerards, 1991] have compact systems, no compact system is known for the matching problem in general graphs. It should be noted that compact systems for matching polytopes that use no extra variables do not exist, not even for planar graphs [Gamble, 1989]. Yannakakis [1988] proved that there is no compact *symmetric* system for the matching polytope. (Here, 'symmetric' refers to an additional symmetry condition imposed on the systems.)

Barahona [1993b] proposes yet a different approach. Given a matching M, one can find a negative circuit with respect to M by searching for an even alternating circuit C with minimum average weight $w_M(C)/|C|$. When using these special negative circuits, $O(|E|^2 \log |V|)$ negative circuit cancellations suffice for finding a minimum weight perfect matching. An even alternating circuit of minimum average weight can be found by solving a polynomially sized linear programming problem. Hence, we can find a minimum weight perfect matching by solving $O(|E|^2 \log |V|)$ compact linear programming problems.

8.4. *An algorithm based on the Edmonds–Gallai structure*

Next we present an algorithm, due to Lovász & Plummer [1986], for finding a largest matching in a graph. Like the blossom algorithm, it searches for alternating paths, but in a quite different manner. For instance, it does not shrink blossoms. The algorithm is inspired by the Edmonds–Gallai structure theorem. The algorithm maintains a list $\mathcal{L}$ of matchings all of size k. Given the list $\mathcal{L}$, define: $D(\mathcal{L}) := \cup_{M \in \mathcal{L}} \exp(M)$, $A(\mathcal{L}) := \Gamma(D(\mathcal{L})) \setminus D(\mathcal{L})$, and $C(\mathcal{L}) := V(G) \setminus (D(\mathcal{L}) \cup A(\mathcal{L}))$. So, if k is $\nu(G)$ and $\mathcal{L}$ is the list all maximum matchings, then $D(\mathcal{L})$, $A(\mathcal{L})$, $C(\mathcal{L})$ is the Edmonds–Gallai structure of G. During the algorithm, however, k ranges from 0 to $\nu(G)$ and $\mathcal{L}$ never contains more than $|V(G)|$ matchings. The clue to the algorithm is the following fact, which will serve as the stopping criterion.

> *If $M \in \mathcal{L}$ is such that $M \cap \langle D(\mathcal{L}) \rangle$ has exactly one exposed node in each component of $G|D(\mathcal{L})$ and no node in $A(\mathcal{L})$ is matched to a node in $A(\mathcal{L}) \cup C(\mathcal{L})$, then M is a maximum matching in G.* (94)

Indeed, in this case each component of $G|D(\mathcal{L})$ is odd and each node in $A(\mathcal{L})$ is matched to a different component of $G|D(\mathcal{L})$. Hence, it is easy to see that $|\exp(M)| = c_0(A(\mathcal{L})) - |A(\mathcal{L})|$; proving that M is maximum (cf. (22)).

The following notions facilitate the exposition of the algorithm. For $u \in D(\mathcal{L})$ we define $\mathcal{L}_u := \{M \in \mathcal{L} \mid u \in \exp(M)\}$. For each $M \in \mathcal{L}_u$ and $M' \in \mathcal{L}$, we denote the maximal path in $M \Delta M'$ starting at u by $P(u; M, M')$ (if M' is also in $\mathcal{L}_u$, this path consists of u only). An M-alternating path from a node in $\exp(M)$ to a node in $A(\mathcal{L})$ with an even number of edges is called *M-shifting*. If P is M-shifting for $M \in \mathcal{L}$, then $M \Delta P$ is a matching of size k with an exposed node $v \notin D(\mathcal{L})$. So

adding $M \Delta P$ to $\mathcal{L}$ adds the node v to $D(\mathcal{L})$. If Q is a path and u and v are nodes on Q then Q_{uv} denotes the uv-path contained in Q.

The algorithm works as follows:

Initially $\mathcal{L} := \{\emptyset\}$. Choose a matching M from $\mathcal{L}$. If it satisfies the conditions in (94) we are done: M is a maximum matching. Otherwise, apply the steps below to M to find an M'-augmenting or an M'-shifting path P with respect to some matching M'. If we find an M'-augmenting path P, we AUGMENT by setting $\mathcal{L}$ equal to $\{M' \Delta P\}$. If we find an M'-shifting path P, we SHIFT by adding $M' \Delta P$ to $\mathcal{L}$. The algorithm continues until we find a matching M satisfying (94).

Step 1: *If there is an edge $uv \in M$, with $u \in A(\mathcal{L})$ and $v \notin D(\mathcal{L})$,* choose $w \in \Gamma(u) \cap D(\mathcal{L})$ and $M_w \in \mathcal{L}_w$. If $P(w; M, M_w)$ has an odd number of edges it is M_w-augmenting and we AUGMENT. If $P(w; M, M_w)$ has an even number of edges and $uv \notin P(w; M, M_w)$ then $P(w; M, M_w) \cup \{wu, uv\}$ is M-shifting. Otherwise, either $P_{wu}(w; M, M_w)$ or $P_{wv}(w; M, M_w)$ does not contain uv and so is M_w-shifting. Select the appropriate path and SHIFT.

Step 2: *If there is a component S of $G|D(\mathcal{L})$ such that $M \cap \langle S \rangle$ is a perfect matching in $G|S$,* choose $w \in S$ and $M_w \in \mathcal{L}$. Since S is even, Step 3 below applies to M_w. Replace M by M_w and go to Step 3.

Step 3: *If there is a path Q in $G|D(\mathcal{L})$, such that $M \cap \langle D(\mathcal{L}) \rangle$ leaves the endpoints u and v of Q exposed,* then, if $uv \in E(G)$, go to Step 4. Otherwise, choose a node w in Q, different from u and v. If $w \in \exp(M)$, apply Step 3 with w in place of v and Q_{uw} in place of Q. If $w \notin \exp(M)$, choose $M_w \in \mathcal{L}_w$. If $P(w; M, M_w)$ is odd, it is M_w-augmenting, AUGMENT. If $P(w; M, M_w)$ is even, then $M' := M \Delta P(w; M, M_w)$ is a matching of size k that leaves w and (at least) one of u and v exposed. Assume $u \in \exp(M')$. Add M' to $\mathcal{L}$ and apply Step 3 with M' in place of M, w in place of v and Q_{uw} in place of Q.

(Note that each time we repeat Step 3, the path Q gets shorter.)

Step 4: *If there is an edge $uv \in G|D(\mathcal{L})$ such that $M \cap \langle D(\mathcal{L}) \rangle$ leaves u and v exposed,* consider the following two cases:

Step 4': *If $u, v \notin \exp(M)$,* let $M_u \in \mathcal{L}_u$. If $P(u; M, M_u)$ is odd, it is M_u-augmenting. Otherwise, define $M' := M \Delta P(u; M, M_u)$. M' has size k and has $u \in \exp(M')$ and $v \in \exp(M' \cap \langle D(\mathcal{L}) \rangle)$. Add M' to $\mathcal{L}$ and go to Step 4" with M' in place of M.

Step 4'': *If $u \in \exp(M)$ or $v \in \exp(M)$,* we may assume that $u \in \exp(M)$. If $v \in \exp(M)$ too, then uv is M-augmenting. If $v \notin \exp(M)$ and $vw \in M$ then $\{uv, vw\}$ is M-shifting.

The correctness of the algorithm follows from its description. It runs in $O(|V(G)|^4)$ time.

8.5. Parallel and randomized algorithms – matrix methods

The invention of parallel computers raised the question which problems can be solved substantially quicker on a parallel machine than on a sequential one. For problems that are polynomially solvable on a sequential machine a measure could be that the parallel running time is 'better than polynomial'. To make this explicit, Pippenger [1979] invented the class NC of problems solvable by an NC-algorithm.

A parallel algorithm is called NC-*algorithm* if its running time is a polynomial in the logarithm of the input size and requires only a polynomial number of processors. For more precise definitions see Karp & Ramachandran [1990].

Many problems have been shown to be in NC [see Karp & Ramachandran, 1990; Bertsekas, Castañon, Eckstein & Zenion, 1995, this volume]. But for matching the issue is still open. Partial answers have been obtained: Goldberg, Plotkin, & Vaidya [1993] proved that bipartite matching can be solved in sub-linear time using a polynomial number of processors [see also Vaidya, 1990; Goldberg, Plotkin, Shmoys & Tardos, 1992; Grover, 1992]. NC-algorithms for matching problems for special classes of graphs (or weights) have been derived by Kozen, Vazirani & Vazirani [1985], Dahlhaus & Karpinski [1988], Grigoriev & Karpinski [1987], He [1991], and Miller & Naor [1989]. But, whether or not: Has G a perfect matching? is in NC remains open.

On the other hand, if we allow algorithms to take some random steps, and also allow some uncertainty in the output, we can say more: there exist randomized NC-algorithms for matching. They rely on matrix methods (for a survey, see Galil [1986b]).

In Section 3 (see (16)) we already saw a relation between matchings in bipartite graphs and matrices. Tutte [1947] extended this to non-bipartite graphs.

> Let $G = (V(G), E(G))$ be an undirected graph, and let $\vec{G} = (V(G), A(\vec{G}))$ be a directed graph obtained by orienting the edges in G. For each edge e in G we have a variable x_e. Then the *Tutte matrix* of G (with respect to $\vec{G}$) is the $V(G) \times V(G)$ matrix $\vec{G}(x)$ defined by:
>
> $$\vec{G}(x)_{uv} := \begin{cases} x_{uv} & \text{if } \overrightarrow{uv} \in A(\vec{G}) \\ -x_{uv} & \text{if } \overrightarrow{vu} \in A(\vec{G}) \\ 0 & \text{if } uv \notin E(G) \end{cases}. \tag{95}$$

Note that the Tutte matrix essentially just depends on G: reversing the orientation of and edge e in G just amounts to substituting $-x_e$ for x_e in $\vec{G}(x)$.

> *G has a perfect matching if and only if the determinant of $\vec{G}(x)$ is a non-vanishing polynomial in the variables x_e ($e \in E(G)$)* [Tutte, 1947]. (96)

To see this, let $\mathcal{F} \subseteq \{0, 1, 2\}^{E(G)}$ denote the collection of perfect 2-matchings. Then $\det(\boldsymbol{G}(x)) = \sum_{f \in \mathcal{F}} a_f \prod_{e \in E(G)} x_e^{f_e}$. Moreover, it is not hard to show that $a_f = 0$ if and only if the 2-matching f contains an odd circuit. On the other hand, perfect 2-matchings without odd circuits contain a perfect matching.

By itself (95) is not that useful for deciding whether or not G has a perfect matching. Determinants can be calculated in polynomial time if the matrix contains specific numbers as entries, but evaluating a determinant of a matrix with variable entries takes exponential time (in fact the resulting polynomial may have

an exponential number of terms). However, by the following lemma, we can still use the Tutte matrix computationally.

Lemma 27 [Schwartz, 1980]. *Let $p(x_1, \ldots, x_m)$ be a non-vanishing polynomial of degree d. If $\hat{x}_1, \ldots, \hat{x}_m$ are chosen independently and uniformly at random from $\{1, \ldots, n\}$ then the probability that $p(\hat{x}_1, \ldots, \hat{x}_m) = 0$ is at most d/n.*

Proof. Write $p(x_1, \ldots, x_m)$ as $\sum_{\ell=0}^{d_m} p_{d-\ell}(x_1, \ldots, x_{m-1}) x_m^\ell$, where each p_k is a polynomial in $x_1, \ldots, x_{m-1}$ of degree at most k.

By induction to the number of variables, the probability that $p_{d-d_m}(\hat{x}_2, \ldots, \hat{x}_m) = 0$ is at most $(d - d_m)/n$. On the other hand, if $p_{d-d_m}(\hat{x}_2, \ldots, \hat{x}_m) \neq 0$ then $p(\hat{x}_1 \ldots, \hat{x}_{m-1}, x_m)$ is a non-vanishing polynomial in x_m of degree d_m, so has at most d_m roots. In other words, if $p_{d-d_m}(\hat{x}_1, \ldots, \hat{x}_{m-1}) \neq 0$, the probability that $p(\hat{x}_1, \ldots, \hat{x}_m) = 0$ is at most d_m/n. Hence the probability that $p(\hat{x}_1, \ldots, \hat{x}_m) = 0$ is at most $(d - d_m)/n + d_m/n = d/n$. □

If we apply Lemma 27 to $p(x) = \det \tilde{G}(x)$, which has degree $|V(G)|$ if it is non-vanishing, and take $n = 2|V(G)|$, we get a randomized polynomial time algorithm with the property that if G has a perfect matching the algorithm discovers this with probability at least $\frac{1}{2}$ [Lovász, 1979b]. Although this randomized algorithm is slower that the fastest deterministic ones, it has the advantage that it can be parallelized. The reason is that calculating a determinant is in NC [Csansky, 1976]. So we have:

> *There exists a randomized* NC*-algorithm that gives output '$\nu(G) = |V(G)|$' with probability at least $\frac{1}{2}$ if the input graph G has a perfect matching* [Lovász, 1979b; Csanski, 1976]. (97)

(Note that by running this algorithm several times, we can improve the probability of success as much as we want.)

More generally, we have a randomized NC-algorithm for deciding whether $\nu(G) \geq k$ (just add $|V(G)| - 2k$ mutually non-adjacent nodes to G, each of them adjacent to all nodes of G and than decide whether the new graph has a perfect matching). If we, combine this with a binary search on k, we get NC-algorithm that gives a number $\ell \leq \nu(G)$, that is equal to $\nu(G)$ with high probability.

These randomized algorithms have one big disadvantage: they are 'Monte Carlo' type algorithms. If G has no perfect matching the Lovász-Csanski algorithm does not discover this. The algorithm presented for $\nu(G)$ always gives an output $\ell \leq \nu(G)$, but never tells us that $\ell = \nu(G)$ (unless by change $\ell = \frac{1}{2}|V(G)|$). Karloff [1986] resolved this problem by deriving a randomized NC-algorithm that determines a set $B \subseteq V(G)$ such that with high probability $c_0(G \setminus B) - |B| = \text{def}(G)$ (cf. Theorem 10). Combining this with the previously described Monte Carlo algorithm for $\nu(G)$ we get a randomized NC-algorithm that provides an upper and a lower bound for $\nu(G)$, which are equal with high probability.

Knowing $\nu(G)$ does not provide us with a maximum matching. Of course, we can delete edges one by one from G, making G smaller and smaller, and

keep track what happens with the maximum size of a matching. If we store the edges whose deletion decreased the maximum size of a matching of the current graph, we get a maximum matching of our original graph. Combining this with a randomized algorithm for the size of a maximum matching we get a randomized algorithm for actually finding a maximum matching. However, this algorithm is highly sequential. Moreover, it is not obvious at all how to parallelize it, how to make sure that the different processors are searching for the same matching. (See Rabin & Vazirani [1989] for another sequential randomized algorithm for finding a maximum matching.) The first randomized NC-algorithm that finds a perfect matching with high probability if it exists is due to Karp, Upfal & Widgerson [1986]. It runs in $O(\log^3(|V(G)|))$ time. Below we sketch a randomized NC-algorithm due to Mulmuley, Vazirani, & Vazirani [1987] that runs in $O(\log^2(|V(G)|))$ time.

The main trick of this algorithm, besides using the Tutte matrix, is that it first implicitly selects a canonical perfect matching; which than is explicitly found. Let $\mathcal{U}(G)$ denote the set of all $w \in \mathbb{Z}_+^{E(G)}$ such that the minimum perfect matching, denoted by M_w, is unique. Mulmuley, Vazirani, and Vazirani proved the following fact: (If $e = uv \in E(G)$, then $\hat{x}_e^w := 2^{w_e}$ and $\vec{G}_e(x)$ denotes the submatrix of $\vec{G}(x)$ obtained by removing the row indexed by u and the column indexed by v.)

If $w \in \mathcal{U}(G)$, then
(1) $2^{-2w(M_w)} \det \vec{G}(\hat{x}^w)$ *is an odd integer,*
(2) $uv \in M_w \iff 2^{2(w_e - w(M_w))} \det \vec{G}_e(\hat{x}^w)$ *is an odd integer.* (98)

So as soon as we have found a $w \in \mathcal{U}(G)$, we can find a perfect matching by calculating the determinants in (98), which can be done in parallel by Csanski's NC-algorithm. The following lemma yields a randomized algorithm for selecting a weight function in $\mathcal{U}(G)$.

Lemma 28 [Mulmuley, Vazirani, & Vazirani, 1987]. *Let $S = \{x_1, \ldots, x_n\}$ be a finite set and $\mathcal{F}$ a collection of subsets of S. Assume that $w_1, \ldots, w_n$ are chosen uniformly and independently at random from $\{1, \ldots, 2n\}$. Then the probability that there is a unique $F \in \mathcal{F}$ minimizing $w(F)$ is at least $\frac{1}{2}$.*

Proof. The probability p that the minimum weight set is not unique is at most n times the probability p_1 that there exists a minimum weight set in $\mathcal{F}$ containing x_1 and a minimum weight set in $\mathcal{F}$ not containing x_1. For each fixed $w_2, \ldots, w_n$ this probability is either 0 or $1/2n$. Hence p_1 is at most $1/2n$. So $p \leq np_1 \leq \frac{1}{2}$. □

Hence, there exists a randomized NC-algorithm for finding a perfect matching and thus also for finding a maximum matching. It requires $O(|E| \log |V|)$ random bits. Chari, Rohatgi & Srinivasan [1993] found a very nice generalization of Lemma 28 that enables the design of randomized NC-algorithms that require only $O(|V| \log(|E|/|V|))$ random bits.

8.5.1. Counting perfect matchings

So randomization can help us where determinism does not (seem to) work. The same feature comes up when considering another computational task related to matchings: Count the number of perfect matchings in G. Over the years this problem has received a lot of attention, leading to many beautiful results. For many of these, and many references, see Lovász & Plummer [1986] and Minc [1978]. As the topic lies beyond the scope of this chapter, we will only mention a few results relevant from a computational point of view.

Valiant [1979] proved that counting the perfect matchings in a graph is as hard as solving any problem in NP, even for bipartite graphs (it is '#P-complete'). So, assuming P $\neq$ NP, there exists no polynomial time algorithm for calculating the number $\phi(G)$ of perfect matchings in G.

Kasteleyn [1963, 1967], however, derived a polynomial algorithm for counting perfect matchings in a planar graph. The main idea behind this algorithm is as follows (for details see Lovász & Plummer [1986]). If $\vec{G}$ is an orientation of G we denote by $p(\vec{G})$ the determinant of the matrix obtained by substituting 1 for each variable x_e of the Tutte matrix $\vec{G}(x)$. It can be shown that $p(\vec{G}) \leq \phi(G)^2$. $\vec{G}$ is called a *Pfaffian orientation* of G if $p(\vec{G}) = \phi(G)^2$. Kasteleyn proved that a planar graph has a Pfaffian orientation which can be found in polynomial time. So counting perfect matchings in planar graphs reduces to calculating a determinant. Not all graphs have Pfaffian orientations. For instance, $K_{3,3}$ does not have one. Little [1974] extended Kasteleyn's result by proving that if a graph has no subdivision of $K_{3,3}$ as a subgraph it has Pfaffian orientation. Vazirani [1989] showed that counting perfect matchings in these graphs is in fact in NC (by deriving an NC-algorithm for finding the Pfaffian orientation for these graphs).

So far for deterministic algorithms. Based on an idea of Broder [1986], Jerrum & Sinclair [1989] derived a polynomial time algorithm to approximate $\phi(G)$ with high probability when G has minimum degree $\frac{1}{2}|V(G)|$. Their algorithm gives a number Y such that $|Y - \phi(G)| \leq \epsilon\phi(G)$ with probability at least $1 - \delta$. The algorithm is polynomial in $|V(G)|$, $\log 1/\epsilon$, $\log 1/\delta$. The existence of such an algorithm for general graphs is still open. The main idea of the algorithm of Jerrum and Sinclair is as follows. Let for each k, $\phi_k(G)$ denote the of matchings of size k. Jerrum and Sinclair approximate $\phi(G)$ by approximating the ratios $r_k := \phi_k/\phi_{k-1}$ and multiplying them. So the problem reduces to approximating r_k. We restrict ourselves to $r_{1/2(|V(G)|)}$ and approximate it by choosing uniformly and independently *almost perfect* matchings (i.e. matchings of size at least $\frac{1}{2}|V(G)| - 1$) at random and counting how many of these are perfect and how many not. Clearly, in this way we can get a good estimate of $r_{1/2(|V(G)|)}$. Remains the question how to select an almost perfect matching at random. The problem is that there are exponentially many of them. This problem is overcome by defining a random walk on the set of almost perfect matchings. The steps in this random walk are as follows. Given an almost perfect matching M, with probability $\frac{1}{2}$ choose $e = uv \in E$ uniformly and independently at random. If M is perfect and $e \in M$, we move from M to $M \setminus \{e\}$. If M is not perfect, $e \notin M$ and $|M \cap \delta(\{u, v\})| \leq 1$, we move from M to $(M \setminus \delta(\{u, v\})) \cup \{e\}$. In all other cases we stay at M. Thus we get a Markov

chain. It has a uniform stationary distribution and each random walk in the Markov chain converges to this stationary distribution. So if we start the Markov process with some arbitrary matching and 'walk forever', the matching will become 'more and more random'. The point is that we do not have to walk forever. This Markov chain is 'rapidly mixing', meaning that the probability distribution after a polynomial number of steps is very close to uniform (irrespective the matching we start off with). As we only need to approximate $r_{1/2(|V(G)|)}$, it suffices to select the almost perfect matching 'almost uniformly at random'. Explaining all the technicalities in full detail would go to far here, but we can sketch the main ideas.

We need some definitions. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the undirected graph with the almost perfect matchings in G as its nodes and with $M'M \in \mathcal{E}$ if $M \triangle M'$ is a path with at most two edges. Let $\mathcal{A} \in \mathbb{Z}^{\mathcal{V}\times\mathcal{V}}$ be defined by $\mathcal{A}_{M'M} = -1$ if $M'M \in \mathcal{E}$, $\mathcal{A}_{M'M} = 0$ if $M'M \notin \mathcal{E}$, and $\mathcal{A}_{MM} = \deg_{\mathcal{G}}(M)$ if $M \in \mathcal{V}$. Then the transition matrix of the above defined Markov chain is $\mathcal{P} := I - 1/(2|E|)\mathcal{A}$: the probability to move to M' being in M is $\mathcal{P}_{M'M}$. Finally we define $q \in \mathbb{R}^{\mathcal{V}}$ by $q_M := 1/|\mathcal{V}|$ for each $M \in \mathcal{V}$, and $x^M \in \{0,1\}^{\mathcal{V}}$ by $x^M_{M'} = 1$ if and only if $M' = M$.

$\mathcal{P}$ is a symmetric doubly stochastic matrix with only positive eigenvalues. The largest these eigenvalue is 1 and has multiplicity 1 (as $\mathcal{G}$ is connected). The corresponding eigenvector is q, i.e. the uniform distribution on $\mathcal{V}$ and the stationary distribution of the Markov chain. If we start our Markov chain with an almost perfect matching M then the probability distribution after k steps is $\mathcal{P}^k x^M$ which tends to q if k goes to ∞. The rate of convergence can be expressed in the second largest eigenvalue λ_2 of $\mathcal{P}$: $|(\mathcal{P}^k x^M)_{M'} - 1/|\mathcal{V}|| \leq \lambda_2^k$ for each $M' \in \mathcal{V}$. So far everything is just standard matrix theory.

Sinclair & Jerrum [1989] derived the following bound on the second largest eigenvalue of $\mathcal{P}$:

$$\lambda_2 \leq 1 - \tfrac{1}{2}\Phi(\mathcal{G})^2. \tag{99}$$

where

$$\Phi(\mathcal{G}) := \frac{1}{2|E|} \min\left\{ \frac{|\delta(S)|}{|S|} \,\middle|\, S \subseteq \mathcal{V}, |S| \leq \frac{1}{2}|\mathcal{V}| \right\}, \tag{100}$$

is the *conductance* of $\mathcal{G}$. Jerrum & Sinclair [1989], in turn, derived the following bound on the conductance.

$$\Phi(\mathcal{G}) \geq |V|^{-6}. \tag{101}$$

Combining all this we get:

$$\textit{If } k \geq \tau(\epsilon) := 2|V|^{12}\left(\log|\mathcal{V}| + \log\frac{1}{\epsilon}\right), \textit{ then } \left|(\mathcal{P}^k x^M)_{M'} - \frac{1}{|\mathcal{V}|}\right| \leq \epsilon\frac{1}{|\mathcal{V}|}. \tag{102}$$

So making $\lceil\tau(\epsilon)\rceil = O(|V|^{12}(|V|\log|V| + \log 1/\epsilon))$ steps in the Markov chain results in a random selection of an almost perfect matching from a probability distribution which is close to uniform.

These are the main ideas of Jerrum and Sinclair's algorithm for counting perfect matchings in graphs with minimum degree $\frac{1}{2}|V|$. The relation between the rate of convergence of a Markov chain and its conductance extends, under mild conditions, to other Markov chains, not related to matchings in graphs. Over the last decennium rapidly mixing Markov chains have become more and more important in the design of randomized counting or optimization algorithms.

9. Applications of matchings

In this section we discuss applications of matchings to other combinatorial optimization problems. In particular, we discuss the traveling salesman problem, shortest path problems, a multi-commodity flow problem in planar graphs, and the max-cut problem in planar graphs.

9.1. The traveling salesman problem

A *traveling salesman tour*, or *Hamiltonian circuit* in a graph $G = (V, E)$ is the edge set of a circuit that spans all the nodes, i.e., a closed walk through G that visits every node exactly once. Given a distance function $d \in \mathbb{R}^E$, the traveling salesman problem is to find a traveling salesman tour F of minimum length $d(F)$. The problem has many applications in many environments: routing trucks for pick-up and delivery services, drilling holes in manufacturing printed circuit boards, scheduling machines, etc.. The traveling salesman problem is NP-hard. In fact, simply finding a traveling salesman tour is NP-hard [Karp, 1972]. The problem has served pre-eminently as an example of a hard problem. For example, Lawler, Lenstra, Rinnooy Kan & Schmoys [1985] chose it as the guide in their tour through combinatorial optimization. Their volume provides a wide overview of research on this problem. For an update of what has emerged since then, see Jünger, Reinelt & Rinaldi [1995, this volume].

In this section we discuss a heuristic for the traveling salesman problem that uses matchings. We also discuss the relation between matching and polyhedral approaches to the traveling salesman problem. We assume from now on that $G = (V, E)$ is complete.

9.1.1. Christofides' heuristic

The problem is NP-hard and therefore is unlikely to be solvable in polynomial time. It makes sense then to take a heuristic approach, i.e., to find a hopefully good, but probably not optimal solution quickly. The heuristic we present here is due to Christofides [1976] and is meant for the case in which the distance function d is non-negative and satisfies the *triangle inequality*: $d_{uv} + d_{vw} \geq d_{uw}$ for each three nodes u, v, and w in G.

Let F be a minimum length spanning tree of G and let T be the set of nodes v in G with $\deg_F(v)$ odd (so F is a T-join). Find a minimum weight perfect matching M in $G|T$ with weight function d. Consider the union of F and M in the

sense that if an edge occurs in both sets it is to be taken twice as a pair of parallel edges. This union forms an Eulerian graph and an Eulerian walk in this graph visits each node of G at least once. The length of the walk is $d(F) + d(M)$. Since G is complete, we may transform the Eulerian walk into a traveling salesman tour by taking short cuts and, by the triangle inequality, the length of this tour is at most $d(F) + d(M)$.

The heuristic runs in polynomial time. There are many polynomial time algorithms for finding a minimum weight spanning tree, e.g., Borůvka's algorithm [Borůvka, 1926], Kruskal's algorithm [Kruskal, 1956], or Jarník's algorithm (Jarník [1930], better known by the names of its re-inventors Prim [1957] and Dijkstra [1959]). Kruskal's algorithm, for instance, runs in $O(|E| \log |V|)$ time. Edmonds' matching algorithm, described in Section 6, finds a minimum weight matching in polynomial time. Once the tree and the matching are known, an Eulerian walk and a traveling salesman tour can be found in linear time. Gabow & Tarjan [1991] showed that the heuristic can be implemented in $O(|V|^{2.5}(\log |V|)^{1.5})$. (Instead of a minimum weight matching, their version finds a matching with weight at most $1 + 1/|V|$ times the minimum weight.)

The following theorem shows that the heuristic produces a tour that is at most 50% longer than the shortest traveling salesman tour.

Theorem 29 [Christofides, 1976]. *Let $G = (V, E)$ be a complete graph and $d \in \mathbb{R}_+^E$ be a distance function satisfying the triangle inequality. Then $\lambda^* \leq \frac{3}{2}\lambda$, where λ is the length of a shortest traveling salesman tour, and λ^* is the length of the tour found by Christofides' heuristic.*

Proof. Let C be a shortest traveling salesman tour, and let F and M be the tree and matching found by the heuristic. Let T be the nodes $t_1, \ldots, t_k$, with odd degree in F, where the numbering corresponds to the order in which C visits these nodes. Let $\tilde{C}$ be the circuit with edges $t_1t_2, t_2t_3, \ldots, t_kt_1$. By the triangle inequality, $\tilde{C}$ is shorter than C. Let $\tilde{M}$ be the shorter of the two perfect matchings on T contained in $\tilde{C}$. Then $\tilde{M}$ is a perfect matching of $G|T$. So, $\lambda = d(C) \geq d(\tilde{C}) \geq 2d(\tilde{M}) \geq 2d(M)$. On the other hand, C contains a spanning tree – just delete an edge – so $\lambda = d(C) \geq d(F)$. Combining these inequalities, we see that $\lambda^* \leq d(F) + d(M) \leq \frac{3}{2}\lambda$. □

9.1.2. A polyhedral approach to the traveling salesman problem

Traveling salesman tours are connected 2-factors. So, the characteristic vectors of traveling salesman tours satisfy the following system of inequalities (compare with (81) and (82)):

$$\begin{array}{lcll} x_e & \geq & 0 & (e \in E) \\ x_e & \leq & 1 & (e \in E) \\ x(\delta(v)) & = & 2 & (v \in V) \\ x(\delta(U) \setminus F) - x(F) & \geq & 1 - |F| & (U \subseteq V,\ F \subseteq \delta(U)) \\ x(\delta(U)) & \geq & 2 & (U \subseteq V;\ \emptyset \neq U \neq V). \end{array} \tag{103}$$

In fact, every integral solution to (103) is the characteristic vector of a traveling salesman tour. Thus, the cutting plane approach described in Section 8.3 for solving the matching problem can be applied to the system (103) to solve the traveling salesman problem. In this case, however, the polyhedron defined by (103) has fractional extreme points and so success is not guaranteed. (Note that without the last set of inequalities, the system describes an integral polyhedron, namely the convex hull of 2-factors. The last set of inequalities, called the *subtour elimination constraints*, are necessary to 'cut-off' each 2-factor that is not a traveling salesman tour. However, adding these constraints introduces new, fractional, extreme points.) One could try to overcome this by adding more constraints to the system [see Grötschel & Padberg, 1985; Jünger, Reinelt & Rinaldi, 1995], but no complete description of the traveling salesman polytope is known. In fact, unless NP = co-NP, no 'tractable' system describing the traveling salesman polytope exists [see Karp & Papadimitriou, 1982].

'Partial' descriptions like (103), however, can be useful for solving traveling salesman problems. Minimum cost solutions to such systems provide lower bounds for the length of a shortest traveling salesman tour. These lower bounds can be used, for instance, to speed up branch-and-bound procedures. In fact, over the last decennium much progress has been made in this direction [see Jünger, Reinelt & Rinaldi, 1995].

The cutting plane approach requires a separation algorithm, or at least good separation heuristics, for the partial descriptions. We have separation algorithms for (103). Determining whether a given solution x satisfies the non-negativity, capacity and degree constraints is trivial. We can use a max-flow algorithm to determine whether x satisfies the subtour elimination constraints. So, all that remains is to find a polynomial time algorithm for the problem:

$$\text{Given } x \in \mathbb{R}_+^{E(G)}, \text{ find a subset } U \subseteq V(G) \text{ and an odd subset } F \text{ of } \delta(U) \text{ such that } x(\delta(U) \setminus F) - x(F) < 1 - |F| \text{ or decide that no such subsets exist.} \tag{104}$$

These constraints are the 'odd cut constraints' for the 2-factor problem and, in view of the reductions of general matching to perfect matching, it should not be surprising that we can solve this problem in much the same way as we solved the separation problem for the odd cut constraints for perfect matching. Construct an auxiliary graph as follows. Replace each edge $e = uv$ in G with two edges in series: $e_1 := uw$ and $e_2 := wv$. Define $x^*_{e_1} := x_e$ and $x^*_{e_2} := 1 - x_e$. Let T be the set of nodes in the resulting graph G^* meeting an odd number of the edges e_2. Consider the problem:

$$\text{Find } U \subseteq V(G^*), \text{ such that } |U \cap T| \text{ is odd and } x(\delta(U)) < 1, \text{ or show that no such } U \text{ exists.} \tag{105}$$

It is not so hard to see that if U in (105) exists, then we may choose U so that for each edge e in G, at most one of e_1 and e_2 is contained in $\delta(U)$. Hence, (104) is equivalent to (105) and the separation problem (105) amounts to finding a minimum weight T-cut.

Above we considered the traveling salesman problem as a matching problem with side constraints, namely of finding shortest connected 2-factors. Other papers on matchings with side constraints are: Ball, Derigs, Hilbrand & Metz [1990], Cornuéjols & Pulleyblank [1980a, b, 1982, 1983], and Derigs & Metz [1992].

9.2. *Shortest path problems*

The *shortest path problem* is: Given two nodes s and t in G, find an s,t-path of shortest length $d(P)$ with respect to a length function $d \in \mathbb{R}^E$. In general, this problem is NP-hard, it includes the traveling salesman problem; but it is polynomially solvable when no circuit C in G has negative length $d(C)$. When all edge lengths are non-negative, the problem can be solved by the labeling methods of Bellman [1958] & Ford [1956], Dijkstra [1959], and Floyd [1962a, b] and Warshall [1962]. The algorithms also find shortest paths in directed graphs, even when negative length edges are allowed (though some adaptations are required) as long as no directed circuit has negative length. The presence of negative length edges makes the problem in undirected graphs more complicated. Simple labeling techniques no longer work. In fact, the problem becomes a matching, or more precisely, a T-join problem. Indeed, let $T := \{s, t\}$. Since no circuit has negative length, a shortest T-join is a shortest s,t-path (possibly joined by circuits of length 0). So we can find a shortest path in an undirected graph with negative length edges but no negative length circuits, by solving a T-join problem. Alternatively we can model the shortest path problem as a generalized matching problem. For each node $v \in V(G) \setminus \{s, t\}$, add a loop $\ell(v)$, then the shortest path problem is the generalized matching problem subject to the constraints:

$$\begin{array}{rcll} 0 \leq & x_e & \leq 1 & e \in E(G) \\ 0 \leq & x_{\ell(v)} & \leq 1 & v \in V(G) \setminus \{s,t\} \\ & x(\delta(v)) + 2x_{\ell(v)} & = 2 & v \in V(G) \setminus \{s,t\} \\ & x(\delta(v)) & = 1 & v \in \{s,t\} \end{array} \tag{106}$$

The reductions described in Section 7.1 reduce this problem to a perfect matching problem in an auxiliary graph.

9.2.1. *Shortest odd and even paths*

The *shortest odd path problem* asks for a shortest path from s to t with an odd number of edges. Similarly, the *shortest even path problem* asks for a shortest s,t-path with an even number of edges. In general these problems are NP-hard. The special case in which no circuit has negative length is, to my knowledge, still unsettled: the problems are not known to be hard, but neither do we know of any polynomial algorithm for them. If all edge lengths are non-negative, the problems are solvable in polynomial time: they are matching problems. We show this by a reduction due to Edmonds [see Grötschel & Pulleyblank, 1981]. We only consider the case of odd paths. The shortest even path problem can be solved by an easy reduction to the shortest odd path problem, or alternatively, by a similar reduction to the matching problem.

To find a shortest odd path between s and t, construct an auxiliary graph H as follows. Add to G a copy G' of G with the nodes s and t deleted (denote the copy in G' of node u by u' and the copy of edge e by e'). For each $u \in V(G) \setminus \{s, t\}$ add an edge from u to its copy u' in G'. The weight function w on H is defined by $w_e := w_{e'} := d_e$ for each $e \in E(G)$ and $w_{u'u} := 0$ for each $u \in V(G) \setminus \{s, t\}$. Let M be a perfect matching in H and define $P_M := \{e \in E(G) \mid e \in M \cap E(G) \text{ or } e' \in M \cap E(G')\}$. It is easy to see that P_M is the node-disjoint union of an odd s, t-path and a collection of circuits. If M has minimum length with respect to w, each of the circuits has length 0 and so minimum weight perfect matchings in H correspond to shortest odd s, t-paths in G.

Recently, Schrijver & Seymour [1994] characterized the *odd s, t-path polyhedron*, i.e., the convex hull of the subsets of $E(G)$ containing an odd s, t-path, thus proving a conjecture of Cook and Sebö. The inequalities describing the polyhedron are: $0 \leq x_e \leq 1$ for all $e \in E(G)$, and

$$2x(\langle W \rangle \setminus F) + x(\delta(W)) \geq 2 \text{ for each subgraph } H = (W, F) \text{ of } G \text{ such that both } s \text{ and } t \text{ are in } W \text{ but no } s, t\text{-path in } H \text{ is odd.} \quad (107)$$

9.3. *Max-cut and disjoint paths in planar graphs*

We conclude this section with the application of T-joins and planar duality to the max-cut problem and a disjoint paths problem in planar graphs. A graph G is *planar* if it can be embedded in the plane so that its edges do not cross. The *planar dual* G^* of G with respect to an embedding is defined as follows. The graph G divides the plane into several connected regions each corresponding to a node in $V(G^*)$. Each edge e in G separates at most two regions of the plane in the sense that, if we removed e, these regions would combine into one. For each edge $e \in E(G)$ there is an edge e^* in G^* joining the nodes in G^* corresponding to the regions separated by e. If e does not separate two regions, then it lies entirely in a single region and e^* is a loop at the corresponding node of $V(G^*)$. We identify each edge e in G with the corresponding edge e^* in G^*.

The graph G^* is planar and its definition suggests a natural embedding. If G is connected and G^* is embedded in the natural way, then $(G^*)^*$ is again G. The most prominent property of planar duality is that $C \subseteq E(G)$ $(= E(G^*))$ is a cycle in G if and only if it is a cut in G^* (recall that a *cycle* is a graph in which the degree of each node is even). The same relation exists between cuts in G and cycles in G^*.

The *max-cut problem* is: Given a weight function $w \in \mathbb{R}^{E(G)}$, find a cut $\delta(U)$ in G with $w(\delta(U))$ maximum. The problem is NP-hard in general [Karp, 1972], but polynomially solvable when G is planar. To see this, consider a planar graph G and a planar dual G^*. Define $T := \{v \in V(G^*) \mid \deg_{G^*}(v) \text{ is odd}\}$. Clearly, $F \in E(G^*)$ is a T-join if and only if $E(G^*) \setminus F$ is a cycle in $E(G^*)$. So, T-joins in G^* correspond to complements of cuts in G. Hence the max-cut problem in G is a T-join problem in G^* [Hadlock, 1975].

Combining planar duality with Seymour's theorem (89) for T-joins and T-cuts in bipartite graphs we obtain the following:

Theorem 30 [Seymour, 1981]. *Let G be a graph and let H be a collection pairs $\{s_1, t_1\}, \ldots, \{s_k, t_k\}$ of nodes. If the graph $G + H$, obtained from G by adding as extra edges the pairs in H, is planar and Eulerian, then the following are equivalent:*

(i) There exist edge-disjoint paths $P_1, \ldots, P_k$ in G such that each P_i goes from s_i to t_i;

(ii) For each $U \subseteq V(G)$, $|\delta_G(U)| \geq |\delta_H(U)|$.

Proof. Clearly, (ii) is necessary for (i), we show that it is also sufficient. Assume that (ii) holds and let $(G + H)^*$ be the planar dual of $G + H$ with respect to some embedding. Since $G + H$ is Eulerian, $E(G + H)$ is a cycle in $G + H$. In other words, $E((G + H)^*)$ is a cut in $(G + H)^*$ and so $(G + H)^*$ is bipartite.

Let T be the set of nodes in $V((G + H)^*)$ that meet an odd number of edges in H. Then H is a T-join in $(G + H)^*$. In fact, H is a minimum cardinality T-join in $(G + H)^*$. To see this, observe that for any other T-join F the symmetric difference $F \Delta H$ is a cycle in $(G + H)^*$ and so a cut in G. By (ii), $F \Delta H$ contains at least as many edges from F as from H. So, $|H| \leq |F|$ and H is a minimum cardinality T-join in $(G + H)^*$.

Now, applying (89) to $(G + H)^*$ and T, we see that there must be $|H| =: k$ disjoint odd cuts $C_1 = \delta(U_1), \ldots, C_k = \delta(U_k)$ in $(G + H)^*$. Clearly, each of these cuts has at least one edge in common with H and so each edge in H must be in exactly one of them. Assume $(s_i t_i)^* \in C_i$ for $i = 1, \ldots, k$. Without loss of generality, we may assume that the cuts are inclusion-wise minimal and so circuits in $G + H$. Then, $P_1 := C_1 \setminus s_1 t_1, \ldots, P_k := C_k \setminus s_k t_k$ are the desired paths. □

Matsumoto, Nishizeki, & Saito [1986] showed that the paths can be found in $O(|V(G)|^{5/2} \log |V(G)|)$ time. When $G + H$ is not Eulerian the problem becomes NP-hard [Middendorf & Pfeiffer, 1990]. For a general overview of the theory of disjoint paths, see Frank [1990].

10. Computer implementations and heuristics

10.1. Computer implementations

Over the years several computer implementations for solving matching problems have been designed, e.g. Pulleyblank [1973], Cunningham & Marsh [1978], Burkhard & Derigs [1980], Derigs [1981, 1986a, b, 1988b], Derigs & Metz [1986, 1991], Lessard, Rousseau & Minoux [1989] and Applegate & Cook [1993]. Grötschel & Holland [1985] used a cutting plane approach and Crocker [1993] and Mattingly & Ritchey [1993] implemented Micali and Vazirani's $O(\sqrt{|V|}|E|)$ algorithm for finding a maximum cardinality matching.

Designing efficient matching codes, especially those intended for solving large problems, involves many issues. Strategic decisions must be made, e.g., what algorithm and data structures to use. Moreover, tactical decisions must be made, e.g., how to select the next edge in the alternating forest and when to shrink blossoms. Finally, of course, numerous programming details affect the efficiency of the code. We restrict our attention to a few key strategic issues.

In solving large problems two paradigms appear to be important. The first of these is 'Find a 'good' starting solution quickly (the 'jump-start')' and the second is 'Avoid dense graphs'. We discuss the second paradigm first.

One feature of Grötschel and Holland's code [1985] (see Section 8.3) that competed surprisingly well with the existing combinatorial codes (based on Edmonds algorithm for instance), was that it first solved a matching problem in a sparse subgraph and then tuned the solution to find a matching in the original graph. Incorporating this approach sped up existing combinatorial codes significantly [Derigs & Metz, 1991]. The idea is to solve a minimum weight perfect matching problem on a (dense) graph G by first selecting a sparse subgraph G_{sparse} of G. A matching code, e.g., Edmonds' algorithm, can find a minimum weight perfect matching M and an optimal (structured) solution π in G_{sparse} quickly. In G the matching may not be of minimum weight and the dual solution may not be feasible. The second phase of the procedure corrects this. A primal algorithm, e.g., Cunningham and Marsh's algorithm described in Section 8.1, is ideal for this phase. Weber [1981], Ball & Derigs [1983], and Applegate & Cook [1993] have developed alternative methods for this.

The typical choice of G_{sparse} is the *k-nearest neighbor graph* of G, which is constructed by taking for each node u the k shortest edges incident to u. Typical choices for k run from 5 to 15. To give an impression of how few edges G_{sparse} can have: Applegate & Cook [1993] used their code to solve an Euclidean problem on 101230 nodes (i.e., the nodes lie in the Euclidean plane and the weight of an edge is given by the L_∞ distance between its endpoints). So, G is complete and has $0.5 \cdot 10^{10}$ edges. When k is 10, G_{sparse} has 10^6 edges or less then 0.05% of the all the edges in G. In fact, Applegate and Cook solved this 101230 node problem – a world record. For more moderately sized problems (up to twenty thousand nodes) their code seems dramatically faster than previously existing matching codes.

Many matching codes incorporate a jump-start to find a good matching and a good dual solution quickly before executing the full matching algorithm. Originally these initial solutions were typically produced in a greedy manner. Derigs and Metz [1986] suggested a jump-start from the fractional matching problem (or equivalently the 2-matching problem). First, solve the 2-matching problem: $\max\{w^\top x \mid x \geq 0;\ x(\delta(v)) = 2\ (v \in V)\}$. Let x^* and π^* be primal and dual optimal solutions to this linear programming problem (which can, in fact, be solved as a bipartite matching problem or a network flow problem). The set $\{e \in E \mid x_e^* > 0\}$ is the node-disjoint union of a matching $M' := \{e \in E \mid x_e^* = 2\}$ and a collection of odd circuits. Jump-start with the matching M obtained from M' and a maximum matching in each of the odd circuits and the dual solution π^* (setting the dual variables

corresponding to the blossoms equal to zero). Since x^* and π^* are primal and dual optimal solutions to the 2-matching problem, they satisfy the complementary slackness conditions. If G is dense, the 2-matching problem is first solved on a sparse subgraph. In fact, Applegate and Cook use different sparse graphs for finding the jump-start and for solving the actual problem (the latter is the k-nearest neighbor graph using the reduced costs with respect to the jump-start dual solution).

10.2. Heuristics

When solving large matching problems, searching for a good jump-start, or applying matchings in a heuristic for some other problem (e.g., Christofides' heuristic for the traveling salesman problem described in Section 9.1) it is often useful to use a heuristic to find a good matching quickly. A straightforward approach, called the greedy heuristic, attempts to construct a minimum weight perfect matching by starting with the empty matching and iteratively adding a minimum weight edge between two exposed nodes. The greedy heuristic runs in $O(|V|^2 \log |V|)$ time and finds a solution with weight at most $\frac{4}{3}|V|^{\log 3/2}$ times the minimum weight of a perfect matching [Reingold & Tarjan, 1981]. The version of the greedy heuristic designed to find a maximum weight matching, finds a solution with at least half the weight of a maximum weight matching. Results on greedy heuristics appear in Avis [1978, 1981], Avis, Davis & Steele [1988], Reingold & Tarjan [1981], Frieze, McDiarmid & Reed [1990] and Grigoriadis, Kalantari & Lai [1986].

Several heuristics have been developed for Euclidean matching problems where the set of points that have to be matched lie in the unit square. Many of these heuristics find the heuristic matching by dividing the unit square into subregions, finding a matching in each subregion and combining these matchings to a perfect matching between all the points. Other heuristics match the points in the order in which they lie on a space-filling curve. For detailed description and the analysis of such heuristics see: Bartholdi & Platzman [1983], Imai [1986], Imai, Sanae & Iri [1984]. Iri, Murota & Matsui [1981, 1982], Papadimitriou [1977], Reingold & Supowit [1983], Steele [1981], Supowit, Plaisted & Reingold [1980], Supowit & Reingold [1983], Supowit, Reingold & Plaisted [1983].

For a good overview on matching heuristics, see the survey of Avis [1983]. Here we mention some recent heuristics in more detail.

When the weight function w satisfies the triangle inequality, each minimum weight V-join is a perfect matching (or, when some edges have weight 0, can be transformed easily into a perfect matching with the same weight). So, when w satisfies the triangle inequality, we can use T-join heuristics as matching heuristics.

Plaisted [1984] developed a T-join heuristic that runs in $O(|V|^2 \log |V|)$ time and produces a T-join with weight at most $2 \log_3(1.5|V|)$ times the weight of an optimal solution.

Given a graph $G = (V, E)$, an even subset T of V and $w \in \mathbb{R}^E$, construct a T-join J as follows. (Note that w need not satisfy the triangle inequality, it would not survive the recursion anyway.)

AUXILIARY GRAPH: If $T = \emptyset$, then set $J := \emptyset$. Otherwise, construct the weighted complete graph H on the node set T. The weight w'_{uv} of each edge uv in H is the length of a shortest uv-path P_{uv} in G (with respect to w).
SHRINK: For each $u \in T$, define $n_u := \min\{w'_{uv} \mid v \in T\}$. Construct a forest F in H as follows. Scan each node in order of increasing n_u. If the node u is not yet covered by F, add to F an edge uv with $w'_{uv} = n_u$. Let $F_1, \ldots, F_k$ denote the trees of F and let $G' := H \times V(F_1) \times \cdots \times V(F_k)$. (If parallel edges occur select one of minimum weight to be in G'.) The pseudo-node corresponding to $V(F_i)$ is in T' if and only if $|V(F_i)|$ is odd. Apply the procedure recursively to G', w' and T' (starting with AUXILIARY GRAPH) and let J' be the resulting T'- join.
EXPAND: Let J^* denote the set of edges in H corresponding to the edges of J' in G. Choose T^* so that J^* is a T^*-join. Then $T_i^* := (T \triangle T^*) \cap V(F_i)$ is even for each i=1, ..., k. Let J_i be the unique T_i^*-join in each tree F_i. Then $J_H := J' \cup J_1 \cup \cdots \cup J_k$ is a T-join in H.
T-JOIN. Let J be the symmetric difference of the shortest paths $\{P_{uv} : uv \in J_H\}$.

Note that each tree F_i contains at least 2 nodes. So, if $|V(F_i)|$ is odd it is at least three. Hence, the depth of the recursion is bounded by $\log_3 |T|$.

Goemans & Williamson [1992] proposed a heuristic that not only yields a T-join F but also a feasible solution π of

$$\begin{aligned} \text{maximize} \quad & \sum_{S \in \Omega} \pi_S \\ \text{subject to} \quad & \sum_{S \in \Omega; \delta(S) \ni e} \pi_S \leq w_e \quad (e \in E) \\ & \pi_S \geq 0 \quad (S \in \Omega), \end{aligned} \tag{108}$$

where $\Omega := \{S \subseteq V \mid |S \cap T| \text{ odd}\}$. (108) is the dual linear programming problem of the T-join problem (cf. (91)). The weight of the heuristic T-join will be at most $(2 - 2/|T|) \sum_{S \in \Omega; \delta(S) \ni e} \pi_S$, so at most $2 - 2/|T|$ times the minimum weight of a T-join.

During the procedure we keep a forest F' (initially $V(F') := V(G)$ and $E(F') := \emptyset$). For each $v \in V(G)$, F'_v denotes the component of F' containing v. We also keep a feasible solution π of (108) (initially, $\pi \equiv 0$).

The basic step of the heuristic is as follows: among all edges $e = uv$ in G with $F'_u \neq F'_v$ and $F'_u \in \Omega$, select one, e^* say, that minimizes the quantity:

$$\frac{1}{p(F'_u) + p(F'_v)} (w_{uv} - \sum_{S \in \Omega; \delta(S) \ni uv} \pi_S), \tag{109}$$

where $p(S) := 1$ if $S \in \Omega$ and $p(S) := 0$ if $S \notin \Omega$. Let ϵ be the value of (109) when $uv = e^*$. Add ϵ to π_S for each component S of F' that is in Ω and replace F' by $F' \cup e^*$. This basic step is repeated until no component of F' is in Ω. Then F' contains a unique T-join, which is the output of the heuristic.

The heuristic can be implemented $O(|V|^2 \log |V|)$ time. Note that when $|T| = 2$, so when the T-join problem is a shortest path problem, the heuristic T-join is in

fact a shortest path. The heuristic also applies to other minimum weight forest problems with side constraints [see Goemans & Williamson, 1992].

Grigoriadis & Kalantari [1988] developed an $O(|V|^2)$ heuristic that constructs a matching with weight at most $2(|V|^{\log_3 7/3})$ times the optimum weight. Given a matching M, let G_M denote the 1-nearest neighbor graph of $G|(\exp(M))$. Begin with the empty matching M. In each component G_i of G_M choose a tour visiting each edge twice. Shortcut the tour to obtain a traveling salesman tour T_i of G_i. Greedily select a matching M_i of small weight from T_i (thus $|M_i| \geq \frac{1}{3}|T_i|$) and add it to M. Repeat the procedure until M is perfect.

The final matching heuristic we describe is due to Jünger & Pulleyblank [1991]. It runs in $O(|V| \log |V|)$ on Euclidean problems. Given a set of points in the plane, construct a complete graph G with a node for each point and let the length of each edge be the Euclidean distance between the corresponding points. So each node u has two coordinates u_1 and u_2 and each edge uv has weight (or length) $w_{uv} := \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2}$. Construct a matching in G as follows.

Let F be a minimum weight spanning tree in G. (The maximum degree of a node in T is five [see Jünger & Pulleyblank, 1991].)

DECOMPOSE: If $|V| \leq 6$, find a minimum weight matching in G. Otherwise, T has a non-pendant edge (i.e., an edge not incident to a node of degree 1). Let uv be a maximum weight non-pendant edge in T, then $T \setminus \{uv\}$ consists of two trees: T_u containing u and T_v containing v. We consider two cases:

Both T_u and T_v contain an even number of nodes: Apply DECOMPOSE, recursively, to $G|V(T_u)$ and T_u, and to $G|V(T_v)$ and T_v. Note that T_u is a minimum spanning tree in $G|V(T_u)$ and T_v is a minimum spanning tree in $G|V(T_v)$. Return $M_u \cup M_v$, where M_u is the matching constructed in $G|V(T_u)$ and M_v is the matching constructed in $G|V(T_v)$.

Both T_u and T_v contain an odd number of nodes: Apply DECOMPOSE to $G|(V(T_u) \cup \{v\})$ and $T_u \cup \{uv\}$ (which is again a minimum spanning tree) to construct a matching M_u. Let x be the node matched to u in M_u and choose $y \in V(T_v)$ with w_{xy} minimum. Then $T_v \cup \{xy\}$ is a minimum spanning tree in $G|(V(T_v) \cup \{x\})$. Applying DECOMPOSE again yields a matching M_v in $G|(V(T_v) \cup \{x\})$. Return $(M_u \setminus \{ux\}) \cup M_v$.

Note that the heuristic computes only one minimum spanning tree and the minimum spanning trees for the decomposed problems are easily obtained from it. Jünger & Pulleyblank [1991] also give a heuristic for finding a dual feasible solution, again based on minimum spanning tree calculations.

We conclude with a result of Grigoriadis & Kalantari [1986]: The running time of a heuristic for the Euclidean matching problem that finds a matching of weight at most $f(|V|)$ times the minimum weight, can be bounded from below by a constant times $|V| \log |V|$. If the heuristic yields a matching of weight at most $f(|V|)$ times the minimum weight for all matching problems, its running time is at least a constant times $|V|^2$.

Acknowledgements

I would like to thank Michele Conforti, Jack Edmonds, Mike Plummer, Bill Pulleyblank, Lex Schrijver, Leen Stougie and John Vande Vate for many helpful comments. John Vande Vate made a tremendous, and highly appreciated, effort editing the paper; improving its English as well as its organization. Needless to say that all remaining failings are on my account.

References

Ageev, A.A., A.V. Kostochka and Z. Szigeti (1994). *A characterization of Seymour graphs*, preprint.

Aho, A.V., J.E. Hopcroft and J.D. Ullman (1974). *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA.

Ahuja, R.K., T.L. Magnanti and J.B. Orlin (1989). Network flows, in: G.L. Nemhauser, A.H.G. Rinnooy Kan and M.J. Todd (eds.), *Optimization*, Handbooks in Operations Research and Management Science, Vol. 1, North-Holland, Amsterdam, pp. 211–369.

Alt, H., N. Blum, K. Mehlhorn and M. Paul (1991). Computing a maximum cardinality matching in a bipatite graph in time $O(n^{1.5}\sqrt{m/\log n})$, *Inf. Process. Lett.* 37, 237–240.

Anstee, R.P. (1985). An algorithmic proof of Tutte's f-factor theorem. *J. Algorithms* 6, 112–131.

Anstee, R.P. (1987). A polynomial algorithm for b-matchings: an alternative approach. *Inf. Process. Lett.* 24, 153–157.

Applegate, D., and W. Cook (1993). Solving large-scale matching problems, in: D.S. Johnson and C.C. McGeoch (eds.), *Network Flows and Matchings: First DIMACS Implementation Challenge*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 12, American Mathematical Society, Providence, RI, pp. 557–576.

Aráoz, J., W.H. Cunningham, J. Edmonds and J. Green-Krótki (1983). Reductions to 1-matching polyhedra. *Networks* 13, 455–473.

Avis, D. (1978). Two greedy heuristics for the weighted matching problem. *Congr. Numerantium* XXI, 65–76.

Avis, D. (1981). Worst case bounds for the Euclidean matching problem. *Comput. Math. Appl.* 7, 251–257.

Avis, D. (1983). A survey of heuristics for the weighted matching problem. *Networks* 13, 475–493.

Avis, D., B. Davis and J.M. Steele (1988). Probabilistic analysis for a greedy heuristic for Euclidean matching. *Probab. Eng. Inf. Sci.* 2, 143–156.

Balas, E., and W. Pulleyblank (1983). The perfectly matchable subgraph polytope of a bipartite graph. *Networks* 13, 495–516.

Balas, E., and W.R. Pulleyblank (1989). The perfectly matchable subgraph polytope of an arbitrary graph. *Combinatorica* 9, 321–337.

Balinski, M.L. (1965). Integer programming: methods, uses and computation. *Manage. Sci.* 12 (A), 253–313.

Balinski, M.L. (1969). Labeling to obtain a maximum matching (with discussion), in: R.C. Bose and T.A. Dowling (eds.), *Combinatorial Mathematics and its Applications*, The University of North California Monograph Series in Probability and Statistics, No. 4, University of North California Press, Chapel Hill, pp. 585–602.

Balinski, M.L. (1972). Establishing the matching polytope. *J. Comb. Theory, Ser. B* 13, 1–13.

Balinski, M.L., and R.E. Gomory (1964). A primal method for the assignment and transportation problems. *Manage. Sci.* 10, 578–593.

Balinski, M.L., and J. Gonzalez (1991). Maximum matchings in bipartite graphs via strong spanning trees. *Networks* 21, 165–179.

Ball, M.O., L.D. Bodin and R. Dial (1983). A matching based heuristic for scheduling mass transit crews and vehicles. *Transp. Sci.* 17, 4–31.

Ball, M.O., and U. Derigs (1983). An analysis of alternative strategies for implementing matching algorithms. *Networks* 13, 517–549.

Ball, M.O., U. Derigs, C. Hilbrand and A. Metz (1990). Matching problems with generalized upper bound side constraints. *Networks* 20, 703–721.

Barahona, F. (1980). *Application de l'Optimisation Combinatoire à Certains modèles de Verres de Spins: Complexité et Simulation*, Master's thesis, Université de Grenoble, France.

Barahona, F. (1990). Planar multicommodity flows, max cut and the Chinese postman problem, in: W. Cook and P.D. Seymour (eds), *Polyhedral Combinatorics*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 1, American Mathematical Society, Providence, RI, pp. 189–202.

Barahona, F. (1993a). On cuts and matchings in planar graphs. *Math. Program.* 60, 53–68.

Barahona, F. (1993b). Reducing matching to polynomial size linear programming. *SIAM J. Opt.* 3, 688–695.

Barahona, F., R. Maynard, R. Rammal and J.P. Uhry (1982). Morphology of ground states of a two-dimensional frustration model. *J. Phys. A: Mathematical and General* 15, 673–699.

Bartholdi III, J.J., and L.K. Platzman (1983). A fast heuristic based on spacefilling curves for minimum-weight matching in the plane. *Inf. Process. Lett.* 17, 177–188.

Bartnik, G.W. (1978). Algorithmes de couplages dans les graphes, Thèse Doctorat 3^e cycle, Unversité Paris VI.

Belck, H.-B. (1950). Reguläre Faktoren von Graphen. *J. Reine Angew. Math.* 188, 228–252.

Bellman, R. (1958). On a routing problem. *Q. Appl. Math.* 16, 87–90.

Berge, C. (1957). Two theorems in graph theory. *Proc. Nat. Acad. Sci. U.S.A.* 43, 842–844.

Berge, C. (1958). Sur le couplage maximum d'un graphe. *C.R. Acad. Sci., Sér. I (Mathematique)* 247, 258–259.

Berge, C. (1962). Sur une conjecture relative au problème des codes optimaux, Commun., 13éme Assemblée Générale de l'URSI, Tokyo.

Berge, C. (1985). *Graphs*, North-Holland, Amsterdam [revised edition of first part of: C. Berge, *Graphs and Hypergraphs*, North-Holland, Amsterdam, 1973].

Berstekas, D.P., D.A. Castañon, J. Eckstein and S.A. Zenion (1995). Parallel computing in network optimization, in: M.O. Ball, T.L. Magnanti, C. Monma and G.L. Nemhauser (eds.), *Network Models*, Handbooks in Operations Research and Management Science, Vol. 7, North-Holland, Amsterdam, Chapter 5, pp. 331–400, this volume.

Bertsekas, D.P. (1979). A distributed algorithm for the assignment problem, Working paper, Laboratory for Information and Decision Systems, M.I.T., Cambridge, MA.

Bertsekas, D.P. (1990). The auction algorithm for assignment and other network flow problems: a tutorial. *Interfaces* 20(4), 133–149.

Birkhoff, G. (1946). Tres observaciones sobre el algebra lineal. *Rev. Fac. Cie. Exactas Puras Apl. Univ. Nac. Tucuman, Ser. A (Matematicas y Fisica Teoretica)* 5, 147–151.

Blum, N. (1990a). A new approach to maximum matching in general graphs (extended abstract), in: M.S. Paterson (ed.), *Proc. 17th Int. Colloq. on Automata, Languages and Programming*, Lecture Notes in Computer Science, Vol. 443, Springer-Verlag, Berlin, pp. 586–597.

Blum, N. (1990b). *A New Approach to Maximum Matching in General Graphs*, Report No. 8546-CS, Institut für Informatik der Universität Bonn.

Bondy, J.A., and U.S.R. Murty (1976). *Graph theory with Applications*, MacMillan Press, London.

Borůvka, O. (1926). O jistém problému minimálním. *Práca Moravské Přírodovědecké Společnosti* 3, 37–48 (in Czech).

Bourjolly, J.-M., and W.R. Pulleyblank (1989). König-Egerváry graphs, 2-bicritical graphs and fractional matchings. *Discrete Appl. Math.* 24, 63–82.

Brezovec, C., G. Cornuéjols and F. Glover (1988). A matroid algorithm and its application to the efficient solution of two optimization problems on graphs. *Math. Program.* 42, 471–487.

Broder, A.Z. (1986). How hard is it to marry at random? (on the approximation of the permanent), in: *Proc. 18th Annual ACM Symp. on Theory of Computing*, Association for Computing Machinery, New York, NY, pp. 50–58 [Erratum in: *Proc. 20th ACM Symp. on Theory of Computing*,

1988, Association for Computing Machinery, New York, p. 551].

Brualdi, R.A., and P.M. Gibson (1977). Convex polyhedra of doubly stochastic matrices I. Applications of the permanent function. *J. Comb. Theory, Ser. A* 22, 194–230.

Burkard, R.E., and U. Derigs (1980). *Assignment and Matching Problems: Solution Methods with FORTRAN-Programs*, Lecture Notes in Economics and Mathematical Systems, Vol. 184, Springer-Verlag, Berlin, Heidelberg.

Burlet, M., and A.V. Karzanov (1993). *Minimum Weight T, d-Joins and Multi-Joins*, Rapport de Recherche RR929-M, Laboratoire ARTEMIS, Université Joseph Fourier, Grenoble.

Chari, S., P. Rohatgi and A. Srinivasan (1993). Randomness-optimal unique element isolation, with applications to perfect matching and related problems, preprint.

Christofides, N. (1976). *Worst-case Analysis of a New Heuristic for the Travelling Salesman Problem*, Technical report, GSIA Carnegie-Mellon University, Pittsburgh, Pennsylvania.

Cook, S.A. (1971). The complexity of theorem-proving procedures, in: *Proc. 3rd Annual ACM Symp. on Theory of Computing*, Association for Computing Machinery, New York, NY, pp. 151–158.

Cook, W. (1983a). A minimal totally dual integral defining system for the b-matching polyhedron. *SIAM J. Algebraic Discrete Methods* 4, 212–220.

Cook, W. (1983b). *On some Aspects of Totally Dual Integral Sytems*, PhD thesis, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario.

Cook, W. and W.R. Pulleyblank (1987). Linear systems for constrained matching problems. *Math. Oper. Res.* 12, 97–120.

Cornuéjols, G. (1988). General factors of graphs. *J. Comb. Theory, Ser. B* 45, 185–198.

Cornuéjols, G., and D. Hartvigsen (1986). An extension of matching theory. *J. Comb. Theory, Ser. B* 40, 285–296.

Cornuéjols, G., D. Hartvigsen and W. Pulleyblank (1982). Packing subgraphs in a graph. *Oper. Res. Lett.* 1, 139–143.

Cornuéjols, G., and W. Pulleyblank (1980a). A matching problem with side conditions. *Discrete Math.* 29, 135–159.

Cornuéjols, G., and W.R. Pulleyblank (1980b). Perfect triangle-free 2-matchings. *Math. Program. Study* 13, 1–7.

Cornuéjols, G., and W. Pulleyblank (1982). The travelling salesman polytope and $\{0,2\}$-matchings. *Ann. Discrete Math.* 16, 27–55.

Cornuéjols, G., and W.R. Pulleyblank (1983). Critical graphs, matchings and tours or a hierarchy of relaxations for the travelling salesman problem. *Combinatorica* 3, 35–52.

Crocker, S.T. (1993). An experimental comparison on two maximum cardinality matching programs, in: D.S. Johnson and C.C. McGeoch (eds), *Network Flows and Matchings: First DIMACS Implementation Challenge*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 12, American Mathematical Society, Providence, RI, pp. 519–537.

Csansky, L. (1976). Fast parallel matrix inversion algorithms. *SIAM J. Comp.* 5, 618–623.

Cunningham, W.H., and J. Green-Krotki (1986). Dominants and submissives of matching polyhedra. *Math. Program.* 36, 228–237.

Cunningham, W.H., and J. Green-Krótki (1991). b-Matching degree-sequence polyhedra. *Combinatorica* 11, 219–230.

Cunningham, W.H., and J. Green-Krótki (1994). A separation algorithm for the matchable set polytope, *Math. Program.* 65, 139–190.

Cunningham, W.H., and A.B. Marsh III (1978). A primal algorithm for optimum matching, in: M.L. Balinski and A.J. Hoffman (eds.), *Polyhedral Combinatorics (dedicated to the memory of D.R. Fulkerson)*, *Mathematical Programming Study* 8, North-Holland, Amsterdam, pp. 50–72.

Cunningham, W.H., and F. Zhang (1992). Subgraph degree-sequence polyhedra, in: E. Balas, G. Cornuéjols and R. Kannan (eds.), *Integer Programming and Combinatorial Optimization*, Proc. Conf. of the Mathematical Progamming Society, Carnegie-Mellon University, May 25–27, 1992, pp. 246–259.

Dahlhaus, E., and M. Karpinski (1988). Parallel construction of perfect matchings and Hamiltonian cycles on dense graphs. *Theor. Comput. Sci.* 61, 121–136.

Dantzig, G.B. (1951). Maximization of a linear function of variables subject to linear inequalities, in: Tj.C. Koopmans (ed.), *Activity Analysis of Production and Allocation*, John Wiley, New York, NY, pp. 339–347.

Deming, R.W. (1979). Independence numbers of graphs – an extension of the Koenig-Egervary theorem. *Discrete Math.* 27, 23–33.

Derigs, U. (1981). A shortest augmenting path method for solving minimal perfect matching problems. *Networks* 11, 379–390.

Derigs, U. (1986a). A short note on matching algorithms. *Math. Program. Study* 26, 200–204.

Derigs, U. (1986b). Solving large-scale matching problems efficiently: a new primal matching approach. *Networks* 16, 1–16.

Derigs, U. (1988a). *Programming in Networks and Graphs*, Lecture Notes in Economics and Mathematical Systems, Vol. 300, Springer-Verlag, Berlin.

Derigs, U. (1988b). Solving non-bipartite matching problems via shortest path techniques. *Ann. Oper. Res.* 13, 225–261.

Derigs, U., and A. Metz (1986). On the use of optimal fractional matchings for solving the (integer) matching problem. *Computing* 36, 263–270.

Derigs, U., and A. Metz (1991). Solving (large scale) matching problems combinatorially. *Math. Program.* 50, 113–121.

Derigs, U., and A. Metz (1992). A matching-based approach for solving a delivery/pick-up vehicle routing problem with time constraints. *Oper. Res. Spektrum* 14, 91–106.

Devine, M.D. (1973). A model for minimizing the cost of drilling dual completion oil wells. *Manage. Sci.* 20, 532–535.

Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. *Numer. Math.* 1, 269–271.

Dilworth, R.P. (1950). A decomposition theorem for partially ordered sets. *Ann. Math.* (2) 51, 161–166.

Dinic, E.A. (1970). Algorithm for solution of a problem of maximum flow in a network with power estimation (in Russian). *Dokl. Akad. Nauk SSSR* 194, 745–757 [English translation: *Soviet Mathemathics Doklady*, 11, 1277–1280].

Dulmage, A.L., and N.S. Mendelsohn (1958). Coverings of bipartite graphs. *Can. J. Math.* 10, 517–534.

Dulmage, A.L., and N.S. Mendelsohn (1959). A structure theory of bipartite graphs of finite exterior dimension. *Trans. R. Soc. Can., Ser. III* 53, 1–13.

Dulmage, A.L., and N.S. Mendelsohn (1967). Graphs and matrices, in: F. Harary (ed.), *Graph Theory and Theoretical Physics*, Academic Press, New York, NY, pp. 167–277.

Edmonds, J. (1965a). The Chinese postman's problem. *Bull. Oper. Res. Soc.* 13, B–73.

Edmonds, J. (1965b). Maximum matching and a polyhedron with 0,1-vertices. *J. Res. Nat. Bur. Stand. – B. Math. Math. Phys.* 69B, 125–130.

Edmonds, J. (1965c). Paths, trees and flowers. *Can. J. Math.* 17, 449–467.

Edmonds, J. (1967). Systems of distinct representatives and linear algebra. *J. Res. Nat. Bur. Stand. – B. Math. Math. Phys.* 71B, 241–245.

Edmonds, J. (1970). Submodular functions, matroids, and certain polyhedra, in: R. Guy, H. Hanani, N. Sauer and J. Schönheim (eds.), *Combinatorial Structures and their Applications*, Gordon and Breach, New York, NY, pp. 69–87.

Edmonds, J., and R. Giles (1977). A min–max relation for submodular functions on graphs. *Ann. Discrete Math.* 1, 185–204.

Edmonds, J., and E. Johnson (1970). Matching: a well-solved class of integer linear programs, in: R. Guy, H. Hanani, N. Sauer and J. Schönheim (eds.), *Combinatorial Structures and their Applications*, Gordon and Breach, New York, NY, pp. 89–92.

Edmonds, J., E.L. Johnson and S.C. Lockhart (1969). *Blossom I, a Code for Matching*, unpublished report, IBM T.J. Watson Research Center, Yorktown Heights, NY.

Edmonds, J., and E. L. Johnson (1973). Matching, Euler tours and the Chinese postman. *Math. Program.* 5, 88–124.

Edmonds, J., and R.M. Karp (1970). Theoretical improvements in algorithmic efficiency for network flow problems, in: R. Guy, H. Hanani, N. Sauer and J. Schönheim (eds.), *Combinatorial Structures and their Applications*, Gordon and Breach, New York, NY, pp. 93–96.

Edmonds, J., and R.M. Karp (1972). Theoretical improvements in algorithmic efficiency for network flow problems. *J. Assoc. Comput. Mach.* 19, 248–264.

Edmonds, J., L. Lovász and W.R. Pulleyblank (1982). Brick decompositions and the matching rank of graphs. *Combinatorica* 2, 247–274.

Egerváry, E. (1931). Matrixok kombinatorius tulajdonságairol (in Hungarian). *Matematikai és Fizikai Lapok* 38, 16–28.

Elias, P., A. Feinstein and C.E. Shannon (1956). Note on the maximum flow through a network. *IRE Trans. Inf. Theory* IT 2, 117–119.

Erdös, P., and T. Gallai (1960). Gráfok elöírt fokú pontokkal (in Hungarian). *Mat. Lapok* 11, 264–274.

Euler, L. (1736). Solutio problematis ad geometriam situs pertinentis. *Comment. Acad. Sci. Imp. Petropolitanae* 8, 128–140.

Even, S., and O. Kariv (1975). An $O(n^{2.5})$ algorithm for maximum matching in general graphs, in: *Proc. 16th Annual Symp. on Foundations of Computer Science*, IEEE, New York, NY, pp. 100–112.

Even, S., and R.E. Tarjan (1975). Network flow and testing graph connectivity. *SIAM J. Comput.* 4, 507–518.

Feder, T., and R. Motwani (1991). Clique partitions, graph compression and speeding-up algorithms, in: *Proc. 23rd Annual ACM Symp. on Theory of Computing*, Association for Computing Machinery, New York, NY, pp. 123–133.

Flood, M.M. (1956). The traveling-salesman problem. *Oper. Res.* 4, 61–75.

Floyd, R.W. (1962a). Algorithm 96: ancestor. *Commun. Assoc. Comput. Mach.* 5, 344–345.

Floyd, R.W. (1962b). Algorithm 97: shortest path. *Commun. Assoc. Comput. Mach.* 5, 345.

Ford Jr., L.R. (1956). *Network Flow Theory*, Paper P-923, RAND Corporation, Santa Monica, CA.

Ford Jr., L.R., and D.R. Fulkerson (1956). Maximal flow through a network. *Can. J. Math.* 8, 399–404.

Ford Jr., L.R., and D.R. Fulkerson (1957). A simple algorithm for finding maximal network flows and an application to the Hitchcock problem. *Can. J. Math.* 9, 210–218.

Frank, A. (1990). Packing paths, circuits and cuts – a survey, in: B. Korte, L. Lovász, H.J. Prömel and A. Schrijver (eds.), *Paths, Flows and VLSI-Layout*, Springer-Verlag, Berlin, Heidelberg, pp. 47–100.

Frank, A. (1993). Conservative weightings and ear-decompositions of graphs. *Combinatorica* 13, 65–81.

Frank, A., A. Sebö and É. Tardos (1984). Covering directed and odd cuts. *Math. Program. Study* 22, 99–112.

Frank, A., and Z. Szigeti (1994). On packing T-cuts, *J. Comb. Theory, Ser. B* 61, 263–271.

Fredman, M.L., and R.E. Tarjan (1987). Fibonacci heaps and their uses in improved network optimization algorithms. *J. Assoc. Comput. Mach.* 34, 596–615.

Frieze, A., C. McDiarmid and B. Reed (1990). Greedy matching on the line. *SIAM J. Comput.* 19, 666–672.

Frobenius, G. (1912). Über Matrizen aus nicht negativen Elementen. *Sitszungsberichte der königlich preussischen Akademie der Wissenschaften zu Berlin*, 456–477.

Frobenius, G. (1917). Über zerlegbare Determinanten. *Sitszungsberichte der königlich preussischen Akademie der Wissenschaften zu Berlin*, 274–277.

Fujii, M., T. Kasami and N. Ninomiya (1969). Optimal sequencing of two equivalent processors. *SIAM J. Appl. Math.* 17, 784–789 [Erratum in: *SIAM J. Appl. Math.* 20 (1971), 141].

Fulkerson, D.R. (1961). An out-of-kilter method for minimal cost flow problems. *SIAM J. Appl. Math.* 9, 18–27.

Gabow, H.N. (1973). *Implementation of Algorithms for Maximum Matching on Non-bipartite Graphs*, PhD thesis, Stanford University, Department of Computer Science, 1973.

Gabow, H.N. (1976). An efficient implementation of Edmonds' algorithm for maximum matching on graphs. *J. Assoc. Comput. Mach.* 23, 221–234.

Gabow, H.N. (1983). An efficient reduction technique for degree-constraint subgraph and bidirected network flow problems, in: *Proc. 15th Annual ACM Symp. on Theory of Computing*, Association for Computing Machinery, New York, NY, pp. 448–456.

Gabow, H.N. (1985). A scaling algorithm for weighted matching on general graphs, in: *Proc. 26th Annual Symp. on Foundations of Computer Science*, IEEE, New York, NY, pp. 90–100.

Gabow, H.N. (1990). Data structures for weighted matching and nearest common ancestors with linking, in: *Proc. 1st Annual ACM–SIAM Symp. on Discrete Algorithms*, Association for Computing Machinery, New York, NY, pp. 434–443.

Gabow, H.N., Z. Galil and T.H. Spencer (1989). Efficient implementation of graph algorithms using contraction. *J. Assoc. Comput. Mach.* 36, 540–572.

Gabow H.N., and R.E. Tarjan (1983). A linear-time algorithm for a special case of disjoint set union, in: *Proc. 15th Annual ACM Symp. on Theory of Computing*, Association for Computing Machinery, New York, NY, pp. 246–251.

Gabow, H.N., and R.E. Tarjan (1991). Faster scaling algorithms for general graph-matching problems. *J. Assoc. Comput. Mach.* 38, 815–853.

Gale, D., H.W. Kuhn and A.W. Tucker (1951). Linear programming and the theory of games, in: Tj.C. Koopmans (ed.), *Activity Analysis of Production and Allocation*, New York, NY, pp. 317–329.

Gale, D., and L.S. Shapley (1962). College admissions and the stability of marriage. *Am. Math. Mon.* 69, 9–15.

Galil, Z. (1986a). Efficient algorithms for finding maximum matching in graphs. *ACM Comput. Surv.* 18, 23–38.

Galil, Z. (1986b). Sequential and parallel algorithms for finding maximum matchings in graphs. *Annu. Rev. Comput. Sci.* 1, 197–224.

Galil, Z., S. Micali and H. Gabow (1986). An $O(EV \log V)$ algorithm for finding a maximal weighted matching in general graphs. *SIAM J. Comput.* 15, 120–130.

Gallai, T. (1950). On factorisation of graphs. *Acta Math. Acad. Sci. Hung.* 1 133–153.

Gallai, T. (1959). Über extreme Punkt- und Kantenmengen. *Ann. Univ. Sci. Budap. Rolando Eötvös Nominatae, Sect. Math.* 2, 133–138.

Gallai, T. (1963). Kritische Graphen II. *Mag. Tud. Akad. Mat. Kut. Intéz. Közl.* 8, 373–395.

Gallai, T. (1964). Maximale Systeme unabhängiger Kanten. *Mag. Tud. Akad. Mat. Kut. Intéz. Közl.* 9, 401–413.

Gamble, A.R. (1989). *Polyhedral Extentions of Matching Theory*, PhD thesis, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario.

Gerards, A.M.H. (1991). Compact systems for T-join and perfect matching polyhedra of graphs with bounded genus, *Oper. Res. Lett.* 10, 377–382.

Gerards, A.M.H. (1992). On shortest T-joins and packing T-cuts. *J. Comb. Theory, Ser. B* 55, 73–82.

Giles, R. (1982a). Optimum matching forests I: special weights. *Math. Program.* 22, 1–11.

Giles, R. (1982b). Optimum matching forests II: general weights. *Math. Program.* 22, 12–38.

Giles, R. (1982c). Optimum matching forests III: facets of matching forest polyhedra. *Math. Program.* 22, 39–51.

Goemans, M.X., and D.P. Williamson (1992). A general approximation technique for constrained forest problems, in: *Proc. 3rd Annual ACM–SIAM Symp. on Discrete Algorithms*, Association for Computing Machinery, New York, NY, pp. 307–316.

Goldberg, A.V., S.A. Plotkin, D.B. Shmoys and E. Tardos (1992). Using interior-point methods for fast parallel algorithms for bipartite matching and related problems. *SIAM J. Comput.* 21, 140–150.

Goldberg, A.V., S.A. Plotkin and P.M. Vaidya (1993). Sublinear-time parallel algorithms for matching and related problems. *J. Algorithms* 14, 180–213.

Goldberg, A.V., É. Tardos and R.E. Tarjan (1990). Network flow algorithms, in: B. Korte, L. Lovász, H.J. Prömel and A. Schrijver (eds.), *Paths, Flows and VLSI-Layout*, Springer-Verlag,

Berlin, Heidelberg, pp. 101–164.
Goldberg, A.V., and R.E. Tarjan (1989). Finding minimum-cost circulations by canceling negative cycles. *J. Assoc. Comput. Mach.* 36, 873–886.
Gondran, M., and M. Minoux (1984). *Graphs and Algorithms*, Wiley/Interscience, New York, NY.
Grigoriadis, M.D., and B. Kalantari (1986). A lower bound to the complexity of Euclidean and rectilinear matching algorithms. *Inf. Process. Lett.* 22, 73–76.
Grigoriadis, M.D., and B. Kalantari (1988). A new class of heuristic algorithms for weighted perfect matching. *J. Assoc. Comput. Mach.* 35, 769–776.
Grigoriadis, M.D., B. Kalantari and C.Y. Lai (1986). On the existence of weakly greedy matching heuristics. *Oper. Res. Lett.* 5, 201–205.
Grigoriev, D.Y., and M. Karpinski (1987). The matching problem for bipartite graphs with polynomially bounded permanents is in NC, in: *28th Annual Symposium on Foundations of Computer Science*, IEEE, New York, NY, pp. 166–172.
Grötschel, M., and O. Holland (1985). Solving matching problems with linear programming. *Math. Program.* 33, 243–259.
Grötschel, M., L. Lovász and A. Schrijver (1981). The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* 1, 169–197 [corrigendum in: *Combinatorica* 4 (1984), 291-295].
Grötschel, M., L. Lovász and A. Schrijver (1984). Polynomial algorithms for perfect graphs. *Ann. Discrete Math.* 21, 325–356.
Grötschel, M., L. Lovász and A. Schrijver (1988). *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, Berlin.
Grötschel, M., and M.W. Padberg (1985). Polyhedral theory, in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys (eds.), *The Traveling Salesman Problem, A Guided tour of Combinatorial Optimization*, John Wiley and Sons, Chichester, pp. 251–305.
Grötschel, M., and W.R. Pulleyblank (1981). Weakly bipartite graphs and the max-cut problem. *Oper. Res. Lett.* 1, 23–27.
Grover, L.K. (1992). Fast parallel algorithms for bipartite matching, in: E. Balas, G. Cornuéjols, and R. Kannan (eds.). *Integer Programming and Combinatorial Optimization*, Proc. Conf. of the Mathematical Programming Society, Carnegie-Mellon University, May 25–27, 1992, pp. 367–384.
Gusfield, D., and R.W. Irving (1989). *The Stable Marriage Problem: Structure and Algorithms*, MIT Press, Cambridge, Massachusetts.
Hadlock, F. (1975). Finding a maximum cut of a planar graph in polynomial time. *SIAM J. Comput.* 4, 221–225.
Hall Jr., M. (1956). An algorithm for distinct representatives. *Am. Math. Mon.* 716–717.
Hall, P. (1935). On representatives of subsets. *J. Lond. Math. Soc.* 10, 26–30.
Helgason, R.V., and J.L. Kennington (1995). Primal simplex algorithms for minimum cost network flows, in: M.O. Ball, T.L. Magnanti, C. Monma and G.L. Nemhauser (eds.), *Network Models*, Handbooks in Operations Research and Management Science, Vol. 7, North-Holland, Amsterdam, Chapter 2, pp. 85–134, this volume.
He, X. (1991). An efficient parallel algorithm for finding minimum weight matching for points on a convex polygon, *Inf. Process. Lett.* 37, 111–116.
Hetyei, G. (1964). 2 × 1-es téglalapokkal lefedhetö idomokról (in Hungarian). *Pécsi Tanárképzö Föiskola Tud. Közl.* 8, 351–367
Hoffman, A.J. (1974). A generalization of max flow-min cut. *Math. Program.* 6, 352–359.
Hoffman, A.J., and J.B. Kruskal (1956). Integral boundary points of convex polyhedra, in: H.W. Kuhn and A.W. Tucker (eds.). *Linear Inequalities and Related Systems*, Annals of Mathematical Studies, Vol. 38, Princeton University Press, Princeton, NJ, pp. 223–246.
Hoffman, A.J., and R. Oppenheim (1978). Local unimodularity in the matching polytope. *Ann. Discrete Math.* 2, 201–209.
Holyer, I. (1981). The NP-completeness of edge-coloring. *SIAM J. Comput.* 10, 718–720.
Hopcroft, J.E., and R.M. Karp (1971). An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs, in: *Conf. Record 1971 12th Annual Symp. on Switching and Automata Theory*, IEEE, New

York, NY, pp. 122–125.

Hopcroft, J.E., and R.M. Karp (1973). An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.* 2, 225–231.

Imai, H. (1986). Worst-case analysis for planar matching and tour heuristics with bucketing techniques and spacefilling curves. *J. Oper. Res. Soc. Jap.* 29, 43–67.

Imai, H., H. Sanae and M. Iri (1984). A planar-matching heuristic by means of triangular buckets, in: *Proc. 1984 Fall Conf. of the Operations Research Society of Japan*, 2-D-4, pp. 157–158 (in Japanese).

Iri, M., K. Murota and S. Matsui (1981). Linear-time approximation algorithms for finding the minimum-weight perfect matching on a plane. *Inf. Process. Lett.* 12, 206–209.

Iri, M., K. Murota and S. Matsui (1982). An approximate solution for the problem of optimizing the plotter pen movement, in: R.F. Drenick and F. Kozin (eds.), *System Modeling and Optimization*, Proc. 10th IFIP Conf., New York, 1981, Lecture Notes in Control and Information Sciences, Vol. 38, Springer-Verlag, Berlin, pp. 572–580.

Iri, M., K. Murota and S. Matsui (1983). Heuristics for planar minimum-weight perfect matchings. *Networks* 13, 67–92.

Iri, M., and A. Taguchi (1980). The determination of the pen-movement of an XY-plotter and its computational complexity, in: *Proc. 1980 Spring Conf. of the Operations Research Society of Japan*, P-8, pp. 204–205 (in Japanese).

Irving, R.W. (1985). An efficient algorithm for the "stable roommates" problem. *J. Algorithms* 6, 577–595.

Jarník, V. (1930). O jistém problému minimálním (in Czech). *Práca Moravské Přírodovědecké Společnosti* 6, 57–63

Jerrum, M., and A. Sinclair (1989). Approximating the permanent. *SIAM J. Comput.* 18, 1149–1178.

Jünger, M., and W. Pulleyblank (1991). New primal and dual matching heuristics, Report No 91.105, Institut für Informatik, Universität zu Köln.

Jünger, M., G. Reinelt and G. Rinaldi (1995). The traveling salesman problem, in: M.O. Ball, T.L. Magnanti, C. Monma and G.L. Nemhauser (eds.), *Network Models*, Handbooks in Operations Research and Management Science, Vol. 7, North-Holland, Amsterdam, Chapter 4, pp. 225–330, this volume.

Kameda, T., and I. Munro (1974). An $O(|V| \cdot |E|)$ algorithm for maximum matching of graphs. *Computing* 12, 91–98.

Kariv, O. (1976). *An $O(n^{5/2})$ Algorithm for Maximum Matching in General Graphs*, PhD thesis, Weizman Institute of Science, Rehovot.

Karloff, H.J. (1986). A Las Vegas RNC algorithm for maximum matching. *Combinatorica* 6, 387–391.

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica* 4, 373–395.

Karp, R.M. (1972). Reducibility among combinatorial problems, in: R.E. Miller and J.W. Thatcher (eds.), *Complexity of Computer Computations*, Plenum Press, New York, NY, pp. 85–103.

Karp, R.M., and E. Upfal an A. Wigderson (1986). Constructing a perfect matching is in random NC. *Combinatorica* 6, 35–48.

Karp, R.M., and C. H. Papadimitriou (1982). On linear characterizations of combinatorial optimization problems. *SIAM J. Comput.* 11, 620–632.

Karp, R.M., and V. Ramachandran (1990). Parallel algorithms for shared-memory machines, in: J. van Leeuwen (ed.), *Handbook of Theoretical Computer Science, Vol. A: Algorithms and Complexity*, Elsevier, Amsterdam, pp. 869–941.

Karzanov, A. (1992). Determining the distance to the perfect matching polytope of a bipartite graph, preprint.

Kasteleyn, P.W. (1963). Dimer statistics and phase transitions. *J. Math. Phys.* 4, 287–293.

Kasteleyn, P.W. (1967). Graph theory and crystal physics, in: F. Harary (ed.), *Graph Theory and Theoretical Physics*, Academic Press, New York, NY, pp. 43–110.

Khachiyan, L.G. (1979). A polynomial algorithm in linear programming (in Russian). *Dokl. Akad. Nauk SSSR* 224, 1093–1096.

König, D. (1915). Vonalrendszerek és determinánsok (in Hungarian). *Mat. Természettudományi Értesitö* 33, 221–229.

König, D. (1916a). Graphok és alkalmazásuk a determinánsok és a halmazok elméletében (in Hungarian). *Mat. Természettudományi Értesitö* 34, 104–119.

König, D. (1916b). Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. *Math. Ann.* 77, 453–465.

König, D. (1931). Graphok és matrixok (in Hungarian). *Mat. Fizikai Lapok* 38, 116–119.

König, D. (1933). Über trennende Knotenpunkte in Graphen (nebst Anwendungen auf Determinanten und Matrizen). *Acta Litt. Sci. Regiae Univ. Hung. Francisco-Josephinae (Szeged), Sectio Sci. Math.* 6, 155–179.

König, D. (1936). *Theorie der endlichen und unendlichen Graphen*, Akademischen Verlagsgesellschaft, Leipzig [reprinted: Chelsea, New York, 1950, and Teubner, Leipzig, 1986].

Korach, E. (1982). *On Dual Integrality, Min-Max Equalities and Algorithms in Combinatorial Programming*, PhD thesis, Department of Combinatorics and Optimization. University of Waterloo, Waterloo, Ontario.

Koren, M. (1973). Extreme degree sequences of simple graphs. *J. Comb. Theory, Ser. B* 15, 213–234.

Kotzig, A. (1959a). Z teórie konečných grafov s lineárnym faktorom I (in Slovak). *Mat.-Fyz. Časopis Slovenskej Akad. Vied* 9, 73–91.

Kotzig, A. (1959b). Z teórie konečných grafov s lineárnym faktorom II (in Slovak). *Mat.-Fyz. Časopis Slovenskej Akad. Vied* 9, 136–159.

Kotzig, A. (1960). Z teórie konečných grafov s lineárnym faktorom III (in Slovak). *Mat.-Fyz. Časopis Slovenskej Akad. Vied* 10, 205–215

Kozen, D., U.V. Vazirani and V.V. Vazirani (1985). NC algorithms for comparibility graphs, interval graphs, and testing for unique perfect matching, in: S.N. Maheshwari (ed.). *Foundations of Software Technology and Theoretical Computer Science, Fifth Conference, New Delhi, 1985*, Lecture Notes in Computer Science, Vol. 206, Springer-Verlag, Berlin, pp. 496–503.

Kruskal, J.B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. Amer. Math. Soc.* 7, 48–50.

Kuhn, H.W. (1955). The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.* 2, 83–97.

Kuhn, H.W. (1956). Variants of the Hungarian method for assignment problems. *Nav. Res. Logist. Q.* 3, 253–258.

Kwan Mei-Ko (1962). Graphic programming using odd and even points. *Chin. Math.* 1, 273–277.

Lawler, E.L. (1976). *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, NY.

Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (1985). *The Traveling Salesman Problem, A Guided tour of Combinatorial Optimization*, John Wiley and Sons, Chichester.

Lessard, R., J.-M. Rousseau and M. Minoux (1989). A new algorithm for general matching problems using network flow subproblems. *Networks* 19, 459–479.

Lipton, R.J., and R.E. Tarjan (1979). A separator theorem for planar graphs. *SIAM J. Appl. Math.* 36, 177–189.

Lipton, R.J., and R.E. Tarjan (1980). Applications of a planar separator theorem. *SIAM J. Comput.* 9, 615–627.

Little, C.H.C. (1974). An extension of Kasteleyn's method for enumerating the 1-factors of planar graphs, in: D.A. Holton (ed.), *Combinatorial Mathematics*, Proc. 2nd Australian Conf., Lecture Notes in Mathematics, Vol. 403, Springer-Verlag, Berlin, pp. 63–72.

Lovász, L. (1970a). The factorization of graphs, in: R. Guy, H. Hanani, N. Sauer and J. Schönheim (eds.), *Combinatorial Structures and their Applications*, Gordon and Breach, New York, NY, pp. 243–246.

Lovász, L. (1970b). Generalized factors of graphs, in: P. Erdös, A. Rényi and V.T. Sós (eds.), *Combinatorial Theory and its Applications II*, Coloq. Math. Soc. János Bolyai, 4, North-Holland,

Amsterdam, pp. 773–781.
Lovász, L. (1970c). Subgraphs with prescribed valencies. *J. Comb. Theory* 8, 391–416.
Lovász, L. (1972a). The factorization of graphs II. *Acta Math. Acad. Sci. Hung.* 23, 223–246.
Lovász, L. (1972b). Normal hypergraphs and the perfect graph conjecture. *Discrete Math.* 2, 253–267.
Lovász, L. (1972c). A note on factor-critical graphs. *Stud. Sci. Math. Hung.* 7, 279–280.
Lovász, L. (1972d). On the structure of factorizable graphs. *Acta Math. Acad. Sci. Hung.* 23, 179–195.
Lovász, L. (1972e). On the structure of factorizable graphs, II. *Acta Math. Acad. Sci. Hung.* 23, 465–478.
Lovász, L. (1973). Antifactors of graphs. *Period. Math. Hung.* 4, 121–123.
Lovász, L. (1975). 2-matchings and 2-covers of hypergraphs. *Acta Math. Acad. Sci. Hung.* 26 (1975) 433–444.
Lovász, L. (1979a). Graph theory and integer programming. *Ann. Discrete Math.* 4, 141–158.
Lovász, L. (1979b). On determinants, matchings and random algorithms, in: L. Budach (ed.), *Fundamentals of Computation Theory*, FCT '79, Proc. Conf. on Algebraic, Arithmetic and Categorial Methods in Computation Theory, Akademie-Verlag, Berlin, pp. 565–574.
Lovász, L. (1983). Ear-decompositions of matching-covered graphs. *Combinatorica* 3, 105–117.
Lovász, L. (1987). Matching structure and the matching lattice. *J. Comb. Theory, Ser. B* 43, 187–222.
Lovász, L., and M.D. Plummer (1975). On bicritical graphs, in: A. Hajnal, R.Rado and V.T. Sós (eds.), *Infinite and Finite Sets, Vol. II*, North-Holland, Amsterdam, pp. 1051–1079.
Lovász, L., and M.D. Plummer (1986). *Matching Theory*, Akadémiai Kiadó, Budapest [also published as: *North-Holland Mathematics Studies* Vol. 121, North-Holland, Amsterdam, 1986].
Marcotte, O., and S. Suri (1991). Fast matching algorithms for points on a polygon. *SIAM J. Comput.* 20, 405–422.
Marsh III, A.B. (1979). *Matching Algorithms*, PhD thesis, The Johns Hopkins University, Baltimore.
Matsumoto, K., T. Nishizeki and N. Saito (1986). Planar multicommodity flows, maximum matchings and negative cycles. *SIAM J. Comput.* 15, 495–510.
Mattingly, R.B., and N.P. Ritchey (1993). Implementing an $O(\sqrt{N}M)$ cardinality matching algorithm, in: D.S. Johnson and C.C. McGeoch (eds.), *Network Flows and Matchings: First DIMACS Implementation Challenge*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 12, American Mathematical Society, Providence, RI, pp. 539–556.
Menger, K. (1927). Zur allgemeinen Kurventheorie. *Fundam. Math.* 10, 96–115.
Micali, S., and V.V. Vazirani (1980). An $O(\sqrt{|v|}|E|)$ algorithm for finding maximum matching in general graphs, in: *Proc. 21th Annual Symp. on Foundations of Computer Science*, IEEE, New York, NY, pp. 17–27.
Miller, G.L., and J. Naor (1989). Flow in planar graphs with multiple sources and sinks, in: *Proc. 30th Annual Symp. on Foundations of Computer Science*, IEEE, New York, NY, pp. 112–117.
Minc, H. (1978). *Permanents*, Addison-Wesley, Reading.
Minty, G.J. (1960). Monotone networks. *Proc. R. Soc. Lond.* 257, 194–212.
Minty, G.J. (1980). On maximal independent sets of vertices in claw-free graphs. *J. Comb. Theory, Ser. B* 28, 284–304.
Mirsky, L. (1971). *Transversal Theory*, Academic Press, London.
Middendorf, M., and F. Pfeiffer (1990). On the complexity of the disjoint paths problem (extended abstract), in: W. Cook and P.D. Seymour, (eds.), *Polyhedral Combinatorics*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 1, American Mathematical Society, Providence, RI, pp. 171–178.
Motzkin, T.S. (1956). The assignment problem, in: J.H. Curtiss (ed.), *Numerical Analysis*, Proc. Symp. in Applied Mathematics, Vol. IV, McGraw-Hill, New York, NY, pp. 109–125.
Mulder, H.M. (1992). Julius Petersen's theory of regular graphs. *Discrete Math.* 100, 157–175.
Mulmuley, K., U.V. Vazirani and V.V. Vazirani (1987). Matching is as easy as matrix inversion. *Combinatorica* 7, 105–113.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *J. Soc. Ind. Appl. Math.* 32–38.

Murota, K. (1993). *Combinatorial Relaxation Algorithm for the Maximum Degree of Subdeterminants: Computing Smith-McMillan Form at Infinity and Structural Indices in Kronecker Form*, RIMS-954, Research Institute for Mathematical Sciences, Kyoto University.

Murty, U.S.R. (1994). *The Matching Lattice and Related Topics*, preliminary report, University of Waterloo, Waterloo, Ontario.

Naddef, D. (1982). Rank of maximum matchings in a graph. *Math. Program.* 22, 52–70.

Naddef, D.J., and W.R. Pulleyblank (1982). Ear decompositions of elementary graphs and GF_2-rank of perfect matchings. *Ann. Discrete Math.* 16, 241–260.

Nemhauser, G.L., and L.A. Wolsey (1988). *Integer and Combinatorial Optimization*, John Wiley and Sons, New York, NY.

von Neumann, J. (1947). *Discussion of a maximum problem*, unpublished working paper, Institute for Advanced Studies, Princeton, NJ [Reprinted in: A.H. Taub (ed.), *John von Neumann, Collected works, Vol. VI,* Pergamon Press, Oxford, 1963, pp. 89–95].

von Neumann, J. (1953). A certain zero-sum two-person game equivalent to the optimal assignment problem, in: H.W. Kuhn and A.W. Tucker (eds.), *Contributions to the Theory of Games II*, Annals of Mathematical Studies, Vol. 28, Princeton University Press, Princeton, NJ, pp. 5–12.

Norman, R.Z., and M.O. Rabin (1959). An algorithm for a minimum cover of a graph. *Proc. Am. Math. Soc.* 10, 315–319.

Ore, O. (1955). Graphs and matching theorems. *Duke Math. J.* 22, 625–639.

Oxley, J.G. (1992). *Matroid Theory*, Oxford University Press, New York, NY.

Padberg, M.W. and M.R. Rao (1982). Odd minimum cut-sets and b-matchings. *Math. Oper. Res.* 7, 67–80.

Papadimitriou, C.H. (1977). The probabilistic analysis of matching heuristics, in: *Proc. 15th Annual Allerton Conf. on Communication, Control, and Computing*, pp. 368–378.

Peled, U.N., and M.N. Srinivasan (1989). The polytope of degree sequences, *Linear Algebra Appl.* 114/115, 349–373.

Petersen, J. (1891). Die Theorie der regulären graphs. *Acta Math.* 15, 193–220.

Pippinger, N. (1979). On simultaneous resource bounds, in: *Proc. 20th Annual Symp. on Foundations of Computer Science*, IEEE, New York, NY, pp. 307–311.

Plaisted, D.A. (1984). Heuristic matching for graphs satisfying the triangle inequality. *J. Algorithms* 5, 163–179.

Plummer, M.D. (1992). Matching theory – a sampler: from Dénes König to the present. *Discrete Math.* 100, 177–219.

Plummer, M.D. (1993). Matching and vertex packing: how "hard" are they? in: J. Gimbel, J.W. Kennedy and L.V. Quintas (eds.), *Quo Vadis, Graph Theory? A Source Book for Challenges and Directions, Ann. Discrete Math.* 55, 275–312.

Prim, R.C. (1957). Shortest connection networks and some generalizations. *Bell System Tech. J.* 36, 1389–1401.

Pulleyblank, W.R. (1973). *Faces of Matching Polyhedra*, PhD thesis, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario.

Pulleyblank, W. (1980). Dual integrality in b-matching problems. *Math. Program. Study* 12, 176–196.

Pulleyblank, W.R. (1981). Total dual integrality and b-matchings. *Oper. Res. Lett.* 1, 28–30.

Pulleyblank, W.R. (1983). Polyhedral combinatorics, in: A. Bachem, M. Grötschel and B. Korte (eds.), *Mathematical Programming, the State of the Art: Bonn 1982*, Springer-Verlag, Berlin, pp. 312–345.

Pulleyblank, W.R. (1989). Polyhedral combinatorics, in: G.L. Nemhauser, A.H.G. Rinnooy Kan and M.J. Todd (eds.), *Optimization*, Handbooks in Operations Research and Management Science, Vol. 1, North-Holland, Amsterdam, pp. 371–446.

Pulleyblank, W.R. (1995). Matchings and stable sets, in: R. Graham, M. Grötschel, and L. Lovász (eds.), *Handbook of Combinatorics*, to appear.

Pulleyblank, W., and J. Edmonds (1974). Facets of 1-matching polyhedra, in: C. Berge and D. Ray-Chaudury (eds.), *Hypergraph Seminar*, Springer-Verlag, Berlin, pp. 214–242.

Rabin, M.O., and V.V. Vazirani (1989). Maximum matchings in general graphs through randomization. *J. Algorithms* 10, 557–567.

Recski, A. (1989). *Matroid Theory and its Applications in Electrical Networks and Statics*, Springer-Verlag, Heidelberg.

Reichmeider, P.F. (1984). *The Equivalence of some Combinatorial Matching Problems*, Polygonal Publishing House, Washington DC.

Reingold, E.W., and K.J. Supowit (1983). Probabilistic analysis of devide-and-conquer heuristics for minimum weighted Euclidean matching. *Networks* 13, 49–66.

Reingold, E.M., and R.E. Tarjan (1981). On a greedy heuristic for complete matching. *SIAM J. Comput.* 10, 676–681.

Roth, A.E., U.G. Rothblum and J.H. Vande Vate (1993). Stable matchings, optimal assignments and linear programming. *Math. Oper. Res.* 18, 803–828.

Sbihi, N. (1980). Algorithme de recherche d'un stable de cardinalité maximum dans une graphe sans étoile. *Discrete Math.* 29, 53–76.

Schneider, H. (1977). The concepts of irreducibility and full indecomposability of a matrix in the works of Frobenius, König and Markov. *Linear Algebra Appl.* 18, 139–162.

Schrijver, A. (1983a). Min–max results in combinatorial optimization, in: A. Bachem, M. Grötschel and B. Korte (eds.), *Mathematical Programming, the State of the Art: Bonn 1982*, Springer-Verlag, Berlin, pp. 439–500.

Schrijver, A. (1983b). Short proofs on the matching polyhedron. *J. Comb. Theory, Ser. B* 34, 104–108.

Schrijver, A. (1986). *Theory of Linear and Integer Programming*, John Wiley and Sons, Chichester.

Schrijver, A. (1995). Polyhedral combinatorics, in: R.Graham, M. Grötschel, and L. Lovász (eds.), *Handbook of Combinatorics*, to appear.

Schrijver, A., and P.D. Seymour (1977). A proof of total dual integrality of matching polyhedra, Mathematical Centre Report ZN 79/77, Mathematisch Centrum, Amsterdam.

Schrijver, A., and P.D. Seymour (1994). Packing odd paths, *J. Comb. Theory, Ser. B* 62, 280–288.

Schwartz, J.T. (1980). Fast probabilistic algorithms for verification of polynomial identities. *J. Assoc. Comput. Mach.* 27, 701–717.

Sebö, A. (1986). Finding the t-join structure of graphs. *Math. Program.* 36, 123–134.

Sebö, A. (1987). A quick proof of Seymour's theorem on t-joins. *Discrete Math.* 64, 101–103.

Sebö, A. (1988). The Schrijver system of odd join polyhedra. *Combinatorica* 8, 103–116.

Sebö, A. (1990). Undirected distances and the postman-structure of graphs. *J. Comb. Theory, Ser. B* 49, 10–39.

Sebö, A. (1993). General antifactors of graphs. *J. Comb. Theory, Ser. B* 58, 174–184.

Seymour, P.D. (1977). The matroids with the max-flow min-cut property. *J. Comb. Theory, Ser. B* 23, 189–222.

Seymour, P.D. (1979). On multi-colourings of cubic graphs, and conjectures of Fulkerson and Tutte. *Proc. Lond. Math. Soc., Third Ser.* 38, 423–460.

Seymour, P.D. (1981). On odd cuts and plane multicommodity flows. *Proc. Lond. Math. Soc., Third Ser.* 42, 178–192.

Shapley, L.S., and M. Shubik (1972). The assignment game I: the core. *Int. J. Game Theory* 1, 111–130.

Sinclair, A., and M. Jerrum (1989). Approximate counting, uniform generation and rapidly mixing Markov chains. *Inf. Comput.* 82, 93–133.

Steele, J.M. (1981). Subadditive Euclidean functionals and nonlinear growth in geometric probability. *Ann. Probab.* 9, 365–376.

Sterboul, F. (1979). A characterization of the graphs in which the transversal number equals the matching number. *J. Comb. Theory, Ser. B* 27, 228–229.

Supowit, K.J., D.A. Plaisted and E.M. Reingold (1980). Heuristics for weighted perfect matching, in: *Proc. 12th Annual ACM Symp. on Theory of Computing*, Association for Computing Machinery,

New York, NY, pp. 398–419.
Supowit, K.J., and E.M. Reingold (1983). Devide and conquer heuristics for minimum weighted Euclidean matching. *SIAM J. Comput.* 12, 118–143.
Supowit, K.J., E.M. Reingold and D.A. Plaisted (1983). The traveling salesman problem and minimum matching in the unit square. *SIAM J. Comput.* 12, 144–156.
Szigeti, Z. (1993). *On Seymour Graphs*, Technical report, Department of Computer Science, Eötvös Loránd University, Budapest.
Tardos, É. (1985). A strongly polynomial minimum cost circulation algorithm. *Combinatorica* 5, 247–255.
Tarjan, R. E. (1983). *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, PA.
Truemper, K. (1992). *Matroid Decomposition*, Academic Press, San Diego.
Tutte, W.T. (1947). The factorization of linear graphs. *J. Lond. Math. Soc.* 22, 107–111.
Tutte, W.T. (1952). The factors of graphs. *Can. J. Math.* 4, 314–328.
Tutte, W.T. (1954). A short proof of the factor theorem for finite graphs. *Can. J. Math.* 6, 347–352.
Tutte, W.T. (1974). Spanning subgraphs with specified valancies. *Discrete Math.* 9, 97–108.
Tutte, W.T. (1981). Graph factors. *Combinatorica* 1, 79–97.
Vaidya, P.M. (1989). Geometry helps in matching. *SIAM J. Comput.* 18, 1201–1225.
Vaidya, P.M. (1990). Reducing the parallel complexity of certain linear programming problems, in: *Proc. 31th Annual Symp. on Foundations of Computer Science*, IEEE, New York, NY, pp. 583–589.
Valiant, L.G. (1979). The complexity of computing the permanent. *Theor. Comput. Sci.* 8, 189–201.
Vande Vate, J.H. (1989). Linear programming brings marital bliss. *Oper. Res. Lett.* 8, 147–153.
Vazirani, V.V. (1989). NC algorithms for computing the number of perfect matchings in $K_{3,3}$-free graphs and related problems. *Inf. Comput.* 80, 152–164.
Vazirani, V.V. (1994). A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{V}E)$ general graph maximum matching algorithm. *Combinatorica* 14, 71–109.
Vizing, V.G. (1964). On an estimate of the chromatic class of a p-graph (in Russian), *Diskretnyĭ Analiz* 3, 25–30
Vizing, V.G. (1965). The chromatic class of a multigraph (in Russian), *Kibernetika* 3, 29–39 [English translation: Cybernetics 1 (3) (1965) 32–41].
Warshall, S. (1962). A theorem on Boolean matrices. *J. Assoc. Comput. Mach.* 9, 11–12.
Weber, G.M. (1981). Sensitivity analysis of optimal matchings. *Networks* 11, 41–56.
Welsh, D.J.A. (1976). *Matroid Theory*, Academic Press, London.
Witzgall, C., and C.T. Zahn, Jr. (1965). Modification of Edmonds' maximum matching algorithm. *J. Res. Nat. Bur. Stand. – B. Math. Math. Phys.* 69B, 91–98.
Yakovleva, M.A. (1959). A problem on minimum transportation cost, in: V.S. Nemchinov (ed.), *Applications of Mathematics in Economic Research*, Izdat. Social'no-Ekon. Lit., Moscow, pp. 390–399.
Yannakakis, M. (1988). Expressing combinatorial optimization problems by linear programs, Working paper, AT&T Bell Laboratories [Extended abstract in: *Proc. 20th Annual ACM Symp. on Theory of Computing*, Association for Computing Machinery, New York, NY, pp. 223–228].