A faster exact separation algorithm for blossom inequalities

Adam N. Letchford[1], Gerhard Reinelt[2], and Dirk Oliver Theis[3]

[1] Department of Management Science, Lancaster University, Lancaster LA1 4YW, England A.N.Letchford@lancaster.ac.uk

[2] Institut für Informatik, University of Heidelberg, Germany Gerhard.Reinelt@informatik.uni-heidelberg.de

[3] Institut für Informatik, University of Heidelberg, Germany Dirk.Theis@informatik.uni-heidelberg.de

Abstract. In 1982, Padberg and Rao gave a polynomial-time separation algorithm for b-matching polyhedra. The current best known implementations of their separation algorithm run in $\mathcal{O}(|V|^2|E|\log(|V|^2/|E|))$ time when there are no edge capacities, but in $\mathcal{O}(|V||E|^2\log(|V|^2/|E|))$ time when capacities are present.
We propose a new exact separation algorithm for the capacitated case which has the same running time as for the uncapacitated case. For the sake of brevity, however, we will restrict our introduction to the case of perfect 1-capacitated b-matchings, which includes, for example, the separation problem for perfect 2-matchings. As well as being faster than the Padberg-Rao approach, our new algorithm is simpler and easier to implement.
Key Words: matching, polyhedra, separation.

1 Introduction

Let $G = (V, E)$ be an undirected graph, and let $b \in \mathbb{Z}_+^{|V|}$ be a vector of vertex capacities with $\sum_{v\in V} b_v$ even. A *(perfect 1-capacitated) b-matching* is a set of edges, such that for each $i \in V$, there are exactly b_i edges in the set incident to i. If we define for each edge e the integer variable x_e, which is one if e appears in the matching and zero otherwise, then the incidence vectors of b-matchings are the solutions of:

$$x(\delta(i)) = b_i, \text{ for all } i \in V \tag{1}$$

$$0 \le x_e \le 1, \text{ for all } e \in E \tag{2}$$

$$x_e \in \{0, 1\}, \text{ for all } e \in E. \tag{3}$$

Here, as usual, $\delta(i)$ represents the set of vertices incident to i, and $x(F)$ denotes $\sum_{e\in F} x_e$.

The convex hull in $\mathbb{R}^{|E|}$ of solutions of (1)–(3) is called the *b-matching polytope*. Edmonds and Pulleyblank (see Edmonds [2] and Pulleyblank [13]) gave a complete linear description of this polytope. It is described by the *degree equalities* (1), the *bounds* (2) and the following so-called *blossom inequalities*:

$$x(\delta(W) \setminus F) - x(F) \geq 1 - |F|, \qquad \text{for all } W \subset V, F \subseteq \delta(W) \text{ with } b(W) + |F| \text{ odd.} \tag{4}$$

Here $\delta(W)$ represents the set of edges with exactly one end-vertex in W and $b(W)$ denotes $\sum_{i \in W} b_i$. The set W is sometimes called the *handle* and the edges in the set F are called *teeth* of the blossom inequalities.

In their seminal paper, Padberg and Rao [10] devised a combinatorial polynomial-time *separation algorithm* for b-matching polytopes. A separation algorithm is a procedure which, given a vector $x^* \in \mathrm{R}^{|E|}$ lying outside of the polytope, finds a linear inequality which is valid for the polytope yet violated by x^* (see for example [6]). Separation algorithms are at the heart of the well known *branch-and-cut* approach (see [11]) to combinatorial optimization.

Clearly, testing if a degree equation or bound is violated can be performed in $\mathcal{O}(|E|)$ time, so the main contribution of [10] is to identify violated blossom inequalities.

The Padberg-Rao separation algorithm involves up to $|V|+|E|-1$ maximum flow computations on a special graph, the so-called *split graph,* which has $|V|+|E|$ vertices and $2|E|$ edges. Using the pre-flow push algorithm [3], this leads to a worst-case running time of $\mathcal{O}(|E|^3 \log |V|)$, or $\mathcal{O}(|V|^6 \log |V|)$ in dense graphs, for solving the separation problem for blossom inequalities.

Grötschel and Holland [5] observed that the above-mentioned max-flow computations can in fact be carried out on graphs with only $\mathcal{O}(|V|)$ vertices and $\mathcal{O}(|E|)$ edges. Although the idea behind this is simple, it reduces the overall running time to $\mathcal{O}(|V||E|^2 \log(|V|^2/|E|))$, which is $\mathcal{O}(|V|^5)$ in the case of a dense graph.

In this paper we give a separation algorithm for blossom inequalities which has running time $\mathcal{O}(|V|^2|E| \log(|V|^2/|E|))$, or $\mathcal{O}(|V|^4)$ in dense graphs. As well as being faster than the Padberg-Rao and Grötschel-Holland approaches, it is simpler and easier to implement.

Besides for matching problems, the algorithm is also important for solving the *Traveling Salesman Problem* (TSP). The special blossom inequalities obtained when $b_i = 2$, for all i, are valid for the TSP, and facet-inducing under mild conditions (see Grötschel and Padberg [7, 8]). Thus we obtain a faster separation algorithm for the TSP as a by-product. In fact, the algorithm is applicable to a general class of cutting planes for integer programs, called $\{0, \frac{1}{2}\}$-Chvátal-Gomory cuts, see Caprara and Fischetti [1].

2 Review of the original algorithm

For the purposes of clarity, we will briefly review the original algorithm.

Let x^* be a given vector to be separated and suppose that it satisfies the degree equalities and bounds. Padberg and Rao proceed as follows. Take the graph G and divide each edge e into two halves by adding an extra vertex, which we call *splitting vertex*, and denote by k_e. One half (which we shall call the *normal half*) is given weight x^*_e, whereas the other half (which we shall call the *flipped half*) is given weight $1 - x^*_e$. The resulting graph $\widehat{G} = (\widehat{V}, \widehat{E})$ is called the *split graph.* It has $|\widehat{V}| = |V| + |E|$ vertices and $|\widehat{E}| = 2|E|$ edges.

Next, Padberg and Rao assign parities to the vertices. For $i \in V$, let f_i denote the number of flipped edges which are incident to the vertex i of $\widehat{G}$. A vertex is labelled *even* or *odd* according to whether $b_i + f_i$ is an even or odd number. It is not difficult to see that there is a violated blossom inequality (4) if and only if there exists an odd cut of weight strictly less than 1 in the split graph. Here a cut $\delta(U')$ is called odd if each of its two shores U' and $\overline{U'}$ contains an odd number of odd vertices. If $\delta(U')$ is an odd cut in the split graph with $\hat{x}(\delta(U')) < 1$, then $U' \cap V$ is the handle of a violated blossom inequality.

This construction enabled Padberg and Rao [10] to use their generic minimum weight odd cut algorithm. This algorithm involves running the classical *cut-tree* algorithm of Gomory and Hu [4] on the split graph, with the odd vertices as terminals. This means that a maximum flow algorithm must be run several times on the split graph (or on certain graphs obtained from the split graph by contraction of edges). As mentioned in the introduction, the number of odd vertices is at least $|E|$ and at most $|V| + |E|$ leading to an overall running time of $\mathcal{O}(|E|^3 \log |V|)$, or $\mathcal{O}(|V|^6 \log |V|)$ in the case of a dense graph.

Grötschel and Holland's improvement

As mentioned in the introduction, a significant improvement to this algorithm was made by Grötschel and Holland [5]. (In fact, they were only interested in 2-matchings, but the same argument applies in the general case.) Consider an arbitrary edge $e = \{i, j\} \in E$. It is obvious that the maximum amount of flow which can be directly sent from i to j in the split graph is equal to the minimum of x^*_e and $1 - x^*_e$. From this observation it follows that prior to computing a max flow, for every edge $e \in E$, one of the two edges incident to the vertex $k_e \in \widehat{V}$ can be contracted. This means that the max-flow problems can be conducted on graphs which are similar in size to G (only the source and sink vertices can be in $\widehat{V} \setminus V$). However, the number of max-flows which have to be computed is still at least $|E|$ and at most $|V| + |E|$, leading to an overall running time of $\mathcal{O}(|V||E|^2 \log(|V|^2/|E|))$, which is $\mathcal{O}(|V|^5)$ in a dense graph.

3 The new algorithm

In this section we give an algorithm which separates blossom inequalities in time $\mathcal{O}(|V|^2|E| \log(|V|^2/|E|))$. In particular, we will need only $|V| - 1$ max-flow computations, and these can be performed in the original graph G.

The basic idea of the algorithm is to compute a cut-tree for G with terminal vertex set V and weights w, where w is defined by letting, for every edge $e \in E$

$$w_e := \min(x^*_e, 1 - x^*_e).$$

Algorithm 1 Blossom_Separation

```
Input: Graph $G$ and vector $x^*$.
Output: A violated blossom inequality, if one exists.
1: Compute a cut-tree for $G$ with weights $w := \min(x^*_e, 1 - x^*_e)$ and $V$ as set of
   terminal vertices by using any cut-tree algorithm.
2: For each of the $|V| - 1$ cuts $\delta(W)$ stored in the cut-tree do
3:   Find the best set $F$ of teeth for the candidate handle $W$, i.e., among all sets
     $F \subseteq \delta(W)$ with $b(W) + |F|$ odd, find one which minimizes the value
     $x^*(\delta(W) \setminus F) + |F| - x^*(F)$.
4:   If $x^*(\delta(W) \setminus F) + |F| - x^*(F) < 1$ then
5:     Output the violated blossom inequality. Stop.
6:   End if
7: End for
8: Output "There exists no violated blossom inequality."
```

Then, we take each of the $|V| - 1$ cuts $\delta(W)$ stored in the cut-tree and consider W as a candidate for the handle of a violated blossom inequality. For each candidate handle we find the best set of teeth, and thus decide if there exists a violated blossom inequality with this handle. Our blossom separation algorithm is displayed as algorithm 1. (Note that the algorithm can be modified to find the most violated blossom inequality.)

Finding the best set of teeth (step 3) can be done by sorting (see [5]), which would take $\mathcal{O}(|E| \log |E|)$ time (which is sufficiently fast to achieve the improved running time of the algorithm). It is possible to reduce the time for step 3 to $\mathcal{O}(|E|)$ by the following easy observation (this fact is implicit in [12]). Let F denote the set of edges $e \in \delta(W)$ with $1 - x^*_e < x^*_e$. If $b(W) + |F|$ is odd, then the minimum is achieved by this set F. Otherwise, let $f \in \delta(W)$ be the edge which minimizes the term $\max(x^*_e, 1 - x^*_e) - \min(x^*_e, 1 - x^*_e)$ over all $e \in \delta(W)$. If we define F' to be the symmetric difference of the sets F and $\{f\}$, then $b(W) + |F'|$ is odd, and F' is an optimal set of teeth for the handle W.

Step 3 has to be carried out at most $|V|$ times, leading to a running time of $\mathcal{O}(|V||E|)$ for the loop in steps 2–7. Thus, the running time is dominated by the computation of the $|V| - 1$ max-flows in step 1.

In section 5, we will prove that algorithm 1 solves the separation problem for blossom inequalities (4). Before we do that, for the purpose of clarity, we repeat some facts about cut-trees and minimum odd cuts in the next section.

4 Cut-trees and minimum odd cuts

The contents and terminology of this section refer to graphs, cut-trees and minimum odd cuts in general. If $G = (V, E)$ is a graph with positive edge weights and $p, q \in V$ are distinct vertices, then we denote by $\lambda(p, q) := \lambda_G(p, q)$ the weight of a minimum (p, q)-cut in the graph G with respect to the edge weights.

The Gomory-Hu algorithm

Given a graph with positive edge weights and a set of *terminal vertices* $T \subseteq V$, a cut-tree is a tree whose nodes, called *supernodes,* are sets which partition the vertex set of G. Each supernode contains precisely one terminal vertex which is called its *representative.* The edges of the cut-tree are weighted. If A and B are two adjacent supernodes and r_A and r_B are their respective representatives, then the weight of the edge $\{A, B\}$ of the cut-tree equals $\lambda(r_A, r_B)$, the value of the maximum (r_A, r_B)-flow in G.

Since the supernodes form a partition of the vertex set of G, every edge of the cut-tree defines a cut in G, because its removal defines a partition of V into two sets. We say that the cut-tree edge *induces* this cut. In the definition of a cut-tree, we require that, if A and B are two adjacent supernodes and r_A and r_B are their respective representatives, then the edge $\{A, B\}$ of the cut-tree induces a minimum (r_A, r_B)-cut in G.

Gomory and Hu [4] gave an algorithm which produces a cut-tree by performing $|T| - 1$ max-flow computations. Their algorithm starts with a (trivial) cut-tree with terminal vertex set $\{r\}$ and successively performs the following tree expansion step, which enlarges the terminal vertex set.

Tree expansion. Suppose that C is a cut-tree with terminal vertex set $T \subsetneq V$, and let $t \in V \setminus T$ be contained in the supernode R of C, whose representative is $r \in T$. Let $S_1, \dots, S_l$ be the neighbors of R in C, and for $i = 1, \dots, l$ let $\delta(U_i)$ be the cut induced by the edge $\{R, S_i\}$ of C, where $S_i \subseteq U_i$. Construct the graph G_R by identifying in G, for each $i = 1, \dots, l$, the set U_i to a single vertex s_i, where loops are deleted and parallel edges are merged while adding their weights. Now let $\delta(X)$ be a minimum (r, t)-cut in the graph G_R. Construct a new tree C$'$ out of C by replacing the supernode R of C with two supernodes $R \cap X$ and $R \setminus X$, which are linked by an edge with weight $\lambda(r, t)$. Then for each edge $\{R, S_i\}$ of C add to C$'$ an edge $\{R \cap X, S_i\}$ if $s_i \in X$ or an edge $\{R \setminus X, S_i\}$, if $s_i \notin X$, while keeping the original weight.

Knowledge of this expansion step will be relevant in section 5.

Theorem 1 **([4]).** *C$'$ is a cut-tree for G with terminal vertex set $T \cup \{t\}$.*

For completeness, we also note the uncrossing lemma of Gomory and Hu [4].

Lemma 1 **([4]).** *With the above terminology, there exists a minimum (r, t)-cut $\delta(Y)$ in G, which does not cross any of the cuts $\delta(U_i)$, $i = 1, \dots, l$, i.e., which satisfies $U_i \subseteq Y$ or $U_i \subseteq \overline{Y}$ for all $i = 1, \dots, l$. In particular $\lambda_{G_R}(r, t) = \lambda_G(r, t)$ holds.*

The Padberg-Rao min-odd-cut algorithm

Now we assume that $|T|$ is an even number. The elements of T are called *odd* vertices. A cut $\delta(U)$ in G is called *odd,* if $|T \cap U|$ is an odd number. The following theorem of Padberg and Rao [10] relates minimum odd cuts and cut-trees and gives rise to their minimum odd cut algorithm.

Theorem 2 ([10]). *Let* C *be a cut-tree for G with terminal vertex set T. We call a cut in* C *odd, if both shores contain an odd number of supernodes.*

Any minimum odd cut in C *induces a minimum odd cut in G.*

Clearly, a minimum odd cut in a tree must consist of a single edge. Thus there exists at least one edge of the cut-tree which induces a minimum odd cut.

In the next section, we will need the following lemma.

Lemma 2. *Suppose that the value of a minimum odd cut in G is strictly less than 1. Let C' be a cut-tree with a certain terminal vertex set $T' \subseteq V$. Denoting by S' the set of supernodes of C', we define the following set of odd supernodes:*

$$\bigl\{S \in \mathrm{S}' \;\bigm|\; |T \cap S| \text{ odd}\bigr\}.$$

Now, we call a cut in C' odd, if each of its shores contains an odd number of odd supernodes.

If $\lambda(p, q) \geq 1$ for every two vertices $p \in T$ and $q \in T'$ which are both contained in the same supernode of C', then any minimum odd cut in C' induces a minimum odd cut in G.

Proof. First we note that the argument of Padberg and Rao [10] which proves theorem 2 still works, if the terminal vertex set of the cut-tree is bigger than the set of odd vertices. This means that if C'' is a cut-tree for G with terminal vertex set $T'' \supset T$, and we call a supernode of C'' odd if its representative is odd, then any minimum odd cut of C'' induces a minimum odd cut in G.

Thus we consider a cut-tree C'' which is obtained from C' by performing tree expansion until the set of terminal vertices is enlarged from T' to $T' \cup T$. Because of what we have just noted, one of the edges of C'' induces a minimum odd cut in G. A minimum odd cut of G cannot separate two vertices $p \in T$ and $q \in T'$ which are contained in the same supernode of C', because $\lambda(p, q) \geq 1$ and the value of a minimum odd cut in G is strictly less than 1. Neither can it separate two vertices $p_1, p_2 \in T$, which are contained in the same supernode of C', because, if q is the representative of this supernode,

$$\lambda(p_1, p_2) \geq \min(\lambda(p_1, q), \lambda(q, p_2)) \geq 1.$$

So none of the edges which have been created by the tree expansion from C' to C'' induces a minimum odd cut in G. Hence, there must exist an edge in C', which induces one.

By the easy fact that by the definition of the set of odd supernodes of C', an edge of C' induces an odd cut in G if and only if it is an odd cut in C', the claim of the lemma follows.

5 Proof of correctness of the new algorithm

Let T denote the set of vertices of the split graph $\widehat{G}$ which are odd according to the Padberg-Rao labelling.

To prove the correctness of algorithm 1, we will first make clear that we can think of any cut-tree C_G of G with terminal vertex set V as a cut-tree of $\widehat{G}$ with terminal vertex set $V \subset \widehat{V}$. Then we will show how a cut-tree of $\widehat{G}$ containing a minimum odd cut might look like, if the value of the minimum odd cut is strictly less than 1. From this we will deduce that among the cuts $\delta(W)$ stored in the cut-tree C_G for G, there is one which is the handle of a violated blossom inequality, if one exists.

We need some notation. For every edge $e = \{i, j\} \in E$ define

$$\phi(e) := \begin{cases} i, & \text{if } \hat{x}_{\{i,k_e\}} \geq \hat{x}_{\{k_e,j\}}, \\ j, & \text{if } \hat{x}_{\{j,k_e\}} > \hat{x}_{\{k_e,i\}}, \end{cases}$$

i.e., $\phi(e)$ is the end vertex of e to which k_e is linked by the $\widehat{G}$-edge with *bigger* $\hat{x}$-weight.

Let C_G denote a cut-tree of G with weights w and set of terminal vertices V. Define a partition $\widehat{\mathrm{S}}$ of the set $\widehat{V} = V \cup \{k_e \mid e \in E\}$ by letting, for each $i \in V$,

$$S_i := \{i\} \cup \left\{ k_e \in \hat{V} \;\middle|\; e \in \delta_G(i) \wedge \phi(e) = i \right\},$$

which means we include a vertex k_e of $\widehat{G}$ corresponding to an edge e of G into that set S_i represented by the neighbor i of k_e to which it is linked by the $\widehat{G}$-edge with *greater* $\hat{x}$-value, i.e., $\hat{x}_{k_e,i} \geq \hat{x}_{k_e,j} = w_e$. See figure 1.

Apparently, these sets define a partition of the vertex set $\widehat{V}$ of $\widehat{G}$. Let $\mathrm{C}_{\widehat{G}}$ denote the tree which results from replacing for every $i \in V$ the supernode $\{i\}$ of C_G by the supernode S_i, while keeping the edges and edge weights. Then we have the following fact (without proof).

Lemma 3. *$\mathrm{C}_{\widehat{G}}$ is a cut-tree of $\widehat{G}$ with weights $\hat{x}$ and set of terminal vertices V.*

If we wanted to solve the blossom separation problem by computing a minimum odd cut in $\widehat{G}$, we could start with C_G, replace the supernodes as just described, and then follow the Gomory-Hu procedure to expand the cut-tree until the terminal vertex set is $V \cup T$. Because of Lemma 2 (with $T' := V \cup T$), we could find a minimum odd cut in this cut-tree.

We will see now that this is not necessary. Instead, we will just show how a cut-tree which contains a minimum odd cut in $\widehat{G}$ might look like. To do this, we simulate tree expansion steps to a point when the condition of Lemma 2 applies.

First note that $T \setminus V$ only contains split vertices. From lemma 2, we know that, if $k_e \in T$ is contained in the supernode together with $v \in V \cup T$, we need to consider the minimum (k_e, v)-cut only if its value $\lambda_{\widehat{G}}(k_e, v)$ is strictly less than 1. As the proof of the following proposition shows, in this case, a minimum (k_e, v)-cut can be obtained by modifying a cut from the cut-tree, and the update of the cut-tree is easy to predict.

Proposition 1. *Define the following subset of the (odd) vertices of $\widehat{G}$:*

$$Q := \left\{ k_e \;\middle|\; e \in E,\; \lambda_{\widehat{G}}(k_e, \phi(e)) < 1 \right\}.$$

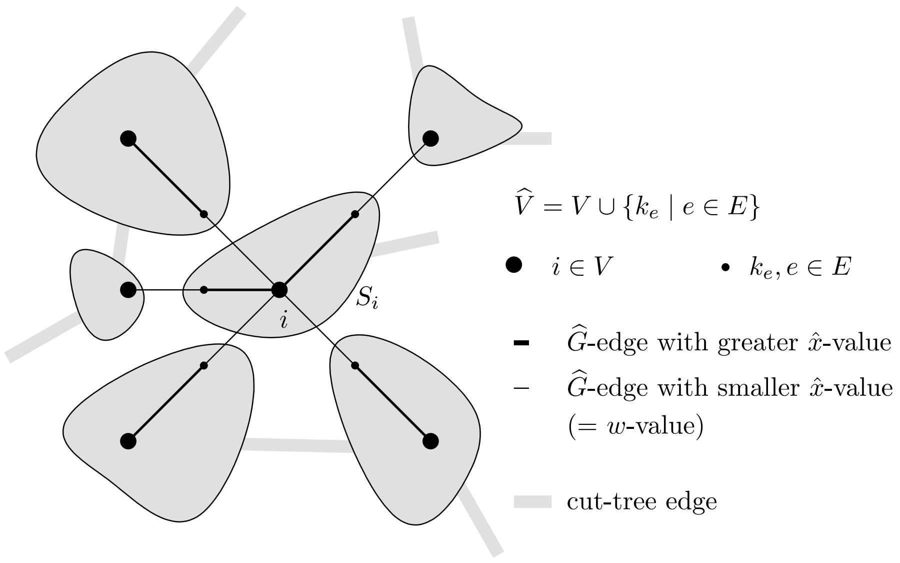
$\widehat{V} = V \cup \{k_e \mid e \in E\}$
● $i \in V$
• $k_e, e \in E$
$\widehat{G}$-edge with greater $\hat{x}$-value
$\widehat{G}$-edge with smaller $\hat{x}$-value (= w-value)
cut-tree edge
S_i
i

Fig. 1. A partition of $\widehat{V}$ based on C_G resulting in a cut-tree for $\widehat{G}$ with set of terminal vertices $V \subset \widehat{V}$.

There is a Gomory-Hu cut-tree C' *of* $\widehat{G}$ *with terminal vertex set* $V \cup Q$ *which is a subdivision of* $\mathrm{C}_{\widehat{G}}$.

To be more precise, every edge $\{S_i, S_j\}$ *is replaced by a path*

$$S'_i,\ \{k_{e_1}\},\ \ldots,\ \{k_{e_r}\},\ S'_j, \quad r \geq 0$$

of supernodes, where the $\{k_{e_h}\}$ *are supernodes of* C' *consisting of a single vertex* $k_{e_h} \in Q$*, and we let* $S'_i := S_i \setminus Q$ *and* $S'_j := S_j \setminus Q$.

Proof. The proof is by induction. We will perform expansion steps on a sequence of cut-trees, which have the property that for each $e \in E$ the vertex k_e is either contained in the supernode whose representative is $\phi(e)$, or it is itself a singleton supernode $\{k_e\}$, subdividing (with some other singleton supernodes maybe) an edge of $\mathrm{C}_{\widehat{G}}$.

The condition clearly holds for the first cut-tree in the sequence, i.e., before performing any tree expansion steps, namely $\mathrm{C}_{\widehat{G}}$.

For the induction step we perform tree expansion. For the notation, see the paragraph about tree expansion in section 4 above. We choose an arbitrary $k_e \in Q$ which is not a representative. By induction hypothesis, this implies that it is contained in a supernode R whose representative is $\phi(e) \in V$. We need to know how a minimum $(\phi(e), k_e)$-cut in G_R looks. Suppose that $e = \{\phi(e), j\}$, i.e., $j \in V$ is the other end of the edge e of G. One of the sets $U_1, \ldots, U_l$, say U_i, contains the vertex j.

Claim. We claim that

$$\delta(\{k_e, s_i\})$$

is a minimum $(\phi(e), k_e)$-cut in G_R.

If the claim is correct, then the tree expansion step produces the new supernodes $R \setminus \{k_e\}$ and $\{k_e\}$, thus making $\{k_e\}$ a singleton supernode. The other split vertices remain untouched, so the induction step is completed.

Proof of the claim. By the uncrossing lemma of Gomory and Hu (Lemma 1), we know that $\lambda_G(\phi(e), k_e) = \lambda_{G_R}(\phi(e), k_e)$. Thus, because $\lambda(\phi(e), k_e) < 1$ and $\hat{x}_{\{\phi(e),k_e\}} + \hat{x}_{\{k_e,j\}} = 1$, we know that j and $\phi(e)$ are on different shores of any minimum $(\phi(e), k_e)$-cut in G_R. This means that any minimum $(\phi(e), k_e)$-cut in G_R is also a $(\phi(e), s_i)$-cut, and this of course remains true if we move the vertex k_e to the $\phi(e)$-shore of such a cut. From this it follows that

$$\lambda_{G_R}(\phi(e), s_i) \leq \lambda_{G_R}(\phi(e), k_e) - \hat{x}_{\{\phi(e),k_e\}} + \hat{x}_{\{k_e,j\}}.$$

It is clear from the definition of a cut-tree that $\delta(s_i)$ is a minimum $(\phi(e), s_i)$-cut in G_R. So we obtain

$$\begin{aligned}\lambda_{G_R}(\phi(e), k_e) &\geq \lambda_{G_R}(\phi(e), s_i) + \hat{x}_{\{\phi(e),k_e\}} - \hat{x}_{\{k_e,j\}} \\ &= \delta(s_i) + \hat{x}_{\{\phi(e),k_e\}} - \hat{x}_{\{k_e,j\}} \\ &= \delta(\{k_e, s_i\}).\end{aligned}$$

Thus the claim is true and the proof of the proposition is finished.

We are now ready to complete the proof of correctness of algorithm 1.

Suppose there exists a violated blossom inequality. This implies that the value of the minimum odd cut in $\widehat{G}$ is strictly less than 1. From lemma 2, we know that a minimum odd cut of $\widehat{G}$ is stored in any cut-tree with terminal vertex set $V \cup Q$—in particular in the cut-tree C' whose existence is proven in the proposition. Let $\delta(U')$ be a minimum odd cut of $\widehat{G}$ induced by an edge of C'.

If we let $W := U' \cap V$, then there exists a set $F \subseteq \delta(W)$ such that $x(\delta(W) \setminus F) - x(F) \geq 1 - |F|$ is a blossom inequality which is violated by x^*. But from the proposition we know that the cut $\delta(W)$ is induced by an edge of the cut-tree C_G, namely the one which is used in algorithm 1. Hence the set W (or $\overline{W}$) is one of those which are considered as candidate handles in step 3 of algorithm 1, so the algorithm finds this (or another) violated inequality.

Hence the new algorithm is correct.

Remark 1. Our algorithm can be easily adapted to more general u-capacitated b-matching problems, perfect or otherwise. Details will be given in the full version of this paper [9].

References

1. Caprara, A., Fischetti, M.: $\{0, \frac{1}{2}\}$-Chvátal-Gomory cuts. Math. Program. **74** (1996) 221–235.

2. Edmonds, J.: Maximum matching and a polyhedron with 0-1 vertices. J. Res. Nat. Bur. Standards **69B** (1965) 125–130.
3. Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum flow problem. J. of the A.C.M. **35** (1988) 921–940.
4. Gomory, R.E., Hu, T.C.: Multi-terminal network flows. SIAM J. Applied Math. **9** (1961) 551–570.
5. Grötschel, M., Holland, O.: A cutting plane algorithm for minimum perfect 2-matching. Computing **39** (1987) 327–344.
6. Grötschel, M., Lovász, L., Schrijver, A.J.: Geometric Algorithms and Combinatorial Optimization. Wiley, New York (1988).
7. Grötschel, M., Padberg, M.W.: On the symmetric travelling salesman problem I: inequalities. Math. Program. **16** (1979) 265–280.
8. Grötschel, M., Padberg, M.W.: On the symmetric travelling salesman problem II: lifting theorems and facets. Math. Program. **16** (1979) 281–302.
9. Letchford, A.N., Reinelt, G., Theis, D.O.: Odd minimum cut-sets and b-matchings revisited. (in preparation)
10. Padberg, M.W., Rao, M.R.: Odd minimum cut-sets and b-matchings. Math. Oper. Res. **7** (1982) 67–80.
11. Padberg, M.W., Rinaldi, G.: Optimization of a 532-city symmetric traveling salesman problem by branch and cut. Oper. Res. Lett. **6** (1987) 1–7.
12. Padberg, M.W., Rinaldi, G.: Facet identification for the symmetric traveling salesman polytope. Math. Program. **47** (1990) 219–257.
13. Pulleyblank, W.R.: Faces of matching polyhedra. Ph.D thesis, University of Waterloo. (1973)