Undirected Postman Problems with Zigzagging Option: A Cutting-Plane Approach

Stefan Irnich [a,*]

[a] *Deutsche Post Endowed Chair of Optimization of Distribution Networks, RWTH Aachen University, Templergraben 64, D-52056 Aachen, Germany.*

Abstract

The paper devises a new model and associated cutting-plane and branch-and-cut approaches for a variant of the undirected Chinese and Rural postman problem where some of the edges offer the flexibility of either being serviced twice by two separate traversals or by a single zigzag traversal. The kernel of the proposed cutting-plane algorithm is a separation procedure for generalized blossom inequalities. We show that the currently best known separation procedure of Letchford *et al.* (2004) is applicable and leads to a highly efficient solution approach which can handle large-scale problem instances.

Key words: postman problems, zigzagging, polyhedral theory

1 Introduction

The paper by Irnich (2005) introduced postman problems with zigzagging option that can be described as follows: A postman has to deliver mail to the street segments of his district. Street segments can be divided into four classes: The first class consists of street segments with houses on one side of the street only. These require a single service, i.e., at least one traversal of the street segment. Second, there are street segments with houses on both sides, which have to be serviced separately. Third, some street segments with houses on both sides provide the option of either servicing them both with a single zigzag traversal or servicing the two sides separately. In all cases, additional traversals (so-called *deadheadings*) of a segment are allowed, but deadheadings and different modes of service can cause different costs. Fourth, so-called non-required street segments may be used by the postman to get from one point to another. The problem is to find a least-cost postman tour for a given district, providing an appropriate service for all street segments.

* Corresponding author.
Email address: `sirnich@or.rwth-aachen.de` (Stefan Irnich).

While the windy rural variant of this problem has been solved by its transformation into an ATSP (cf. Irnich, 2005), this paper considers the undirected version of the problem and a solution approach based on a new model. If the subgraph induced by edges which require servicing is connected, the problem is called the *undirected Chinese postman problem with zigzagging option* (UCPPZ). An instance of the UCPPZ is defined on an undirected connected graph $G = (V, E)$ with node set V and edge set E. Edges are partitioned into $E = E^0 \cup E^1 \cup E^2 \cup E^3$ where E^0 is the set of non-required edges, E^1 and E^2 are the sets of edges that require single and double service respectively (but zigzag service is not allowed), and E^3 is the set of edges that provide the zigzagging option. Recall that in *Chinese* postman problems the set $R = E \setminus E^0$ of *required edges* spans a connected subgraph of G. If R spans more than one component, the resulting problem is called the *undirected rural postman problem with zigzagging option* (URPPZ). Four different costs $c_e^k \in \mathbb{Q}_+$ with $k \in \{0, 1, 2, 3\}$ are associated with an edge $e \in E$. $k = 0$ stands for deadheading, $k = 1$ for servicing the first side, $k = 2$ the opposite side of the street segment, and $k = 3$ for zigzag service. Obviously, for the edges $e \in E \setminus E^3$ only some of the costs are relevant, i.e., edges $e \in E^2$ need three and edges $e \in E^1$ two cost coefficients, while edges $e \in E^0$ only need the cost coefficient c_e^0.

The contribution of this paper is twofold: First, we show that the UCPPZ can be interpreted as a T-join problem (Schrijver, 2003, p. 485f) and, therefore being solved by shortest path and matching algorithms. This approach efficiently solves the UCPPZ but is not applicable to extensions of the problem, such as the URPPZ or mixed, windy, and hierarchical postman problems with zigzagging option. Therefore, a mixed-integer linear programming formulation is needed to provide a basis for efficient cutting-plane and branch-and-cut solution algorithms. The second and main contribution is, therefore, the devising of a new model for the UCPPZ and URPPZ and the development of an associated branch-and-cut algorithm. The kernel of such a cutting-plane algorithm is a separation procedure for *generalized blossom inequalities* (blossom inequalities, also referred to as *odd-cut inequalities*, are well-known in the context of b-matchings as well as different types of postman problems). We show that the currently best known separation procedure of Letchford *et al.* (2004) is applicable and leads to a highly efficient solution approach which can handle large-scale problem instances.

The paper is structured as follows: We start in Section 2 with the solution of the UCPPZ as a T-join problem. Section 3 briefly reviews models for the undirected Chinese postman problem (UCPP). A new model for the UCPPZ is introduced in Section 4 and its validity is proven. An extension to the rural case is discussed in Section 5. Section 6 reviews and analyzes separation procedures for the blossom inequalities. Section 7 gives computational results, starting with a comparison of the T-join solution approach and the cutting-plane algorithm for the UCPPZ. Moreover, it shows the effectiveness of the branch-and-cut algorithm when applied to large-scale URPPZ instances. Final conclusions are given in Section 8.

2 T-joins and the Solution of the UCPPZ

We refer to (Schrijver, 2003, Sectoin 29.1) for notations and solution methods for T-join problems. First recall that in a graph $G = (V, E)$ and for a subset $T \subseteq V$ any subset $J \subseteq E$ is a *T-join* if T is the set of odd degree nodes in the subgraph spanned by J. Any T-join is the edge-disjoint union of circuits and $|T|/2$ paths connecting disjoint pairs of nodes of T. A *shortest T-join* in the weighted graph $G = (V, E, c)$ is a T-join J^* with $c(J^*) = \min_{J:J \text{ is a } T\text{-join}} c(J)$. For *non-negative* edge weights $c \in \mathbb{Q}_+^{|E|}$, a shortest T-join can be found by the following procedure: Compute the shortest-path lengths d_{ij} in (V, E, c) for all $i, j \in T$. The corresponding shortest paths are referred to as p_{ij}. Compute a minimum weight perfect matching M in the complete weighted graph $K_T = (T, E_T, d)$ over the node set T. The symmetric difference of the edges of p_{ij} for $\{i, j\} \in M$ is a shortest T-join in the weighted graph (V, E, c).

A solution to the *undirected Chinese postman problem* (UCPP) can be found as a shortest T-join in G with edge weights c^0, i.e., edge weights corresponding to the costs of deadheadings. This is essentially the procedure first proposed by Edmonds (1965). Note that the edges of paths induced by the perfect matching do not produce circuits or multiple copies of edges as long as all edge weights are positive. Thus, the symmetric difference of edges implied by the matching is identical to the union of the edges. Moreover, the same procedure can be applied to UCPPs with additional edges for deadheading and edges requiring two services, i.e., with $E = E^0 \cup E^1 \cup E^2$. Here edges $e \in E^0 \cup E^2$ simply do not contribute to the edge degrees (or their parities).

The UCPPZ with $E^3 \neq \varnothing$ requires a different solution procedure: Let T be the set of odd nodes in $(V, E^1 \cup E^3)$, i.e., nodes incident to an odd number of edges in $E^1 \cup E^3$. Define a new graph $G' = (V, E', c')$, in which $E' = E \cup p(E^3)$. All edges $e \in E$ are weighted with the deadheading costs $c'_e = c^0_e$ and additional edges $p(e)$ parallel to $e \in E^3$ are weighted with $c'_{p(e)} = c^1_e + c^2_e - c^3_e$. A shortest T-join J' of G' directly implies an optimal solution to the UCPPZ: Note first that the definition of odd nodes T reflects the assumption that initially all edges $e \in E^3$ are serviced by zigzagging, i.e., the edges $e \in E^3$ contribute with $d_e = 1$ to the node degrees. Hence, the interpretation of a solution to the T-join problem is the following: The presence of an edge $p(e) \in J'$ in the T-join means that the initial assumption is discarded, i.e., instead of a single zigzag traversal, the edge e is serviced by two separate traversals implying a change of the costs given by $c'_{p(e)}$. Edges with $p(e) \notin J'$ correspond to those street segments $e \in E^3$ serviced by zigzagging. All other edges $e \in E \subset E'$ in the T-join imply deadheadings.

A subtle complication arise from the fact that $c'_{p(e)} = c^1_e + c^2_e - c^3_e$ may be negative, so that the above procedure cannot be applied directly. However, a T-join problem with positive and negative edge weights can be transformed into an equivalent T''-join problem with positive weights only (cf. Schrijver, 2003, p. 485): Let $N = \{e \in E' : c'_e < 0\}$. The T-join problem in G' with weights c' can be solved as a T''-join problem, where T'' is the symmetric difference of T and the odd nodes in (V, N). Moreover, new edge weights are defined as the absolute values of the original weights, i.e., $c''_e = |c'_e|$. A shortest T-join w.r.t. c' can then be computed as the symmetric difference of J'' and N, where J'' is a shortest T''-join w.r.t. c''.

Concluding, a UCPPZ instance with $n = |V|$ nodes and $m = |E|$ edges can be solved in $\mathcal{O}(n(m + n \log n))$ time: First, the transformation into a T-join problem without negative edge weights takes $\mathcal{O}(m)$ time. Second, the solution of the all-pairs shortest path problem takes $\mathcal{O}(n(m + n \log n))$ time (Fredman and Tarjan, 1984). Third, the solution of the weighted perfect matching problem can be achieved in $\mathcal{O}(n(m + n \log n))$ time (Gabow, 1990).

3 Models for the UCPP

Although the UCPP(Z) can be solved efficiently as T-join problem, there is still a need for appropriate integer-programming formulations of the problem. First, knowing more about the structure of the associated polyhedra is of theoretical interest in itself. Second, extensions of the UCPP(Z) cannot be solved as T-join problems and, thus, cutting-plane or branch-and-cut algorithms are among the reasonable and promising solution approaches.

The basic analysis of UCPP models and their polyhedra goes back to the work of Edmonds and Johnson (1973). They assume that the underlying graph $G = (V, E)$ is connected. Variables x'_e indicate the number of deadheadings through each edge $e \in E$. A solution to the UCPP requires the selection of edges for deadheading such that the augmented graph has nodes of even degree only. As usual, we use the following compact notation: Given vectors $\alpha \in \mathbb{R}^{|V|}$ and $\beta \in \mathbb{R}^{|E|}$, $\alpha(W)$ refers to the sum $\sum_{i \in W} \alpha_i$ and $\beta(F)$ to the sum $\sum_{e \in F} \beta_e$ (for $W \subseteq V, F \subseteq E$). For any subset $S \subseteq V$, the *cut set* $\delta(S)$ is $\{e \in E : e \in (S : V \setminus S)\}$. Edmonds and Johnson (1973) formulated the UCPP with so-called *blossom* (or *odd-cut*) inequalities of the form $x'(S) \geq 1$ for $S \subseteq V$ with $|\delta(S)|$ odd. Defining $d(S) = |\delta(S)|$, the blossom model is

$$\begin{aligned}
\min \quad & {c^0}^\top x' \\
\text{s.t.} \quad & x'(S) \geq 1 \qquad \text{for all } S \subseteq V\text{: } d(S) \text{ odd} && \text{(1a)} \\
& x' \geq 0. && \text{(1b)}
\end{aligned}$$

Let P^{UCPP} be the set of feasible solutions to (1). It is one of the main results of Edmonds and Johnson (1973) that all extreme points of P^{UCPP} are integral and represent feasible solutions to the UCPP. In order to solve the UCPP, one can use model (1), which requires the solution of a 1-matching problem (cf. Edmonds and Johnson, 1973). Note, however, that the P^{UCPP} contains (interior) integer solutions x' that do not imply feasible solutions to the UCPP, since the corresponding augmented graph is not even.

An important remark is that both models remain valid if we demand for multiple services along the edges. From now on, let $d_e \in \mathbb{Z}_+$ denote the *minimum number of traversals* of edge $e \in E$. We also redefine the node degree to be $d_i = d(\{i\})$

and $d(S) = \sum_{e\in\delta(S)} d_e$, so that the model (1) is still well-defined. By substituting G by a non-simple graph that contains d_e 'parallel' copies of edge e, the validity of the models follows with straightforward arguments if $d_e \geq 1$ holds for all $e \in E$. Moreover, the case $d_e = 0$ for some $e \in E$ is identical to the case $d_e = 2$ w.r.t. the node degree congruences modulo 2; the only difference is an offset of $2c^0_e$ in the objective. As long as $R = \{e \in E : d_e > 0\}$ spans a connected graph, model (1) solve this generalized UCPP.

Finally, it will be more convenient for the analysis undertaken in the following section to consider the *number of traversals* instead of the number of deadheadings. This is the substitution of x' by variables x which fulfill $x'_e = x_e - d_e$ for all $e \in E$. The resulting UCPP model, also valid for $d \in \mathbb{Z}^{|E|}_+$, is

$$\begin{aligned}
\min\ & {c^0}^\top x \qquad \left(-{c^0}^\top d \text{ const}\right) & \\
\text{s.t.}\ & x(\delta(S)) \geq d(S) + 1 \qquad \text{for all } S \subseteq V\text{: } d(S) \text{ odd} & \text{(2a)}\\
& x \geq d. & \text{(2b)}
\end{aligned}$$

4 A Model for the UCPPZ

We propose a model for the UCPPZ that uses traversal variables x_e for all $e \in E$ and indicator variables $y_e \in \{0,1\}$ for all edges $e \in E^3$. $y_e = 1$ indicates that both sides of the street are serviced separately and $y_e = 0$ that they are serviced simultaneously by zigzagging. Once all zigzagging decisions have been taken, the UCPPZ reduces to a simple UCPP. Recall that $d_e \in \mathbb{Z}_+$ has been defined as the *minimum number of traversals* of edge $e \in E$. For the UCPPZ it means, $d_e = 0$ for $e \in E^0$, $d_e = 1$ for $e \in E^1 \cup E^3$, and $d_e = 2$ for $e \in E^2$. For a given $\bar{y} \in \{0,1\}^{|E^3|}$, the vector $d^{\bar{y}} \in \mathbb{Z}^{|E|}_+$ defined by $d^{\bar{y}}_e = d_e$ for $e \in E \setminus E^3$ and $d^{\bar{y}}_e = d_e + \bar{y}_e = 1 + \bar{y}_e$ for $e \in E^3$ qualifies the number of *service traversals* for each edge. Thus, let UCPP($\bar{y}$) be the undirected Chinese postman problem implied by $d^{\bar{y}}$. The new model for the UCPPZ is

$$\begin{aligned}
\min\ & {c^0}^\top x + \sum_{e\in E^3} (c^1_e + c^2_e - c^0_e - c^3_e) y_e \qquad (\text{+constant term}) & \text{(3a)}\\
\text{s.t.}\ & x(\delta(S)) - 2y(F) \geq d(S) - |F| + 1 \qquad \text{for all } S \subseteq V,\ F \subseteq \delta(S) \cap E^3\text{:}\\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad d(S) + |F| \text{ odd} & \text{(3b)}\\
& x_e \geq d_e \qquad \text{for all } e \in E \setminus E^3 & \text{(3c)}\\
& x_e \geq 1 + y_e \qquad \text{for all } e \in E^3 & \text{(3d)}\\
& y \in \{0,1\}^{|E^3|}. & \text{(3e)}
\end{aligned}$$

We refer to inequalities (3b) as *generalized blossom inequalities* and we will prove the validity of the above model by the following propositions.

Proposition 1 *For any $y = \bar{y} \in \{0,1\}^{|E^3|}$, the generalized blossom inequalities* (3b) *are valid for UCPP($\bar{y}$).*

Proof: Choose any pair (S, F) with $d(S) + |F|$ odd. By defining $I = I(\bar{y}) = \{e \in E_3 : \bar{y}_e = 1\}$, one gets $d^{\bar{y}}(S) = d(S) + |I \cap \delta(S)|$ and $|I \cap F| = \bar{y}(F)$.

Case 1: $d^{\bar{y}}(S)$ is odd. Inequality (2a) for UCPP($\bar{y}$) imposes

$$\begin{aligned} x(\delta(S)) &\geq d^{\bar{y}}(S) + 1 = d(S) + |I \cap \delta(S)| + 1 \\ &= d(S) + |I \cap \delta(S)| + |F| - |F| + 1 \\ &\geq d(S) + 2|I \cap F| - |F| + 1 = d(S) + 2\bar{y}(F) - |F| + 1 \end{aligned}$$

Case 2: $d^{\bar{y}}(S)$ is even. Summing up the lower bounds (2b) for all $e \in \delta(S)$ of the UCPP($\bar{y}$) model implies

$$\begin{aligned} x(\delta(S)) &\geq d^{\bar{y}}(S) = d(S) + |I \cap \delta(S)| + |F| - |F| \\ &= d(S) + (|I \cap F| + |(I \cap \delta(S)) \setminus F|) + (|I \cap F| + |F \setminus I|) - |F| \\ &= d(S) + 2|I \cap F| - |F| + (|(I \cap \delta(S)) \setminus F| + |F \setminus I|) \\ &\geq d(S) + 2\bar{y}(F) - |F| + 1 \end{aligned}$$

The last inequality follows from the fact that $d(S) + |I \cap \delta(S)|$ is even, while $d(S) + |F|$ is odd, and, therefore, $I \cap \delta(S) \neq F$. This implies $(I \cap \delta(S)) \setminus F \neq \varnothing$ or $F \setminus (I \cap \delta(S)) = F \setminus I \neq \varnothing$, so that $(|(I \cap \delta(S)) \setminus F| + |F \setminus I|)$ is at least 1. Concluding, in both cases inequality (3b) holds. $\diamond$

Proposition 2 *For any $y = \bar{y} \in \{0,1\}^{|E^3|}$, (3) is a valid model for UCPP($\bar{y}$).*

Proof: We show that model (3) for given $\bar{y}$ reduces to model (2) with $d = d^{\bar{y}}$. First, variables x and the non-fixed part of the objective are identical. Second, the lower bounds (2b), i.e., $x \geq d^{\bar{y}}$, are equivalent to (3c) and (3d). Third, all constraints (3b) are shown to be valid for UCPP($\bar{y}$) by Proposition 1. What remains to show is that *all* blossom inequalities (2a), i.e., $x(\delta(S)) \geq d^{\bar{y}}(S) + 1$ for an arbitrary $S \subseteq V$ with $d^{\bar{y}}(S)$ odd, are contained in model (3). For any inequalities (2a), i.e., an $S \subseteq V$ with $d^{\bar{y}}(S)$ odd, there exists an equivalent inequality (3b). By defining F as $\{e \in \delta(S) : \bar{y}_e = 1\}$, the result follows directly from $|F| = \bar{y}(F)$ and $d^{\bar{y}}(S) = d(S) + |F|$. $\diamond$

Theorem 1 *Model (3) is valid for UCPPZ.*

Proof: Because of Proposition 2, the only aspect left to prove is that the objective (3a) is correct. Given any $\bar{y} \in \{0,1\}^{|E^3|}$, the number of deadheadings is determined by $x'_e = x_e - d^{\bar{y}}_e$. Therefore, the contribution of an edge $e \in E^3$ to the cost of the postman tour is

$$\begin{aligned} &c^0_e x'_e + (c^1_e + c^2_e) y_e + c^3_e (1 - y_e) \\ =& c^0_e (x_e - d_e - y_e) + (c^1_e + c^2_e) y_e + c^3_e - c^3_e y_e \\ =& c^0_e x_e + (c^1_e + c^2_e - c^0_e - c^3_e) y_e + (c^3_e - c^0_e d_e), \end{aligned}$$

where the last term in brackets is constant and the other terms are included in (3a). Those terms in the objective (3a) which belong to edges $e \in E \setminus E^3$

are identical to the ones in the UCPP model. Hence, the objective (3a) contains all cost-relevant components. $\diamond$

Since the number of generalized blossom inequalities (3b) is, in general, exponential in $|V|$, the new model (3) cannot be solved directly with integer linear programming techniques except for tiny and trivial instances. Hence, we propose to apply a cutting-plane procedure: Start with the relaxed model (3a), (3c), (3d), and $0 \leq y_e \leq 1$ for all $e \in E^3$, and dynamically add violated inequalities (3b) until no more violated inequalities exist. Section 6 will discuss separation procedures which solve the subproblem of identifying violated inequalities (3b) in polynomial time. As a consequence of the ellipsoid method, solving the LP-relaxation can also be performed in polynomial time (see Grötschel *et al.*, 1981).

Let P^{LP} be the polyhedron of the linear-programming relaxation of (3). As in the UCPP case, P^{LP} is a polyhedron that contains integer solutions $(\bar{x}, \bar{y})$ that are infeasible to our postman problem. However, infeasible integer solutions are never results of the cutting-plane algorithm if LPs are solved with the simplex algorithm (solutions are extreme points). This follows from Proposition 2 and the properties of the UCPP models (1) and (2).

Finally, we denote by P^I the convex hull of feasible integer solutions to (3). Clearly, $P^I \subseteq P^{LP}$. For the moment, it is an open question as to whether equality holds or not. If $P^I = P^{LP}$, the direct consequence would be that the cutting-plane procedure could always solve the UCPPZ to optimality. If $P^I \subsetneq P^{LP}$, a branch-and-cut procedure (Padberg and Rinaldi, 1991) can be applied to solve UCPPZ: Whenever the cutting-plane approach terminates with a fractional solution (x^*, y^*), one can select a fractional variable for branching. Because of Proposition 2, one can restrict oneself to branching on fractional variables $\bar{y}_e$, so that the depth of the branch-and-bound tree can never exceed $|E^3|$.

5 The Undirected Rural Postman Problem with Zigzagging Option

If the required edges $R = E \setminus E^0$ span more than one component in G, i.e., the spanning subgraph $G(R) = (V_R, R)$ is not connected, the resulting problem is an *undirected rural postman problem* (URPP). Pre-processing techniques allow the simplification of any problem instance to the case that $V_R = V$ holds (cf. Christofides *et al.*, 1981; Eiselt *et al.*, 1995); we assume $V = V_R$ in the following. A model for the URPP extends the UCPP formulation (1) by *connectivity constraints* $x'(S) \geq 2$ for all $S \subset V$ with $\varnothing \neq S \subsetneq V = V_R$ and $d(S) = 0$, as shown by Corberán and Sanchis (1994). Obviously, this translates into

$$x(\delta(S)) \geq 2 \qquad \text{for all } S \subsetneq V, S \neq \varnothing \text{ with } d(S) = 0 \tag{4}$$

for model (2), because $d(S) = 0$ is equivalent to $\delta(S) \subseteq E^0$ implying $x_e = x'_e$ for all $e \in \delta(S)$. The separation of violated connectivity constraints can be performed on a graph in which all edges $e \in R$ are shrunk. Nodes of the shrunk graph correspond to components of $G(R)$ and edge weights are the flows between the components. The algorithm of Gomory and Hu (1961) can be used to efficiently find cut sets S_R with capacity less than 2 in the shrunk graph. These impose sets $S \subset V$ of violated connectivity constraints (4).

Extending model (3) by (4) yields a model for the URPPZ. This follows with the same arguments as those used in Section 4. Since the resulting model extends the URPP model, the solution of the LP-relaxation can be fractional, so that a branch-and-cut procedure has to be applied. Additional cutting planes, referred to as *R-odd cuts*, have been used by Corberán and Sanchis (1994) and Ghiani and Laporte (2000). These have exactly the form of blossom inequalities (1a) (or their reformulation (2a)) if d is appropriately defined. The generalized blossom inequalities (3b) already include the R-odd cuts for $F = \varnothing$ and, therefore, they can be separated with any method used in the following section.

6 Split Graph and Separation Procedures

The heart of our solution method, the separation procedure for generalized blossom inequalities, is inspired by the work of Padberg and Rao (1982). In 1982, Padberg and Rao developed a cutting-plane algorithm to solve the separation sub-problem for b-matching problems. In the b-matching case, *blossom inequalities* are of the form $x(\delta(S)) - 2x(F) \geq 1 - |F|$ for all $S \subseteq V, F \subseteq \delta(S)$ with $b(S) + |F|$ odd. Notice the similarity between these and our generalized blossom inequalities (3b), where the essential difference is that $x(F)$ is replaced by the term $y(F)$ in new variables y_e for the edges $e \in E^3$ from a (proper) subset of E.

We will use the terminology of (Letchford *et al.*, 2004) to describe the basic solution procedure, originally proposed by Padberg and Rao, as well as several improvements that lead to enhanced separation procedures with better worst-case running time. Given a point (x^*, y^*) outside the polyhedron P^{LP}, the separation procedure solves the problem of finding a generalized blossom inequality (3b), i.e, determines sets $S \subseteq V$ and $F \subseteq \delta(S)$ with $d(S) + |F|$ odd, separating (x^*, y^*) from P^{LP}. In order to solve the separation problem, construct a so-called *split graph* $\hat{G} = (\hat{V}, \hat{E})$ from $G = (V, E)$ by the following rules. All of the original edges $e \in E \setminus E^3$ are also in $\hat{E}$ and weights of $\kappa_e := x^*_e - d_e$ are assigned to them. Contrary, the edges $e = \{i, j\} \in E^3$ are divided into two halves by adding an extra node k_e. One half (the so-called *normal half*) is $\{i, k_e\} \in \hat{E}$ with a weight of $\kappa_e := x^*_e - d_e$, and the second half (the so-called *flipped half*) is $f_e = \{k_e, j\}$ with a weight of $\kappa_e + \mu_e$, where $\mu_e := 1 - 2y^*_e$. Because of the obvious correspondence, nodes corresponding to nodes of the original graph are denoted by $V \subset \hat{V}$. All nodes $k_e, e \in E^3$ of the split edges constitute the subset $K \subset \hat{V}$. The split graph $\hat{G}$ has $|V| + |E_3|$ nodes and $|E| + |E_3|$ edges. Nodes $i \in K \cup V$ are labeled *odd* if $d(\{i\})$ plus the number of flipped edges incident to i is odd; otherwise they are labeled *even*. Note that all $i \in K$ are odd.

It is straightforward to prove that there exists a violated generalized blossom inequality (3b) if and only if $\hat{G}$ contains an odd minimum cut $\delta(\hat{S})$ with capacity less than 1. First, the weighted graph $\hat{G}$ is well-defined since all weights $\kappa_e = x^*_e - d_e$ and $\kappa_e + \mu_e$ are non-negative as long as constraints (3c) and (3d) hold. Second, any odd minimum cut set $\hat{S}$ with capacity less than 1 must include at least one original node from $V \subseteq \hat{V}$. This follows from the observation, that the sum of the capacities of a flipped edge and its corresponding normal edge are always greater or equal to one ($\kappa_e + (\kappa_e + \mu_e) = 2(x^*_e - y^*_e) - 2 + 1 \geq 1$). Consequently, any cut with capacity less than 1 cannot contain both edges, i.e., the cut set cannot contain k_e without at least one of the corresponding endpoint from V. Finally, one gets the violated generalized blossom inequality by defining $S := \hat{S} \cap V \neq \varnothing$ and $F := \{e \in \delta(S) : f_e \in \delta(\hat{S})\}$. Herein, $\hat{S}$ is odd in $\hat{G}$ if and only if $d(S) + |F|$ is odd.

We next derive results about the worst-case running time of separation procedures that are based on the split graph. Note first, that the split graph has $p := |V| + |E^3|$ nodes and $|E| + |E^3| = \mathcal{O}(m)$ edges. The odd-min-cut algorithm of Padberg and Rao (1982) requires a maximum of $\mathcal{O}(p)$ max-flow computations, each of which can be performed in $\mathcal{O}(pm \log(p^2/m))$ time using the preflow-push algorithm of Goldberg and Tarjan (1986). The result is an $\mathcal{O}(p^2 m \log(p^2/m))$ separation procedure. For UCPPZ instances with $|E^3| = \Theta(|E|)$ zigzag edges, i.e., $p = \mathcal{O}(m)$, the resulting complexity of the separation procedure is $\mathcal{O}(m^3 \log m)$. Grötschel and Holland (1987) reduced the effort by shrinking flipped and normal edges of the split graph in the underlying max-flow computations. The result for the UCPPZ is that in the worst-case time for the separation reduces to $\mathcal{O}(pnm \log(n^2/m))$. Hence, for UCPPZ instances on dense graphs, i.e., $m = \Theta(n^2)$, with $|E^3| = \Theta(|E|)$ zigzag edges, the time bound is $\mathcal{O}(n^5)$.

An even faster separation procedure with worst-case bound $\mathcal{O}(n^2 m \log(n^2/m))$ has been developed by Letchford *et al.* (2004). Again, their procedure remains applicable to our UCPPZ separation problem yielding an $\mathcal{O}(n^4)$ algorithm in the dense graph case. The adaptation of the procedure proposed by Letchford *et al.* (2004) to the UCPPZ works as follows: Consider the original graph $G = (V, E)$ but with edge weights $w_e = \kappa_e$ for $e \in E \setminus E^3$ and $w_e = \min\{\kappa_e, \kappa_e + \mu_e\}$ for $e \in E^3$. This graph is called *support graph* in the following. Construct a cut tree of the support graph with terminal nodes V by any cut-tree algorithm (the classical Gomory and Hu (1961) algorithm in combination with a pre-flow push algorithm (e.g., Goldberg and Tarjan, 1986) guarantees the $\mathcal{O}(n^2 m \log(n^2/m))$ worst-case bound). For each cut set $S \subset V$ stored in the cut tree, compute the best set $F \subseteq \delta(S)$ minimizing $x(\delta(S)) - 2y(F) - |F|$. Sets S are potential *handles* of a blossom inequality. Finding a best set $F = F(S)$ (of so-called *teeth*) can be computed with a linear algorithm in $\mathcal{O}(|\delta(S) \cap E^3|)$ steps. We refer the reader to (Letchford *et al.*, 2004) and (Letchford *et al.*, 2006) for more detailed explanations on their and alternative blossom separation procedures.

7 Computational Results

In order to empirically test the proposed UCPPZ model and cutting-plane algorithm, we have randomly generated a set of 160 instances. All instances are defined on graphs $G = (V, E)$ in which the node set V is located on a rectangular grid, which mimics street networks. Each inner node of the grid is connected to at least four neighboring nodes. Some additional 'diagonal' connections can exist. A typical instance is depicted in Fig. 1. Different instances of the same size are constructed by randomly choosing the type of each edge $e \in E^k$, $k \in \{0, 1, 2, 3\}$, varying the costs and selecting different diagonal connections. The setup guarantees for the UCPPZ instances that the required edges form a connected subgraph.

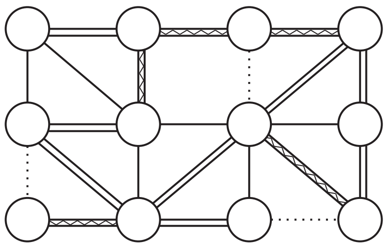
Fig. 1. Example of a UCPPZ Instance on a 4×3 Grid; Edges $e \in E^0$ are depicted dotted, $e \in E^1 \cup E^2$ with single/double Lines, $e \in E^3$ with Zigzag Lines

Instances of the URPPZ are generated by merging several UCPPZ instances together using additional edges $e \in E^0$ such that each UCPPZ instance forms a single connected component. The 450 URPPZ instances are grouped according to size and number of connected components and each group consists of 10 instances. All instances are online available at `www.dpor.rwth-aachen.de/uppz-instances`.

7.1 Solution of the UCPPZ

The UCPPZ can be solved either as a T-join problem requiring the solution of shortest-path and minimum weight perfect matching problems, or by using model (3) and a cutting-plane algorithm using one of the separation procedures of Sections 6. For the first approach we implemented the Dijkstra algorithm with Fibonacci heaps (we used the C++ implementation by Dietmar Kuehl, `http://www.dietmar-kuehl.de/cxxrt/heaps.tar.gz`, yielding an $\mathcal{O}(n(m + n \log n))$ all-pairs shortest path algorithm). The minimum weight perfect matching problems are solved with an $\mathcal{O}(n^3)$ implementation coded in C (by Edward Rothberg `http://elib.zib.de/pub/Packages/mathprog/matching/weighted/`). We refer to the direct T-join approach as (spp+match) and to the cutting-plane algorithm as (lp+cut). Our cutting-plane and the branch-and-cut approach for the UCPPZ and URPPZ uses ILOG/CPLEX (version 9.1 and the *concert* library) (CPLEX, 2005) for LP solution and branch-and-bound. All problem-specific algorithms have been coded in C and C++, compiled in release mode with MS-Visual C++ .NET 2003 version 7.1; all runs were performed on a standard PC (Intel x86 family 15 model 2) with 2.8 GHz, 1GB main memory, on MS-Win 2000.

The ability to solve large-scale UCPPZ instances with the cutting-plane approach mostly depends on the efficiency of the separation procedures. Our setup uses the $\mathcal{O}(|V|^2|E| \log(|V|^2/|E|))$ exact separation procedure of Letchford *et al.* (2004), adapted to the UCPPZ. In order to accelerate the separation procedure from an average-case point of view, connected components C of the support graph are computed first. For each component C and its cut tree, the cut sets $S \subset C$ as well as the component C are tested as handles of a violated blossom inequality. Note that for an optimal selection of edges $F \subseteq \delta(S) \cap E^3$ one must not only consider edges inside C but also the edges of $\delta(C)$. The computation of cut trees takes substantially less time if it is performed for each component separately contrary to computing cut trees in the entire support graph. This can be explained by the observation that the support graph typically decomposes into many components with only very few nodes in each component.

It is worth mentioning that we also tested *heuristic* separation procedures based on ideas proposed by Grötschel and Holland (1985) (i.e., the consideration of potential handles S that are components of the graph induced by edges with flow of at least ε, for a parameter $\varepsilon \in [0, 1)$). However, these heuristics did not consistently accelerate the cutting-plane approach. The reason for their failure is that the exact procedure—applied to components of the support graph—is already very fast and a significant part of the computing time is spent on solving the LP. Heuristic separation procedures tend to cause more LP iterations, so that the overall time for these multiple runs exceeds the running times of the exact separation procedures, even if single heuristic runs are faster.

The results of our computational test for the UCPPZ are summarized in the Tables 1 and 2. The first table shows the characteristics of the test instances and a comparison of the running times of (spp+match) and (lp+cut). The sizes of the instances are given in the first two columns. Since diagonal connections are generated randomly with a fixed probability of between 0 and 1, the number of edges can vary between (approximately) $2|V|(|V|-1)$ and $3|V|^2$. The second column indicates the minimum and maximum number of edges over 10 instances per group. The third column refers to the percentage of required edges, i.e., $|E \setminus E^0|/|E|$. Since the running times t_1 of (spp+match) and t_2 of (lp+cut) vary substantially within the groups, we report the minimum, the average, and the maximum running times in the columns four and six. Moreover, we show the factor t_2/t_1 to indicate how much the cutting plane approach is slower compared to the T-join solution approach.

We interpret the results in the following way: As could be expected, the direct (spp+match) approach is superior to the cutting-plane approach (lp+cut) w.r.t. the *absolute* running time: UCPPZ instances with up to 1,000 edges can be solved by (spp+match) in less than a second, while the cutting plane approach needs up to 1 minute. All larger instances with up to 10,000 edges can be solved by (spp+match) in less than 5 minutes. Here, the cutting-plane approach failed to solve three of the largest instances within the limit of 2 hours computing time. Interestingly, the comparison of the *relative* running times using the factor t_2/t_1 shows no clear trend when considering instances of increasing size. The minimum values of the factor t_2/t_1 indicate that there is always at least one larger-sized instance for which the cutting-plane and (spp+match) approach require nearly the same amount of time.

For the (spp+match) approach, the percentage of the running time spent on solv-

$\|V\|$	$\|E\|$	$\frac{\|E \setminus E^0\|}{\|E\|}$	Time t_1 [s]	%Time spp	Time t_2 [s]	Factor
		in %	(spp+match)	of t_1	(lp+cut)	t_2/t_1
	min/max	min/max	min/avg/max	min/max	min/avg/max	min/avg/max
10×10	199/255	76/84	0.008/0.009/0.011	72/84	0.06/0.11/0.14	6.7/12.4/17.9
12×12	297/378	77/83	0.017/0.023/0.03	59/75	0.09/0.32/0.83	4.2/14.4/38
14×14	371/527	77/84	0.036/0.042/0.047	61/76	0.19/0.38/1.05	4/9.1/24
16×16	525/683	78/82	0.072/0.083/0.092	58/67	0.3/1.33/3.31	4/16/40.8
18×18	696/897	79/82	0.125/0.142/0.158	56/61	1.34/2.55/4.91	9.2/17.8/32.1
20×20	784/1100	78/82	0.2/0.3/0.4	40/54	1.2/2.8/4.3	4.8/10.1/16.7
22×22	932/1270	78/82	0.5/0.5/0.6	34/40	2/9.9/58.1	3.9/19.2/112.7
24×24	1234/1621	79/82	0.7/0.8/1.3	32/43	4.7/12.8/23.9	4.7/16/29.4
26×26	1319/1890	78/81	1.3/3.3/10.4	28/89	3.5/13/47	0.9/5.7/12.2
28×28	1594/2191	78/81	2.5/4.6/14.7	20/88	4.3/21.6/52.6	0.9/6.1/12.4
30×30	1952/2531	79/81	3.3/7.8/24.5	19/85	8.9/82/418	1.1/18.5/125.5
35×35	2395/3328	78/81	9.2/12.6/28.7	13/67	50/316.7/2350.6	2.7/26.9/198.5
40×40	3469/4436	78/81	24/29.8/41.1	7/20	119/217.6/591	4/7.5/20.8
45×45	4277/5621	78/81	62/72.1/86.1	6/14	78.2/559.3/1478.7	1.2/7.5/19
50×50	4994/6724	77/81	141.3/160.5/183.7	4/7	216.3/1234.8/5120.9	1.3/8.1/36.2
60×60	7272/10278	78/81	633.2/703/805	2/6	732.5/3691.3/TL^+	1.1/5.2/10.9

Table 1
Randomly generated UCPPZ Instance and Comparison of (spp+match) and (lp+cut) Solution Approaches; Each Group consists of 10 Instances; $^+$ Failed to solve 3 Instances within the Time Limit TL =7,200s

ing shortest path problems (column *%Time spp of* t_1) decreases with the size of the instances. The main part of the workload is the solution of the matching problem. Probably, a better implementation for solving the minimum weight perfect matching problems (we did not have a $\mathcal{O}(n(m+n\log n))$ implementation at hand) could lead to a better balance between the two algorithmic components and a even faster (spp+match) solution procedure.

Details of the behavior of the cutting-plane algorithm (lp+cut) are presented in Table 2. The three columns *#Calls sep*, *#Cuts*, and *#LP iter* show the number of calls of the separation procedure, the number of cuts separated, and the number of simplex iterations respectively. Again, we report the minimum, the average, and the maximum. The two last columns give the time (in seconds) for solving the UCPPZ to optimality (column *Time*) and the percentage of the time spent on separation (*%Time sep*).

There are significant differences in the computation times of the randomly generated instances: The longest computation time for a 35×35-node instance was more than 2000 seconds, while all 40×40-node and 45×45-node instances were solved faster. The values *#Calls sep*, *#Cuts*, and *#LP iter* can differ by more than factor 10 within one group (but there is no significant correlation between $|E|$ and these numbers). In all cases, the separation procedure worked efficiently, because, on average, more than 5 cuts are generated per second; for small instances the ratio is significantly better. Moreover, for many instances the part of the overall running time spent on the separation routine was below 80%. Typically, cutting-plane algorithms spend more time on separation (as a rule of thumb, more than 90% of the time). This is, therefore, another indicator that the separation algorithm is

$\|V\|$	$\|E\|$	#Calls sep	#Cuts	#LP iter	%Time sep	Time [s]
	min/max	min/avg/max	min/avg/max	min/avg/max	min/max	min/avg/max
10×10	199/255	6/12.4/18	44/57/69	86/104/117	11/75	0.06/0.11/0.14
12×12	297/378	5/20/46	66/103/201	136/175/275	50/85	0.09/0.32/0.83
14×14	371/527	8/14.8/35	88/128/247	189/257/358	55/76	0.19/0.38/1.05
16×16	525/683	8/31.1/65	116/204/313	285/414/622	54/76	0.3/1.33/3.31
18×18	696/897	17/37.5/56	204/298/602	482/589/950	72/81	1.34/2.55/4.91
20×20	784/1100	16/29.8/40	203/284/406	498/628/786	74/83	1.2/2.8/4.3
22×22	932/1270	21/48.5/170	244/479/1668	534/1155/4770	74/87	2/9.9/58.1
24×24	1234/1621	23/56.1/85	380/538/886	890/1096/1495	84/91	4.7/12.8/23.9
26×26	1319/1890	16/49.2/156	402/618/1247	847/1345/3339	83/93	3.5/13/47
28×28	1594/2191	15/53.3/94	437/704/1212	1084/1434/2222	85/92	4.3/21.6/52.6
30×30	1952/2531	21/80.3/260	525/1276/4575	1276/3835/15289	69/93	8.9/82/418
35×35	2395/3328	57/126.1/460	944/2166/8260	2231/8292/58317	41/95	50/316.7/2350.6
40×40	3469/4436	53/104.8/200	1218/1869/4225	2781/3864/8618	89/96	119/217.6/591
45×45	4277/5621	32/148.1/264	1327/3932/8439	3161/9842/24995	78/96	78.2/708.9/1662.3
50×50	4994/6724	60/164.7/369	1628/5381/14599	3669/12825/54052	80/97	216.3/1234.8/5120.9
60×60	7272/10278	56/194.3/363	2492/9207/21258	6260/34418/102111	65/97	732.5/3691.3/TL^+

Table 2
Details of the UCPPZ Cutting-Plane Algorithm; Each Group consists of 10 Instances; $^+$ Failed to solve 3 Instances within the Time Limit TL =7,200s

sufficiently fast and works efficiently.

Concerning integrality, our computational test did not find any UCPPZ instance for which the cutting-plane procedure ended with a fractional solution. Hence, branching was never necessary. Generalized blossom inequalities were always sufficient to produce integer solution to the UCPPZ. Based on this empirical observation, we conjecture that $P^{LP} = P^I$ holds, i.e., the polyhedron P^{LP} of the LP-relaxation of model (3) may be integral. It was beyond the scope of this paper to undertake a detailed polyhedral analysis (results on the dimension of the polyhedron P^I, its facets etc.). Probably, some extreme point preserving transformation from P^I to some matching polyhedron could yield the desired integrality result.

7.2 Solution of the URPPZ

For the URPPZ the T-join approach is not applicable and we have to rely on a branch-and-cut algorithm. Computational results are given in Tables 3 and 4. Column *#Comp* shows the number of connected components, which is the main indicator for the difficulty of an instance. The next column *#Opt/#Int* shows how many instances of a group are solved to optimality and how often integer feasible solutions were found within the time limit of TL =1,800s (no entry means that all 10 instances are solved). The integrality gap *%Gap* is defined as $(z^* - lb)/lb \cdot 100\%$, where lb is the lower bound at the root node of the branch-and-bound tree and z^* the cost of an optimal solution (min/avg/max are taken w.r.t. instances solved to optimality). The column *#BaB Nodes* gives the number of branch-and-bound nodes explored by the branch-and-cut algorithm. Here, 0 means that an instance was solved to optimality solely by applying generalized blossom cuts. If the cutting-plane procedure yields a fractional solution, CPLEX first tries to round variables in order to find a feasible integer solution. Since all coefficients of our test instances are integer, this rounding heuristic sometimes finds optimal solutions and can prove their optimality if the integrality gap is smaller than 1.0. Thus, *#BaB Nodes* is 1 in order to indicate that the solution is computed by the rounding heuristic. Values greater than 1 indicate that branching was performed. Finally, column *Time* gives the overall running time in seconds.

The branch-and-cut algorithm can consistently solve large-scale URPPZ instances with up to 10 components and about 2,000 edges. The smallest instances that could not be solved within 1,800s have 16×16 nodes and 49 and 64 components respectively. The branch-and-cut algorithm failed to inspect all branch-and-bound nodes but integer solutions were always found. Another small 18×18-node instance with 828 edges and 16 components could not be solved, because the cutting-plane algorithm was not able to solve the root node. Here, already 8,551 generalized blossom inequalities and 5 connectivity constraints were separated (in addition to the 16 a priori added connectivity constraints). Nevertheless, the majority ($>$90%) of the instances with up to 1,000 edges could be solved to optimality and for more than 97% an integer solution was computed (with a remaining gap of about 0.7% on average and less than 2.5% in the worst case).

In general, if the number *#Comp* of components increases, instances become more difficult to solve. For instances of the same size (number of node), it means that integrality gaps, numbers of branch-and-bound nodes, and computing times grow rapidly, while less instances can be solved within the given time limit. If we compare instances with an identical number of components but with increasing size, we can observe a property one would not expect: The larger the instances, the smaller the integrality gaps and, therefore, the lower the number of branch-and-bound nodes. One can interpret this behavior as follows: Solving an URPPZ instance consists of two interdependent subproblems, i.e, making the resulting graph even (connecting odd nodes) and connecting the components spanned by required edges. Some of the inter-component edges may at the same time be favorable to eliminate odd degrees and to connect components. Hence, if components consist of more nodes, there is a better chance that such favorable edges exist. The result is that by solving the first subproblem, already more components get connected and, therefore, the second subproblem becomes easier to solve.

Finally, there is still room for improvement: Several classes of URPP-specific valid inequalities are known, e.g., inequalities from the graphical TSP and K-C-inequalities, see (Corberán and Sanchis, 1998; Eglese and Letchford, 2000). These additional cutting planes and corresponding (heuristic) separation procedures might help to further reduce the integrality gap and, thus, allow larger problem instances to be solved.

8 Conclusions

Up to now, undirected and directed Chinese postman problems are the only postman problems belonging to the complexity class $\mathcal{P}$. With the results of Section 2 we have added the UCPPZ to the complexity class $\mathcal{P}$. Any (minor) extension of these problems studied thus far makes the resulting problems hard to solve, i.e., belong to $\mathcal{NP}$. Well-know examples are rural, mixed, and windy postman problems. For these, branch-and-cut is one of the most promising solution approaches, requiring a mixed-integer linear programming model for the problem. The paper has introduced a basic formulation for the UCPPZ in which generalized blossom inequalities constitute the heart of the model. The new model is efficiently solvable with a cutting-plane algorithm. This results from the presented adaptation of the currently fastest separation procedure of Letchford *et al.* (2004) for blossom inequalities.

The proposed model for the UCPPZ is easily extendible to more general postman problems, which has been exemplified for the URPPZ. The computational tests clearly indicate that large-scale instances of the URPPZ with a few thousand edges can be solved with branch-and-cut in reasonable time.

Acknowledgement

I would like to thank Adam Letchford (Department of Management Science at Lancaster University, England) for suggesting T-join-based approaches and for comments on an earlier version of the paper.

References

Christofides, N., Campos, V., Corberán, A., and Mota, E. (1981). An algorithm for the rural postman problem. Technical Report Report IC.O.R.81.5, Imperial College, London.

Corberán, A. and Sanchis, J. (1994). A polyhedral approach to the rural postman problem. *European Journal of Operational Research*, **79**, 95–114.

Corberán, A. and Sanchis, J. (1998). The general routing problem polyhedron: Facets from the RPP and GTSP polyhedra. *European Journal of Operational Research*, **108**, 538–550.

CPLEX (2005). *ILOG CPLEX and Concert C++, API 9.1, Reference Manual.* ILOG, France.

Edmonds, J. (1965). The Chinese postman's problem. *Bulletin of the Operations Research Society of America*, **13**, B–73.

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

Eglese, R. and Letchford, A. (2000). Polyhedral theory for arc routing problems. In M. Dror, editor, *Arc Routing: Theory, Solutions, and Applications*, chapter 6, pages 199–230. Kluwer, Boston.

Eiselt, H., Gendreau, M., and Laporte, G. (1995). Arc routing problems, Part II: The rural postman problem. *Operations Research*, **43**(3), 399–414.

Fredman, M. and Tarjan, R. (1984). Fibonacci heaps and their uses in improved network optimization algorithms. In *25th Annual Symposium on Foundations of Computer Science* (25th FOCS, Singer Island, Florida, 1984), pages 338–346, New York. IEEE.

Gabow, H. (1990). Data structures for weighted matching and nearest common ancestors with linking. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms* (San Francisco, California, 1990), pages 434–443, Philadelphia, Pennsylvania. SIAM.

Ghiani, G. and Laporte, G. (2000). A branch-and-cut algorithm for the undirected rural postman problem. *Mathematical Programming*, **87**(3), 467–482.

Goldberg, A. and Tarjan, R. (1986). A new approach to the maximum flow problem. In *Proceedings of the Eight Annual ACM Symposium on the Theory of Computing (Berkley, CA, 1986)*.

Gomory, R. and Hu, T. (1961). Multi-terminal network flows. *SIAM Journal on Applied Mathematics*, **9**, 551–570.

Grötschel, M. and Holland, O. (1985). Solving matching problems with linear programming. *Mathematical Programming*, **33**, 243–259.

Grötschel, M. and Holland, O. (1987). A cutting plane algorithm for minimum perfect 2-matching. *Computing*, **39**(4), 327–344.

Grötschel, M., Lovasz, L., and Schrijver, A. (1981). The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, **1**(2), 169–197.

Irnich, S. (2005). A note on postman problems with zigzag service. *INFOR*, **43**(1), 33–39.

Letchford, A., Reinelt, G., and Theis, D. (2004). A faster exact separation algorithm for blossom inequalities. In G. Nemhauser and D. Bienstock, editors, *Integer Programming and Combinatorial Optimization*, volume 3064, chapter 10. Springer, Berlin.

Letchford, A., Reinelt, G., and Theis, D. (2006). Odd minimum cut sets and b-matchings revisited. Technical report, Department of Management Science, Lancaster University, Lancaster, England.

Padberg, M. and Rao, M. (1982). Odd minimum cut-sets and b-matchings. *Mathematics of Operations Research*, **7**, 67–80.

Padberg, M. and Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, **33**(1), 60–100.

Schrijver, A. (2003). *Combinatorial Optimization. Polyhedra and Efficiency.* Number 24 in Algorithms and Combinatorics. Springer, Berlin Heidelberg New York.

$\|V\|$	$\|E\|$	#Comp	#Opt/#Int	%Gap	#BaB Nodes	Time [s]
	min/max			min/avg/max	min/avg/max	min/avg/max
10×10	184/246	4		0/0.01/0.09	0/0.3/2	0.1/0.2/0.6
	184/257	9		0/0.03/0.33	0/1.3/12	0.1/0.3/0.7
	187/256	16		0/0.17/0.5	0/6.9/25	0.1/1.3/4.1
	205/257	25		0.06/0.49/1.09	1/144.9/958	0.7/41.1/314
12×12	267/369	4		0/0.01/0.06	0/0.5/4	0.1/0.4/1
	266/378	9		0/0.03/0.23	0/1.6/11	0.1/0.5/1.8
	274/358	16		0/0.1/0.42	0/7.7/35	0.1/9.9/58.2
	285/376	25		0/0.18/0.45	0/28.5/120	0.8/18.8/81.8
	272/343	36		0.09/0.31/0.79	3/72.2/475	0.4/38.6/207.6
14×14	390/507	4		0/0/0	0/0/0	0.2/1.4/10.3
	384/531	9		0/0.01/0.05	0/1.1/5	0.2/2.4/15.6
	388/523	16		0/0.03/0.11	0/5.6/47	0.4/3/13.6
	365/502	25		0.03/0.13/0.26	2/19.7/61	1.7/17.9/73.7
	366/524	36		0/0.19/0.39	0/63.4/262	1.7/74.8/361.4
	395/529	49		0.17/0.34/0.58	12/195.4/677	4.7/459.3/1720.6
16×16	493/687	4		0/0/0.01	0/0.2/2	0.6/4.9/38.6
	485/703	9		0/0/0.02	0/0.6/3	0.4/12.1/105.4
	497/678	16		0/0.03/0.1	0/2.2/7	0.8/11.9/36.6
	496/705	25		0/0.07/0.27	0/15.5/113	0.5/26.4/146.3
	482/700	36		0/0.16/0.29	0/78.4/264	0.5/386.4/1390.2
	546/685	49	8/10	0.05/0.23/0.41	16/167.2/480	95.2/753.5/TL
	484/688	64	3/10	0.25/0.27/0.3	36/216.1/612	499.3/1562.4/TL
18×18	698/900	4		0/0/0	0/0/0	0.8/2/3.8
	612/894	9		0/0/0.01	0/0.3/2	0.8/9/56
	628/879	16	9/9	0/0.01/0.03	0/0.8/4	0.8/186.8/TL
	638/885	25		0/0.03/0.07	0/4.3/12	2.1/24.6/109.6
	621/854	36	9/10	0/0.03/0.09	0/22.1/129	1.4/203.9/TL
	652/883	49	8/9	0/0.07/0.18	0/38.3/92	1.3/444.4/TL
	626/901	64	6/10	0.03/0.12/0.19	1/50.1/173	5.5/881.5/TL
	669/830	81	3/6	0.04/0.14/0.2	1/72.1/187	335.6/1508.2/TL
20×20	761/1111	4		0/0/0.03	0/1/10	1.2/3.9/13.8
	781/1025	9		0/0/0.02	0/0.3/3	0.9/5.5/17.4
	760/1118	16		0/0.01/0.06	0/5/49	1.6/33.2/270.5
	796/1017	25		0/0.01/0.04	0/3.1/16	2.4/61.6/288.7
	808/1023	36	9/9	0/0.05/0.13	0/11.3/53	1.9/379.3/TL
	768/1041	49	8/9	0/0.03/0.06	0/6.2/28	4.2/463.3/TL
	831/1105	64	6/10	0.04/0.09/0.18	10/143.3/410	62.1/1177/TL
	775/1114	81	2/4	0.11/0.13/0.15	6/38.2/137	29.4/1563.5/TL
22×22	934/1362	4		0/0/0	0/0/0	2.4/5.8/11.3
	936/1365	9		0/0/0	0/0.1/1	1.2/8.4/26.9
	978/1358	16		0/0/0.03	0/2.2/18	2.2/41.6/327.8
	950/1263	25		0/0.01/0.03	0/2.5/14	1.4/36.7/77.4
	1045/1357	36	9/9	0/0.01/0.06	0/2.8/19	2.5/211.7/TL
	1091/1350	49	7/7	0/0.03/0.09	0/13.1/41	3/827/TL
	933/1295	64	4/7	0.01/0.08/0.18	1/41.2/95	3.3/1312.2/TL
24×24	1139/1625	4		0/0/0	0/0/0	3.5/11.5/22.1
	1124/1612	9		0/0/0	0/0.2/2	1.9/13.8/35.5
	1161/1624	16		0/0/0.01	0/0.3/2	4.1/10.1/23.4
	1120/1624	25	8/8	0/0.01/0.02	0/4.1/22	1.6/388.1/TL
	1130/1527	36	9/10	0/0.02/0.04	0/18.8/112	4.4/379.2/TL
	1144/1567	49	6/6	0/0.01/0.04	0/11.3/50	5.9/875.3/TL
	1151/1567	64	3/4	0/0.05/0.1	0/24.6/130	6.4/1337.8/TL

Table 3
Details of the URPPZ Branch-and-Cut Algorithm for Smaller Instances; Each Group consists of 10 Instances; Time limit TL =1,800s

$\|V\|$	$\|E\|$	#Comp	#Opt/#Int	%Gap	#BaB Nodes	Time [s]
	min/max			min/avg/max	min/avg/max	min/avg/max
24×24	1139/1625	4		0/0/0	0/0/0	3.5/11.5/22.1
	1124/1612	9		0/0/0	0/0.2/2	1.9/13.8/35.5
	1161/1624	16		0/0/0.01	0/0.3/2	4.1/10.1/23.4
	1120/1624	25	8/8	0/0.01/0.02	0/4.1/22	1.6/388.1/TL
	1130/1527	36	9/10	0/0.02/0.04	0/18.8/112	4.4/379.2/TL
	1144/1567	49	6/6	0/0.01/0.04	0/11.3/50	5.9/875.3/TL
	1151/1567	64	3/4	0/0.05/0.1	0/24.6/130	6.4/1337.8/TL
26×26	1354/1759	4		0/0/0.01	0/0.1/1	4.5/53.3/257.8
	1332/1898	9		0/0/0	0/0/0	4/159.7/1260.2
	1323/1788	16	9/9	0/0/0	0/0.3/2	3.6/249.1/TL
	1328/1890	25		0/0/0.03	0/0.4/4	4.2/21.1/117.7
	1303/1904	36	7/7	0/0.01/0.04	0/4.5/17	13.3/677.7/TL
	1390/1918	49	8/8	0/0.02/0.04	0/7.3/35	33.2/568.3/TL
28×28	1623/2213	4		0/0/0	0/0/0	8.2/18.2/43.1
	1568/2122	9		0/0/0	0/0/0	6.1/29.2/97.9
	1538/2201	16	9/9	0/0/0.01	0/0.3/3	10.3/236.5/TL
	1670/2225	25		0/0/0.02	0/0.9/4	7.4/112/561.4
	1516/2139	36	8/8	0/0.01/0.01	0/1.5/3	40/551.5/TL
30×30	1782/2439	4		0/0/0	0/0/0	10.5/189.5/1479.8
	1775/2556	9	9/9	0/0/0	0/0/0	17/214.5/TL
	1741/2563	16	9/9	0/0/0	0/0.1/1	9.7/249.8/TL
	1786/2577	25	9/9	0/0/0.02	0/1.3/9	30.4/265/TL
	1760/2544	36	8/8	0/0/0.02	0/3.4/19	20.1/615.2/TL
35×35	2574/3507	4		0/0/0	0/0/0	30.2/210/774.7
	2467/3460	9		0/0/0	0/0/0	22.8/131.3/573.6
	2476/3421	16	9/9	0/0/0	0/0.1/1	25.9/262.3/TL
	2867/3519	25	9/9	0/0/0	0/0.2/1	55.3/347/TL
	2446/3509	36	6/6	0/0/0.01	0/4.2/31	75.1/1047/TL
40×40	3282/4027	4		0/0/0	0/0/0	100.2/280.4/727.9
	3125/4361	9		0/0/0	0/0/0	50.8/204.6/638.1
	3453/4401	16		0/0/0	0/0/0	52.6/226.8/868.9
	3142/4568	25	9/9	0/0/0	0/0.1/1	47.9/505.9/TL
45×45	4168/5466	4	9/9	0/0/0	0/0/0	73.8/579.9/TL
	4483/5861	9	8/8	0/0/0	0/0/0	162.5/710/TL
	3988/5751	16	6/6	0/0/0	0/0.5/4	153.6/956.1/TL
	4061/5733	25	8/8	0/0/0	0/0/0	135.2/714.1/TL
50×50	5258/6438	4	7/7	0/0/0	0/0/0	222.5/1062.1/TL
	4938/7135	9	8/8	0/0/0	0/0/0	148.4/924.1/TL
	5332/7113	16	7/7	0/0/0	0/0/0	303.9/891.9/TL
	5076/6903	25	8/8	0/0/0	0/0/0	268.3/1054.3/TL
60×60	8110/10367	4	6/6	0/0/0	0/0/0	667.8/1506.2/TL
	7176/10425	9	4/4	0/0/0	0/0/0	938.1/1617.1/TL
	7517/10216	16	3/3	0/0/0	0/0/0	947/1653.7/TL
	7255/10541	25	5/5	0/0/0	0/0/0	885.4/1578.9/TL

Table 4
Details of the URPPZ Branch-and-Cut Algorithm for Larger Instances; Each Group consists of 10 Instances; Time limit TL =1,800s