Edmonds, Matching and the Birth of Polyhedral Combinatorics

William R. Pulleyblank

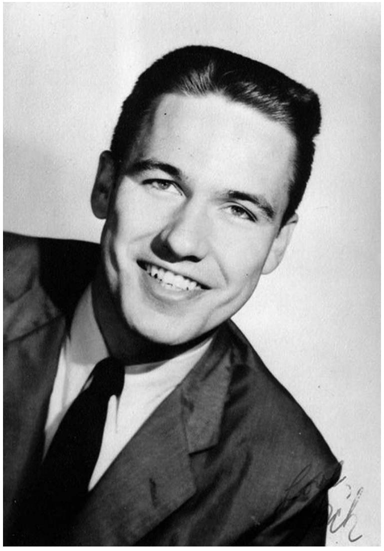
2010 Mathematics Subject Classification: 05C70, 05C85, 90C10, 90C27, 68R10, 68W40
Keywords and Phrases: Matchings, factors, polyhedral combinatorics, nonbipartite matching, integer programming

1 Summer of 1961, a Workshop at RAND

In the summer of 1961, Jack Edmonds, a twenty-seven year old mathematician, was attending a high powered workshop on combinatorics at the Rand Corporation in Santa Monica, California. His participation had been arranged by Alan Goldman, his manager at the National Bureau of Standards (now NIST), supported by Edmonds' Princeton mentor, A.W. Tucker. It seemed to Edmonds that every senior academician doing combinatorics was there. This included such luminaries as George Dantzig, Alan Hoffman, Ray Fulkerson, Claude Berge and Bill Tutte. The only "kids" participating were Michel Balinski, Larry Brown, Chris Witzgall, and Edmonds, who shared an office during the workshop.

Edmonds was scheduled to give a talk on his research ideas. At that time, he was working on some big questions. He had become intrigued by the possibility of defining a class of algorithms which could be proven to run more efficiently than exhaustive enumeration, and by showing that such algorithms existed. This was a novel idea. At this time, people were generally satisfied with algorithms whose running times could be proved to be finite, such as Dantzig's Simplex Algorithm for linear programming. In 1958, Ralph Gomory [14], [15] had developed an analogue of the Simplex Algorithm that he showed solved integer programs in finite time, similar to the Simplex Algorithm. Many people in the Operations Research community viewed a problem as "solved" if it could be formulated as an integer programming problem. However, unlike the Simplex Algorithm, Gomory's integer programming algorithm seemed to take so long on some problems that it was often unusable in practice.

At this time, the combinatorics community was not very interested in algorithms. Generally, graphs considered were finite and so most problems had



Jack Edmonds 1957 (courtesy Jeff Edmonds)

trivial finite solution methods. In 1963, Herb Ryser [23] published his monograph which noted that there were two general types of problems appearing in the combinatorics literature: existence problems (establish conditions characterising whether a desired structure exists) and enumeration problems (if a structure exists, determine how many of them there are). (A decade later, in 1972, Ryser, speaking at a conference on graph theory, added a third type of problem: develop an efficient algorithm to determine whether a desired object exists.)

Earlier, in 1954, Dantzig, Fulkerson and Selmer Johnson [4] had published what proved to be a ground breaking paper. They showed that a traveling salesman problem, looking for a shortest tour visiting the District of Columbia plus a selected major city in each of the (then) 48 states, could be solved to provable optimality by combining the ideas of linear and integer programming. They did not make any claims as to the efficiency of their solution method. What they did show was that it was possible to present an optimal solution to an instance of a combinatorial optimization problem, and a proof of optimality, that required much less time to check than it would have taken to try all possible solutions.

Through the 1950s, the world was seeing rapid development in the power and availability of digital computers. This provided another impetus to algorithmic development. Many combinatorial optimization problems were recognized as having practical applications. However even with the speed of the "high performance" computers of the day, it was recognized that improved algorithms were needed if problems of realistic size were to be solved in practice.

What Edmonds wanted was a specific concrete open example for which he could produce a better than finite algorithm and thereby illustrate the power and importance of his ideas.

The *perfect matching problem* in a graph $G = (V, E)$ is to determine whether there exists a set of edges meeting each node exactly once. If the graph is *bipartite* – its nodes can be partitioned into $V_1 \cup V_2$ and every edge joins a node in V_1 to a node of V_2 – then a rich theory had already been developed which not only characterized those bipartite graphs which had perfect matchings (Hall, [17]), but showed that this problem could be formulated as a small linear program. However, the more general case of *nonbipartite* graphs, graphs that contain odd cardinality cycles, seemed different. A necessary condition was that the number of nodes had to be even, but that was far from sufficient. Tutte [25] in 1947 had proved a generalization of Hall's theorem to nonbipartite graphs. However, it did not seem to lead to an algorithm more efficient than simply trying all possible subsets of the edges in hope that one would be a perfect matching.

A *matching* M in a graph G is a set of edges which meets each node at most once. M is *perfect* if it meets every node. Let U be the set of nodes not met by edges in M. An *augmenting path* with respect to M in G is a simple path joining two nodes of U whose edges are alternately not in M and in M. If an augmenting path exists, then a matching can be made larger – just remove the edges of the path that are in M and add to M the edges of the path not in M. In 1957 Claude Berge [1] showed that this characterized maximum matchings.

Theorem 1 (Berge's augmenting path theorem). *A matching M in a graph G is of maximum size if and only if there exists no augmenting path.*

This result was not only simple to prove, but also applied both to bipartite and nonbipartite graphs. However, whereas there were efficient methods for finding such augmenting paths, if they existed, in bipartite graphs, no such algorithms were known for nonbipartite graphs.

The night before his scheduled talk, Edmonds had an inspiration with profound consequences. A graph is nonbipartite if and only if it has an odd cycle. It seemed that it was the presence of these odd cycles that confounded the search for augmenting paths. But if an odd cycle was found in the course of searching for an augmenting path in a nonbipartite graph, the cycle could be *shrunk* to form a *pseudonode*. Thereby the problem caused by that odd cycle could be eliminated, at least temporarily. This simple and elegant idea was the key to developing an efficient algorithm for determining whether a nonbipartite graph had a perfect matching. Equally important, it gave Edmonds a concrete specific example of a problem that could illustrate the richness and the power of the general foundations of complexity that he was developing. This became the focal point of his talk the next day which launched some of the most significant research into algorithms and complexity over the next two decades.

Alan Hoffman recounted an exchange during the discussion period following Edmonds' lecture. Tutte's published proof of his characterization of nonbipartite graphs having perfect matchings was an ingenious application of matrix theory. Responding to a question, Edmonds ended a sentence by saying "using methods known only to Tutte and God". Tutte rarely made comments at the end of another person's lecture. There was a pause, at which point it was appropriate for Tutte to say something, but he said nothing. Hoffman intervened, asking "Would either of those authors care to comment?" Tutte did respond.

2 Context I: Bipartite graphs and the Hungarian method

The problem of determining whether a bipartite graph had a perfect matching had already been encountered in many different guises, and there were several equivalent characterizations of bipartite graphs having perfect matchings. See Schriver [24].

A *node cover* is a set C of nodes such that each edge is incident with at least one member of C. Each edge in any matching M will have to be incident with at least one member of C, and no member of C can be incident with more than one member of M. Therefore, the size of a largest matching provides a lower bound on the size of a smallest node cover. In 1931, Dénes Kőnig [18] had published a min-max theorem showing that these values are equal.

Theorem 2 (Kőnig's Bipartite Matching Theorem). *The maximum size of a matching in a bipartite graph $G = (V, E)$ equals the minimum size of a node cover.*

In 1935, in the context of transversals of families of sets, Phillip Hall [17] proved the following:

Theorem 3 (Hall's Bipartite matching Theorem). *A bipartite graph $G = (V, E)$ has a perfect matching if and only if, for every $X \subseteq V$, the number of isolated nodes in $G - X$ is at most $|X|$.*

These two theorems are equivalent, in that each can be easily deduced from the other. (Deducing Hall's Theorem from Kőnig's Theorem is easier than going the other direction.)

If a bipartite graph G has no perfect matching, then either of these provides a guaranteed simple way of showing that this is the case. We can exhibit a node cover of size less than $|V|/2$ or exhibit a set $X \subseteq V$ such that $G - X$ has at least $|X| + 1$ isolated nodes. (For now, do not worry about the time that it takes to find the cover or the set X.)

Note how these contrast with Berge's augmenting path theorem. Berge's theorem does suggest an approach for constructing a perfect matching if one exists, but if we wanted to use it to show that G had no perfect matching, we would have to start with a less-than-perfect matching in G and somehow prove that no augmenting path existed. How could this be done?

In 1931, Jenő Egerváry [12] published an alternate proof and a weighted generalization of Kőnig's theorem. (See [24].) Suppose that we have a bipartite graph $G = (V, E)$ and a real edge weight c_j for each $j \in E$. The *weight* of a matching is the sum of the weights of its edges. He proved a min-max theorem characterizing the maximum possible weight of a matching in G by showing that it was equal to the minimum weight of a weighted node cover of the edges of G.

Theorem 4 (Egerváry's Theorem). *Let $G = (V, E)$ be a bipartite graph and let $(c_j : j \in E)$ be a vector of edge weights. The maximum weight of a matching in G equals the minimum of $\sum_{v \in V} y_v$, where $y = (y_v : v \in V)$ satisfies $y_u + y_v \geq c_j$ for every $j = \{u, v\} \in E$.*

This implied that the existence of a perfect matching in a bipartite graph $G = (V, E)$ could be determined by solving a linear system. For each edge $j \in E$, define a variable x_j. Then $x = (x_j : j \in E)$ is a real vector indexed by the edges of G.

Consider the following system of linear equations and (trivial) inequalities:

$$\sum (x_j : j \in E \text{ incident with } v) = 1 \text{ for each node } v \in V, \tag{1}$$

$$x_j \geq 0 \text{ for each } j \in E. \tag{2}$$

If G has a perfect matching M, we can define $\hat{x}_j = 1$ for $j \in M$ and $\hat{x}_j = 0$ for $j \in E \backslash M$. Then $\hat{x}$ is a feasible solution to this linear system. Conversely, if we have an integer solution to this linear system, all variables will have value 0 or 1 and the edges with value 1 will correspond to the edges belonging to a perfect matching of G.

Theorem 5. *A bipartite graph $G = (V, E)$ has a perfect matching if and only if the linear system* (1), (2) *has an integer valued solution.*

However, in general there also exist fractional solutions to this system. Could there exist fractional solutions to this linear system but no integer valued solutions? In this case, the solution to the linear system might not tell us whether the graph had a perfect matching. Egerváry's Theorem showed that this was not the case.

Egerváry's Theorem is not true in general for nonbipartite graphs. It already fails for K_3. In this case, the linear system has a solution obtained by setting $x_j = 1/2$ for all three edges, but there is no integer valued solution. (The conditions of Hall's and Kőnig's Theorems also fail to be satisfied for K_3.)

Egerváry's Theorem showed that the maximum weight matching problem for bipartite graphs could be solved by solving the *linear* program of maximizing $\sum (x_j \cdot c_j : j \in E)$ subject to (1), (2). The dual linear program is to minimize $\sum_{v \in V} y_v$, where $y = (y_v : v \in V)$ satisfies $y_u + y_v \geq c_j$ for every $j = \{u, v\} \in E$. His proof showed how to find an integer x and (possibly) fractional y which were optimal primal and dual solutions.

In 1955, Harold Kuhn [19] turned Egerváry's proof of his theorem into an algorithm which would find a maximum weight matching in a bipartite graph.

The algorithm was guaranteed to stop in finite time. In 1957, James Munkres [20] showed that this algorithm, called "The Hungarian Method", would terminate in time $O(n^4)$ for a simple bipartite graph with n vertices.

3 Context II: Tutte's theorem and the Tutte–Berge formula

In 1947, William Tutte [25] had generalized Hall's theorem to nonbipartite graphs. He proved that replacing "isolated nodes" by "odd cardinality components" yielded a characterization of which nonbipartite graphs have perfect matchings.

Theorem 6 (Tutte's matching Theorem). *A (nonbipartite or bipartite) graph $G = (V, E)$ has a perfect matching if and only if, for every $X \subseteq V$, the number of odd cardinality components of $G - X$ is at most $|X|$.*

As in the case of Hall's Theorem, the necessity of the condition is straightforward. If there exists a perfect matching M, then an edge of M must join some node of each odd component of $G - X$ to a node of X, since it is impossible to pair off all the nodes of an odd component K using only edges with both ends in K. The important part of the theorem is the sufficiency, which asserts that if G does not have a perfect matching, then there exists an X whose removal creates more than $|X|$ odd cardinality components.

Hall's Theorem does strengthen Tutte's theorem in the bipartite case as follows. It shows that, in this case, we can restrict our attention to components of $G - X$ which consist of single nodes, rather than having to consider all possible components. But Tutte's theorem works for all graphs. For example, whereas Hall's condition is not violated for K_3, Tutte's Theorem shows that no perfect matching exists, by taking $X = \emptyset$.

In 1958, Berge [2] noted that Tutte's theorem implied a min-max theorem for $\nu(G)$, the size of a largest matching in a graph $G = (V, E)$. For any $X \subseteq V$, we let odd(X) be the number of odd cardinality components of $G - X$.

Theorem 7 (Tutte–Berge Formula). *For any graph $G = V, E$,*

$$\nu(G) = \frac{1}{2}(|V| - \min(\mathrm{odd}(X) - |X| : X \subseteq V)).$$

The formula shows that the smallest number of nodes which must be left unmet by any matching equals the largest possible difference between odd(X) and $|X|$.

Here then were the challenges: Could the notion of "efficient" be made precise mathematically? Was it possible to develop an efficient algorithm for determining whether an arbitrary graph had a perfect matching? Given an arbitrary graph $G = (V, E)$, could you either find a perfect matching or find a set $X \subseteq V$ for which $|X| < \mathrm{odd}(X)$?

4 Paths, Trees and Flowers; $\mathcal{P}$ and $\mathcal{NP}$

Edmonds' landmark paper [5], *Paths, Trees and Flowers*, evolved from the talk that he presented at Rand in 1961. His algorithm for determining whether a nonbipartite graph $G = (V, E)$ has a perfect matching can be summarized as follows.

Start with any matching M. If M is perfect, then the algorithm is done. If not, some node r is not met by any edge of M. In this case, grow an alternating search tree T rooted at r which will either find an augmenting path, enabling the matching to be made larger, or find a set $X \subseteq V$ for which $|X| < \text{odd(X)}$.

The search tree initially consists of just the root node r. Each node v of T is classified as *even* or *odd* based on the parity of the length of the (unique) path in T from r to v. The algorithm looks for an edge j of G that joins an even node u of T to a node w which is not already an odd node of T. If such a j exists, there are three possibilities.

1. *Grow Tree:* If w is met by an edge k of M, then T is grown by adding j, k and their end nodes to T.

2. *Augment M:* If w is not met by an edge of M, then we have found an augmenting path from r to w. We augment M using this path, as proposed by Berge, and select a new r if the matching is not perfect.

3. *Shrink:* If w is an even node of T, then adding j to T creates a unique odd cycle C. Shrink C by combining its nodes to form a *pseudonode*. The pseudonode C will be an even node of the tree created by identifying the nodes of G belonging to C.

If no such j exists, then let X be the set of odd nodes of T. Each even node w of T will correspond to an odd cardinality component of $G - X$. If w is a node of G, then the component consists of the singleton w. If w was formed by shrinking, then the set of all nodes of G shrunk to form w will induce an odd component of G.

If G is bipartite, then the Shrink step will not occur and the algorithm reduces to a previously known matching algorithm for bipartite graphs.

One point we skipped over is what happens to an augmenting path when it passes through a pseudo-node. It can be shown that by choosing an appropriate path through the odd cycle, an augmenting path in a graph obtained by shrinking can be extended to an augmenting path in the original graph. See Edmonds [5] or Cook et al [3] for details.

Edmonds [5] presents his algorithm for the closely related problem of finding a maximum cardinality matching in an arbitrary graph. If the above algorithm terminates without finding a perfect matching, then he calls the search tree T *Hungarian.* He lets G' be the graph obtained from G by deleting all vertices in T or contained in pseudonodes of T. He shows that a maximum matching of G', combined with a maximum matching of the subgraph of G induced by the nodes belonging to T or contained in pseudonodes of T, forms a maximum matching of G.

The second section of Edmonds [5] is entitled "Digression". This section began by arguing that finiteness for an algorithm was not enough. He defined a *good algorithm* as one whose worse case runtime is bounded by a polynomial function of the size of the input. This criteria is robust, it is independent of the actual computing platform on which the algorithm was run. Also, it has the attractive feature that good algorithms can use other good algorithms as subroutines and still be good. He stressed that this idea could be made mathematically rigorous.

The maximum matching algorithm, which Edmonds (conservatively) showed had run time $O(|V|^4)$, provided an initial case study. This was the first known algorithm for maximum matching in nonbipartite graphs with a running time asymptotically better than trying all possible subsets. The bound on the running time was about the same as the bound on solving the matching problem for a bipartite graph.

One concern raised about Edmonds' notion of a good algorithm was that a good algorithm with a high degree polynomial bound on its run times could still take too long to be practical. Edmonds stressed that his goal was to develop a mathematically precise measure of running times for algorithms that would capture the idea of "better than finite". A second concern arose from the simplex algorithm for linear programming. This algorithm was proving itself to be very effective for solving large (at the time) linear programs, but no polynomial bound could be proved on its running time. (It would be almost two decades later that a good algorithm would be developed for linear programming.) So the concept of "good algorithm" was neither necessary nor sufficient to characterize "efficient in practice". But there was a high degree of correlation, and this concept had the desired precision and concreteness to form a foundation for a study of worst case performance of algorithms.

Part of the reason for the lasting significance of [5] is that the paper promoted an elegant idea – the concept of a *good* (polynomially bounded) algorithm. It also gave the first known such algorithm for the matching problem in nonbipartite graphs, a fundamental problem in graph theory. Edmonds also raised the question of whether the existence of theorems like Tutte's Theorem or Hall's Theorem – min-max theorems or theorems characterizing the existence of an object (a perfect matching in a bipartite graph) by prohibiting the existence of an obstacle (a set $X \subset V$ for which $G - X$ has at least $|X| + 1$ isolated nodes) – could enable the construction of efficient algorithms for finding the objects if they existed. He had shown how this worked in the case of matchings in bipartite graphs and his algorithm had extended this to nonbipartite graphs. He called these sorts of theorems *good characterizations*.

Some people argued that nobody could possibly check all subsets X and see how many isolated nodes existed in $G - X$. There were simply too many of them; the number grew exponentially with the size of G. What did this have to do with answering the original question?

But here was the point. Consider the question: does G have a perfect matching? If the answer is "Yes", we can prove this by exhibiting a perfect matching M. If the answer is "No", then we can prove this by exhibiting a single $X \subseteq V$ for which $G - X$ has at least $|X| + 1$ isolated nodes. This has not yet described an effective method for finding M or X, but at least it provided a polynomially bounded proof for either alternatives. It gave a stopping criterion for an algorithm.

A decade later, these concepts were essential ideas embodied in the classes $\mathcal{P}$ and $\mathcal{NP}$. The question Edmonds asked relating the existence of good characterizations to the existence of good algorithms became what is now recognized as the most important open question in theoretical computer science: Is $\mathcal{P} = \mathcal{NP}$?

5 Weighty matters

Edmonds quickly generalized his nonbipartite matching algorithm to the corresponding edge weighted problem (Edmonds [6]). (Recall, each edge j is given a cost c_j and the algorithm constructs a matching M for which $\sum(c_j : j \in M)$ is maximum.) He did this by an elegant extension of Egerváry's approach that had worked for bipartite graphs. He showed how to use the primal-dual method for linear programming and the operation of shrinking to extend the cardinality case to the weighted case.

Edmonds began by formulating the maximum weight matching problem as a linear programming problem:

$$\text{Maximize } \sum(c_j x_j : j \in E)$$

subject to

$$\sum(x_j : j \in E \text{ incident with } v) \leq 1 \text{ for each node } v \in V, \tag{3}$$

$$\sum_{j \in E}(x_j : j \text{ has both ends in } S) \leq (|S| - 1)/2 \text{ for each } S \subseteq V \text{ such that } |S| \geq 3 \text{ is odd}, \tag{4}$$

$$x_j \geq 0 \text{ for each } j \in E. \tag{5}$$

This was really an audacious idea. The number of inequalities (4) grows exponentially with the number of nodes of G. No available linear programming code could read and store the set of constraints for a moderate sized weighted matching problem, let alone solve the problem. However Edmonds' idea was this: the real value of linear programming for a problem like weighted matching is not the simplex algorithm. It is that linear duality theory provides a method of giving a short proof of optimality.

His algorithm constructed a vector $x = (x_j : j \in E)$ which was the (0-1)-incidence vector of a matching in G. It also constructed a feasible solution to the dual linear program to maximizing $c \cdot x$ subject to (3), (4) and (5). Moreover, x and the dual solution would satisfy the complementary slackness conditions of linear programming which established their optimality.

The algorithm had essentially the same bound on its run time as the maximum cardinality algorithm. There was a minor complication. The bound had to take into account the complexity of arithmetic operations on the costs c_j. These operations were addition, subtraction, comparison and division by 2. This required either the introduction in the bound of a factor $\sum_{j \in E} \log(c_j)$ or else a “fixed word” assumption that all costs were within some bounded range.

6 Generality and extensions

Soon after this, Ellis L. Johnson, a recent Berkeley PhD student of Dantzig, began to work with Edmonds. They wanted to see how much they could generalize this theory of matchings in general graphs, in the context of linear and integer programming. They extended the algorithm to accommodate the following extensions (see [8]):

6.1 General degree constraints

Generalize the constraints (3) to

$$\sum (x_j : j \in E \text{ incident with } v) \leq b_v \text{ for each node } v \in V, \tag{6}$$

where, for each $v \in V$, b_v is a nonnegative integer. This extends the graph theoretic idea of a matching to a vector $x = (x_j : j \in E)$ of nonnegative integers such that, for each $v \in V$, the sum of the x_j on the edges j is at most b_v. Such a vector x is called a *b-matching*. If $b_v = 1$ for all $v \in V$, then a b-matching is the incidence vector of a matching. Let $b(V)$ denote $\sum_{v \in V} b_v$.

Tutte [26] had already shown that this problem could be transformed into a matching problem in which $b_v = 1$ for all $v \in V$ by replacing each vertex for which $b_v > 1$ by $|b_v|$ new vertices, and each edge $j = \{u, v\}$ with a complete bipartite graph joining the sets of new vertices corresponding to u and v. For a b matching x, the *deficiency* $d(x, v)$ of x at vertex v is defined as $b_v - \sum(x_j : j \in E, j \text{ incident with } v)$. The *deficiency* $D(x)$ of x is defined as $\sum_{v \in V} d(x, v)$.

The Tutte–Berge Formula generalizes to b-matchings as follows: For each $X \subseteq V$, let $K^0(X)$ be the nodes belonging to one node components of $G - X$; let $\text{odd}(X)$ be the number of components K of $G - X$ having at least three nodes for which $\sum_{i \in V(K)} b_i$ is odd.

Theorem 8 (Tutte–Berge Formula for b-matchings). *For any graph $G = V, E$ and any vector $b = (b_v : v \in V)$ of nonnegative integers,*

$$\begin{aligned} &\min\ (D(x) : x \textit{ is a } b - \textit{matching of } G) \\ &\qquad = \max\Big(\sum_{v \in K^0(X)} b_v + \text{odd}(X) - \sum_{v \in X} b_v : X \subseteq V\Big). \end{aligned}$$

Edmonds’ matching algorithm, described in Section 4, generalized to a direct algorithm for finding a maximum weight b-matching. It used a similar primal/dual framework to reduce the weighted problem to a cardinality problem. It started with an arbitrary b-matching $\bar{x}$ and defined a node v to be *unsaturated* if $\sum(\bar{x}_j : j \in E$ incident with $v) < b_v$. Now an augmenting path became a path in G joining two unsaturated nodes such that for each even edge j in the path, $\bar{x}_j > 0$. This would enable an augmentation to be made by increasing $\bar{x}_j$ for the odd edges in the path and decreasing $\bar{x}_j$ for the even edges. Similar to before, the algorithm grew an alternating search tree T rooted at an unsaturated node r. If it found an unsaturated even node of T other than r, it augmented the b-matching. If an edge j was found joining two even nodes of T, then it had found an odd cycle which it shrunk. But in this case any nodes of the tree joined to the odd cycle by paths in the tree for which every edge j had $\bar{x}_j > 0$ were also shrunk with the odd cycle. Set $bv = 1$ for the resulting pseudonode v.

Let $\bar{x}$ be the initial b-matching. This algorithm had worst case running time of $\mathrm{O}(D(\bar{x}) \cdot |V|^2)$. The bound came from the fact that each augmentation reduced the sum of the deficiencies by at least 2, and the time taken to find an augmentation, if one existed, was $\mathrm{O}(|V|^2)$. If we started with $\bar{x} = 0$, then the bound was $\mathrm{O}(b(V) \cdot |V|^2)$.

This created a potential problem. The length of a binary encoding of the input was polynomial in $|V|$ and $\sum_{v \in V} \log b_v$. However, $b(V)$ grows exponentially with $\sum_{v \in V} \log b_v$ and so the bound on the run time was growing exponentially with the size of a "natural" encoding of the input. How could it be made into a good algorithm?

Creating a good algorithm for finding a maximum (or minimum) weight perfect b-matching required three ideas. First, for each $v \in V$, let $\hat{b}_v$ be the largest *even* integer no greater than b_v. The resulting $\hat{b}$-matching problem can be transformed into a network flow problem in a bipartite directed graph G' having $2|V|$ nodes. For each node $v \in V$, create two nodes v' and v'' in G' and for each edge $\{u, v\}$ in G, create two directed arcs (u', v'') and (v', u'') in G'. Let $b'_v = b_v/2$ and let $b''_v = -b_v/2$. Edmonds and Richard Karp [11] created a good algorithm for finding a maximum flow in G' having maximum cost. By adding together the flows in the arcs (u', v'') and (v', u'') for each edge $\{u, v\}$ of G, we get a $\hat{b}$-matching $\bar{x}$ of G having minimum deficiency with respect to $\hat{b}$.

Second, use $\bar{x}$ as a starting matching to find a maximum weight b-matching in G.

The third idea was to show that the deficiency of $\bar{x}$ cannot be too large. let R be the set of nodes v for which b_v is odd. By the Tutte-Berge formula for b-matchings, if the deficiency of $\bar{x}$ is greater than $|R|$, then G does not have a perfect b-matching. Otherwise, the weighted b-matching algorithm performs at most $|R|$ augmentations, so the bound on the running time becomes $\mathrm{O}(|R| \cdot |V|^2)$ and we have a good algorithm.

See Gerards [13].

6.2 Edge capacities

For each edge $j \in E$, let u_j be an integral upper bound and let l_j be an integral lower bound on the value of x_j for the edge j. That is, the inequalities (5) are replaced with

$$l_j \leq x_j \leq u_j \text{ for each } j \in E. \tag{7}$$

The constraints (3) and (5) of the original weighted matching problem forced every edge j to have a value 0 or 1. However we now permit x_j to be any integer in the range $[l_j, u_j]$. If we add this to the b-matching problem, we obtain the *capacitated b-matching problem*.

In the special case that $l_j = 0$ and $u_j = 1$ for all $j \in E$, we obtain a *factor* problem. Now we want to find a maximum weight subset of the edges that meet each vertex v at most b_v times. We have now gone to a significantly more general set of linear constraints on our problem.

The case $b_v = 2$ for all $v \in V$ and $c_j = 1$ for all $j \in E$ is particularly interesting. This is the *maximum 2-factor problem* – find a set of vertex disjoint cycles in a graph that contain the maximum possible number of vertices.

6.3 Bidirected graphs

Edmonds and Johnson recognized that they could develop a unified model that included matching in general undirected graphs as well as network flow problems in directed graphs by introducing the idea of *bidirected* graphs. Each edge of the graph will have one or two *ends*. Each end will be either a *head* or a *tail*. Some edges will have a head and a tail. These are called *directed* edges. Some will have two heads or two tails. These are called *links*. An edge with one end is called a *slack* and that end can be either a head or a tail. The constraints (6) are now changed to the following:

$$\begin{aligned}&\sum(x_j : j \in E, j \text{ has a head incident with } v)\\ &\quad - \sum(x_j : j \in E, j \text{ has a tail incident with } v) = b_v \text{ for every node } v \in V.\end{aligned}$$

If all edges are links with both ends heads, then this becomes the capacitated b-matching problem. If all edges are directed, then this becomes a network flow problem. However, allowing a mixture of links, slacks and arcs provides a mixture of the two models, plus more. Note that by allowing slacks, all degree constraints can be turned into equations.

Combining these extensions, Edmonds and Johnson had developed a good algorithm for integer programming problems,

$$\text{maximize } cx$$

subject to

$$\begin{aligned}Ax &= b\\ l \leq x &\leq u\end{aligned}$$

where b, l, and u are integral, A is a matrix all of whose entries are $0, 1, -1, 2, -2$ and, for each column of A, the sum of the absolute values of the entries is at most 2.

6.4 Parity constraints

Edmonds and Johnson [9] also extended the idea of capacitated b-matching to allow so called parity constraints at the nodes. For each $v \in V, b_v = 0$ or 1. The constraints (6) became:

$$\sum (x_j : j \in E \text{ incident with } v) \equiv b_v \bmod 2 \text{ for each node } v \in V.$$

This enabled the so-called *Chinese Postman Problem* or T-join problem to be formulated as a capacitated b-matching problem. They provided both a direct algorithm and a reduction to this problem. See also Grötschel and Yuan [16].

At this time, Edmonds, Johnson and Scott Lockhart [10] developed a FORTRAN computer code for the weighted capacitated b-matching problem in bidirected graphs. This showed convincingly that this algorithm was a practical way to solve very large matching problems. It also provided a concrete instantiation of the algorithm which enabled precise calculation of an upper bound on its running time as a function of the input size.

Part of the motivation for doing this appeared in Section 2 of [5]. The described FORTRAN machine was an alternative to a Turing machine, a widely adopted model of computation for theoretical computing science. The FORTRAN machine was very close to the machine architectures of the day, and there existed a good algorithm for a FORTRAN machine if and only if there existed a good algorithm for a Turing machine. Also, the upper bound of the run time on a FORTRAN machine was much lower than for a Turing machine.

Edmonds and Johnson [8] also described reductions that enabled these extensions to be transformed to weighted matching problems in larger graphs.

7 Combinatorial polyhedra

In the early 1960s, it was recognized that a great many combinatorial optimization problems could be formulated as *integer* linear programs. It was also known that an integer linear program could be transformed into a linear program by adding a sufficient set of additional inequalities, called *cuts*, that trimmed the polyhedron of feasible solutions so that all vertices were integer valued, without removing any feasible integer solutions. Gomory's algorithm for integer programming gave a finite procedure for solving any integer program by successively adding cuts and re-solving until an optimum solution was found which was integer valued. His algorithm seemed to be a simple extension of the simplex algorithm for linear programming. However it had already been observed empirically that whereas the simplex algorithm was very successful for linear programs, Gomory's algorithm often failed to obtain a solution to an integer program in an acceptable amount of time. The only bound on the number of cuts that might be generated was exponential. This supported Edmonds' view that "finite was not good enough".

There were classes of integer programs for which no cuts needed to be added, for example, network flow problems and maximum weighted matching in bipartite graphs. Most of these classes of problems had total unimodularity at the core. A matrix $A = (a_{ij} : i \in I, j \in J)$ is *totally unimodular* if for any square submatrix M of A, $\det(M) = 0, 1, \text{ or } -1$. Note that this implies that all entries of A have value $0, 1, \text{ or } -1$. Suppose that A is totally unimodular and b is integral valued. It follows directly from Cramer's rule that, for any c, if the linear program maximize cx subject to $Ax = b, x \geq 0$ has an optimum solution, then it has one that is integer valued. It was well known that if G was a bipartite graph, then the matrix A defined by (1) is totally unimodular, so a maximum matching in a bipartite graph could be obtained by solving the linear program of maximizing cx subject to (3) and (2). If A was the node-arc incidence matrix of a directed graph, then the maximum flow problem could be formulated as a linear program with a totally unimodular matrix implying that if the node demands and arc capacities were integral, then there existed an integral optimal flow. See Cook et al [3].

It was well known that the weighted matching problem could be formulated as the *integer* linear programming problem of maximizing $\sum(c_j x_j : j \in E)$ subject to (3) and $x_j \geq 0, \textit{integer}$ for all $j \in E$. Edmonds had shown that the weighted matching algorithm correctly solved the problem by showing that it gave an integer valued optimum solution to the linear programming problem of maximizing $\sum(c_j x_j : j \in E)$ subject to (3), (4) and (5). That is, he had shown that the integrality constraint could be replaced by adding the cuts (4).

This was the first known example of a general combinatorial problem which could be formulated as a linear programming problem by adding an explicitly given set of cuts to a natural integer programming formulation. Dantzig et al [4] had shown that a particular instance of a traveling salesman problem could be solved starting from an integer programming formulation by adding a small set of cuts. What Edmonds had shown was that for *any* maximum weight matching problem, by adding the cuts (4), the integer program could be transformed to a linear program. He and Johnson had also shown for all the extensions in the previous section that the same paradigm worked. They gave explicit sets of cuts that, when added, transformed the problem to a linear programming problem.

This motivated further research on other problems amenable to this approach. It worked in many cases (for example, matroid optimization, matroid intersection, optimum branchings, triangle-free 2-matchings) but there are still many natural problems for which no explicit set of cuts is known.

The matching polyhedron $M(G)$ is the convex hull of the incidence vectors of the matchings of a graph $G = (V, E)$. Edmonds showed that $M(G) = \{x \in \Re^E : x \text{ satisfies } (3), (4) \text{ and } (5)\}$. This problem of finding a linear system sufficient to define a polyhedron defined by a combinatorial optimization problem – or equivalently, formulating the problem as a linear program – became a very active area of research through the 1970s, building on the successes obtained with matching problems.

The fundamental role of shrinking in solving nonbipartite matching problems had another interesting consequence. In general, not all constraints (4) are necessary to obtain a linear system sufficient to define $M(G)$. For example, if $|S|$ is odd, but $G[S]$, the subgraph of G induced by S, is not connected, then the constraint (4) corresponding to S is unnecessary. It is implied by these constraints for the nodesets of the odd cardinality connected components of $G[S]$. Edmonds and Pulleyblank [22] showed that the essential constraints (4) for $M(G)$ correspond to those sets $S \subseteq V$ for which $G[S]$ is 2-connected and is *shrinkable*. Shrinkable means that $G[S]$ will be reduced to a single pseudonode if the maximum matching algorithm is applied to it. Equivalently, a graph $G[S]$ is shrinkable if and only if $G[S]$ has no perfect matching, but for every node $v \in S$, the graph obtained from $G[S]$ by deleting v and all incident edges does have a perfect matching. The generalizations to b-matching appeared in Pulleyblank's PhD thesis [21], prepared under the supervision of Edmonds.

The problem of determining the essential inequalities to convert an integer program to a linear program is called *facet determination*. This became an active research area over the 1970s and 1980s – determining the facets of combinatorially defined polyhedra.

Acknowledgements. I am grateful to Kathie Cameron, Bill Cunningham, Alan Hoffman and, especially, Jack Edmonds for assistance with the primary source research for this chapter.

References

[1] C. Berge, Two theorems in graph theory, Proc. Nat. Academy of Sciences (U.S.A.) 43 (1957) 842–844.

[2] C. Berge, Sur le couplage maximum d'un graphe, Comptes Rendu de l'Académie des Sciences Paris, series 1, Mathématique 247 (1958), 258–259.

[3] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank and A. Schrijver, Combinatorial Optimization, Wiley-Interscience (1998).

[4] G. Dantzig, D.R, Fulkerson and S. Johnson, Solution of a large scale traveling salesman problem, Operations Reserach 2 (1954) 393–410.

[5] J. Edmonds, Paths, trees and flowers, Canadian J. of Math. 17 (1965) 449–467.

[6] J. Edmonds, Maximum matching and a polyhedron with 0,1 vertices, J. Res. Nat'l. Bureau of Standards 69B (1965) 125–130.

[7] J. Edmonds, A glimpse of heaven, in History of Mathematical Programming: A collection of Personal Reminiscences (J.K. Lenstra, A.H.G. Rinnoy Kan and A. Schrijver eds.), North-Holland (1991), pp. 32–54.

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

[9] J. Edmonds and E.L. Johnson, Matchings, Euler tours and the Chinese Postman, Mathematical Programming 5 (1973) 88–124.

[10] J. Edmonds, E.L. Johnson and S.C. Lockhart, Blossom I, a code for matching, unpublished report, IBM T.J. Watson Research Center, Yorktown Heights, New York (1969)

[11] J. Edmonds and R.M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems, J. of the ACM 19 (1972) 248–264.

[12] J. Egerváry, Matrixok kombinatorius tulajdonságairól, (in Humgarian)(On combinatorial properties of matrices), Matematikai és Fizikai Lapok 38 (1931) 16–28.

[13] A.M.H. Gerards, Matching, Chapter 3 in M.O. Ball et al eds., Handbooks in OR and MS Vol. 7 (1995) pp. 135–224.

[14] R.E. Gomory, Outline of an algorithm for integer solutions to linear programs, Bulletin of the American Mathematical Society 64 (1958), 275–278.

[15] R.E. Gomory, Solving linear programming problems in integers, in Combinatorial Analysis (R. Bellman and M. Hall Jr. eds.), American Mathematical Society (1960), pp. 211–215.

[16] M. Grötschel and Ya-Xiang Yuan, Euler, Mei-Ko Kwan, Königsberg, and a Chinese Postman, this volume, Chapter 7 (2012).

[17] P. Hall, On representatives of subsets, J. London Math. Soc. 10 (1935), 26–30.

[18] D. Kőnig, Graphok és matrixok, Matematikai és Fizikai Lapok 38 (1931) 116–119.

[19] H.W. Kuhn, The Hungarian method for the assignment problem, Naval Research Logistics Quarterly 2 (1955) 83–97.

[20] J. Munkres, Algorithms for the assignment and transportation problems, J. of Soc. for Industrial and Applied Mathematics 5 (1957) 32–38.

[21] W.R. Pulleyblank, Faces of Matching Polyhedra, PhD Thesis, University of Waterloo (1973).

[22] W.R. Pulleyblank and J. Edmonds, Facets of 1-matching polyhedra, in Hypergraph Seminar (C. Berge and D. Ray-Chaudhuri, eds.) Springer, Berlin (1974) pp. 214–242.

[23] H.J. Ryser, Combinatorial Mathematics, Math. Assoc. of America, John Wiley and Sons, Inc. (1963).

[24] A. Schrijver, Combinatorial Optimization, Springer Verlag (2003).

[25] W.T. Tutte, The factorization of linear graphs, J. London Math. Soc. 22 (1947) 107–111.

[26] W.T. Tutte, A short proof of the factor theorem for finite graphs, Canadian J. of Math. 6 (1954) 347–352.

William R. Pulleyblank
Department of
Mathematical Sciences
United States Military
Academy, West Point
West Point, NY 10996, USA
William.Pulleyblank@usma.edu