Solving the Optimum Communication Spanning Tree Problem

Carlos Armando Zetina[a], Ivan Contreras[a,*], Elena Fernández[b,c], Carlos Luna-Mota[a,b]

[a] *Concordia University and Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, Canada H3G 1M8*
[b] *Department of Statistics and Operations Research, Universitat Politècnica de Catalunya - Barcelona Tech, Barcelona, Spain*
[c] *Barcelona Graduate School of Mathematics (BGSMath), Barcelona, Spain*

Abstract

This paper presents an exact algorithm based on Benders decomposition to solve the optimum communication spanning tree problem. The algorithm integrates within a branch-and-cut framework a strong Benders reformulation, combinatorial lower bounds, in-tree heuristics, fast separation algorithms, and a tailored branching rule. Computational experiments show solution time savings of up to two orders of magnitude compared to state-of-the-art exact algorithms. In addition, our algorithm is able to prove optimality for seven unsolved instances in the literature and three of a new set of larger instances.

Keywords: Networks, Spanning trees, Benders decomposition

1. Introduction

From public transportation to the wireless connectivity of our smartphones, the use of networks has become ubiquitous in our daily routine. Their proper management and design have become essential for maintaining our standard of living. Network optimization models are able to capture the system-wide interactions of decisions inherent to these. They have thus become an important tool for designing and managing networks (Ahuja et al., 1993).

While the most generic network design problem seeks the ideal trade-off between initial investment and operational costs, there are often other efficiency criteria such as network topology, capacity, and robustness that need to be considered. Some examples are the design of minimum spanning trees

*Corresponding author.
Email addresses: c_zetina@encs.concordia.ca (Carlos Armando Zetina), icontrer@encs.concordia.ca (Ivan Contreras), e.fernandez@upc.edu (Elena Fernández), carlos.luna-mota@upc.edu (Carlos Luna-Mota)

(Kruskal, 1956; Prim, 1957), Hamiltonian cycles (Tutte, 1946), survivable networks (Kerivin and Mahjoub, 2005), and networks with connectivity requirements (Magnanti and Raghavan, 2005).

The *optimum communication spanning tree problem* (OCT) is another such example. Introduced by Hu (1974), the OCT seeks to find a spanning tree with minimal operational cost for communicating a set of node-to-node requests R. The use of optimum communication spanning trees arises when communication requests between node pairs are known in advance and the objective is to minimize the communication costs after the network is built. When this is the sole efficiency criterion, the optimal solution is the union of shortest paths between node pairs which, except for particular distance structures, will contain more than $|N| - 1$ links. Optimum communication spanning trees offer a balance between low operational costs on networks using a minimum number of links.

Formally, given an undirected graph $G = (N, E)$ with distances $d_{ij} \geq 0$ associated with each edge $(i, j) \in E$ and requests $r_{ij} \geq 0$ for each node pair $i, j \in R$, the goal is to select a spanning tree T that minimizes

$$\sum_{i,j \in R} r_{ij} C_T(i, j), \tag{1}$$

where $C_T(i, j)$ denotes the length of the unique path in T going from i to j.

The OCT and the *minimum spanning tree problem* (MST) are closely related. Seen as general multicommodity network design problems, they have the same set of feasible solutions with different objective functions. The MST considers only fixed investment costs, while the OCT has only operational transportation costs. This discrepancy marks the difference between a polynomial-time solvable problem, MST, and one that is $\mathcal{NP}$-hard, OCT (Hu, 1974).

The OCT appears as a subproblem in complex hub network design problems in which a tree-star topology is sought (Contreras et al., 2009, 2010b). Thus, efficient solution procedures for the OCT may serve as subroutines for solution procedures for more general network design problems. Outside network optimization, a special case of the OCT where all requirements r_{ij} are equal is used in computational biology to find optimal alignments of genetic sequences (Wu et al., 2000c; Fischetti et al., 2002).

There are two cases presented by Hu (1974) for which the optimal solution of the OCT can be obtained in polynomial running time. The first case is when the distances between all nodes are equal to one. Known as the optimum requirement spanning tree problem, its optimal solution is the Gomory-Hu cut tree defined over the network with edge capacities $b_{ij} = r_{ij}$ for each $i, j \in N$. This can be obtained in polynomial running time by using the algorithm presented in Gomory and Hu (1961) over the network defined by requests R. The second case occurs when the requests between all nodes are one and the distances satisfy a stronger variant of the triangle inequality. Under these assumptions, Hu (1974) shows that an optimal tree is a star and presents a simple algorithm for obtaining it.

With the exception of these two cases, solving the OCT to optimality in reasonable time is still an open problem. In fact, Papadimitriou and Yannakakis (1991) showed that unless $\mathcal{P} = \mathcal{NP}$, no polynomial time approximation scheme exists. Approximation algorithms for special cases of the OCT have been proposed in Peleg and Reshef (1998), Wu et al. (2000a,b,c), Wu (2002), Sharma (2006), and references therein. On the other hand, efficient heuristics have been proposed to obtain high quality solutions in reasonable computational times as in Ahuja and Murty (1987), Palmer (1994); Palmer and Kershenbaum (1995), Soak (2006), Rothlauf and Goldberg (2002); Rothlauf (2007, 2009), and Fischer and Merz (2007).

The first exact algorithm to solve the general OCT was presented by Ahuja and Murty (1987). They proposed a branch-and-bound procedure where lower bounds are obtained using lower planes. Rothlauf (2007) presented a MILP formulation for the general case of the OCT that is able to solve instances with up to 12 nodes. Later, Contreras et al. (2009) presented another integer programming formulation which, enforced with additional valid inequalities, was able to produce optimal solutions for instances with up to 25 nodes in reasonable computational times. Contreras et al. (2010a) proposed a Lagrangian relaxation which produced good lower and upper bounds for instances with up to 50 vertices. Fernández et al. (2013) developed a more compact flow-based formulation with a weaker linear programming (LP) relaxation. Finally, Tilk and Irnich (2016) introduced Dantzig-Wolfe reformulations of arc-based formulations and used column-and-row generation procedures to solve them to obtain optimal solutions for instances with up to 40 nodes.

The main goal of this paper is to contribute to the exact solution of the OCT. In particular, we introduce a Benders reformulation for the OCT based on an arc-based formulation (Contreras et al., 2010a). This reformulation is strengthened with subtour elimination constraints that avoid the separation of (weaker) Benders feasibility cuts, and with combinatorial bounds that are tight for some particular cases. We embed our Benders reformulation into a branch-and-cut algorithm and develop several algorithmic features to speedup its convergence. These include efficient separation algorithms for non-dominated optimality cuts, a tailored branching rule for faster exploration of the enumeration tree, and in-tree heuristics to efficiently explore partitions of the solution space. Our algorithm's computational performance improves significantly that of the state-of-the-art exact algorithms for the OCT by proving optimality for seven unsolved instances in the literature as well as new larger instances. Our algorithm allows to expand the limits of solvability for the OCT from 40 to 60-node instances. It is worth mentioning that the arc-based formulation contains about 1.28 million variables and constraints for a 40-node instance whereas for a 60-node instance the number substantially increases to 6.48 million, which is about five times larger.

The remainder of the paper is organized as follows. Section 2 presents the formal definition of the OCT and the formulation from which we obtain our Benders reformulation. Section 3 describes the Benders decomposition we apply to the reformulation. Section 4 contains the algorithmic enhancements to our

Benders decomposition method and is followed by Section 5 which presents the results of our computational experiments and its comparison to the state-of-the-art solution methods. We present our conclusions in Section 6.

2. Problem Definition

The OCT is defined on a connected, undirected graph $G = (N, E)$ where N is the set of nodes and E is the set of edges. In addition, functions $d : E \mapsto \mathbb{R}^+$ and $r : N \times N \mapsto \mathbb{R}^+$ associate each edge with a distance and every node pair with a request quantity, respectively. The OCT seeks to find a spanning tree T with least total communication cost.

We define $C_T(i, j)$ as the length of the unique path in T between a pair of nodes $i, j \in N$. The communication cost of a request from node pair $i, j \in N$ is calculated as $r_{ij}C_T(i, j)$. The total communication cost is the sum over all node pairs. In our study, we assume the distance function is symmetric, i.e. $d_{ij} = d_{ji}$, there exist communication requirements between each node pair, and the underlying network is complete.

Given these assumptions, we can define $W_{ij} = W_{ji} = r_{ij} + r_{ji}$ as the total request quantity between node pair $i, j \in N$ with $i < j$ and $R = \{(i, j) \in N \times N | i < j \text{ and } W_{ij} > 0\}$ as the set of node pairs with strictly positive communication requirements. This reduces the cardinality of the set of requests by a half and leads to the following definition of the OCT

$$\min_{T \in \Omega(G)} \sum_{(i,j) \in R} W_{ij} C_T(i, j),$$

where $\Omega(G)$ is the set of all spanning trees of G.

In this work we consider an arc-based formulation that uses decision variables indicating the arcs used in the paths connecting pairs of nodes. For the OCT, such a formulation provides strong LP relaxation bounds at the expense of using a large number of variables and constraints, which limits the size of the instances that can be solved using a general purpose solver. The flow-based formulation is obtained by aggregating by origins some subsets of variables of the arc-based formulation. As a consequence, its LP relaxation is weaker than that of the arc-based formulation. Even with the addition of valid inequalities to the flow-based formulation as in Fernández et al. (2013), there is no noticeable improvement in its performance. We refer to Luna-Mota (2015) for details on alternative formulations for the OCT.

We define A as the set of directed arcs corresponding to edge set E, i.e. for each $(i, j) \in E$ there are corresponding arcs $(i, j), (j, i) \in A$. The arc-based formulation is comprised of binary variables y_{ij}, for each edge $(i, j) \in E$, that represent whether the edge is part of the spanning tree solution or not, and continuous variables x^r_{ij}, for each arc $(i, j) \in A$ and request $r = (o_r, d_r) \in R$. These continuous variables indicate the portion of communication request $r \in R$ routed on arc $(i, j) \in A$. Using these variables, the arc-based formulation, P_{OCT}, is:

$$\text{minimize} \sum_{r \in R} \sum_{(i,j) \in E} W^r d_{ij} (x^r_{ij} + x^r_{ji}) \tag{2}$$

$$\text{subject to} \sum_{j \in N:(j,i) \in A} x^r_{ji} - \sum_{j \in N:(i,j) \in A} x^r_{ij} = \begin{cases} -1 & \text{if } i = o_r \\ 1 & \text{if } i = d_r \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in N, r \in R \tag{3}$$

$$x^r_{ij} + x^r_{ji} \leq y_{ij} \qquad \forall (i,j) \in E, r \in R \tag{4}$$

$$\sum_{ij \in E} y_{ij} = |N| - 1 \tag{5}$$

$$x^r_{ij} \geq 0 \qquad \forall (i,j) \in A, r \in R \tag{6}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in E. \tag{7}$$

The objective function (2) calculates the total communication cost while flow conservation constraints (3) ensure that all requests are routed. Constraint set (4) ensures that flow is only sent on edges that form part of the spanning tree while (5) enforces that exactly $|N| - 1$ edges form part of the resulting network. Finally (6) and (7) are the non-negativity and binary definitions of x and y, respectively. P_{OCT} contains $\mathcal{O}(n^4)$ variables and $\mathcal{O}(n^4)$ constraints. As we will see in Section 5, solving this formulation with CPLEX is impractical for instances of over 30 nodes.

3. Benders Decomposition for the OCT

Benders decomposition (Benders, 1962) reformulates a mixed integer linear program to one with significantly fewer variables and an exponential number of constraints, which can be separated efficiently via the solution to an LP subproblem, known as the *dual subproblem* (DSP). This can be accomplished by projecting the original formulation into the discrete variable space, resulting in the reformulation known as the Benders *master problem* (MP). As a result, the contribution of the continuous variables in the original formulation is estimated by two sets of constraints known as feasibility and optimality cuts indexed in the sets of the extreme rays and the extreme points of DSP, respectively.

To apply Benders decomposition to the OCT, we begin by fixing the variables $y_{ij} = \bar{y}_{ij}$ of P_{OCT} leading to the following primal subproblem (PSP):

$$\text{minimize} \sum_{r \in R} \sum_{(i,j) \in E} W^r d_{ij} (x^r_{ij} + x^r_{ji})$$

$$\text{subject to} \sum_{j \in N:(j,i) \in A} x^r_{ji} - \sum_{j \in N:(i,j) \in A} x^r_{ij} = \begin{cases} -1 & \text{if } i = o_r \\ 1 & \text{if } i = d_r \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in N, \forall r \in R \tag{8}$$

$$x^r_{ij} + x^r_{ji} \leq \bar{y}_{ij} \qquad \forall (i,j) \in E, r \in R \tag{9}$$

$$x_{ij}^r \geq 0 \qquad \forall (i,j) \in A, r \in R.$$

Note that PSP can be split into $|R|$ independent shortest path problems PSP_r, one for each request. Let λ and μ denote the dual variables of constraints (8) and (9), respectively. From strong duality, each PSP_r can be substituted by its LP dual, denoted as DSP_r, of the form:

$$\begin{aligned}
\text{maximize} \quad & (\lambda_{d_r}^r - \lambda_{o_r}^r) - \sum_{(i,j)\in A} \mu_{ij}^r \bar{y}_{ij} \\
\text{subject to} \quad & \lambda_j^r - \lambda_i^r - \mu_{ij}^r \leq W^r d_{ij} \qquad \forall (i,j) \in E \\
& \lambda_i^r - \lambda_j^r - \mu_{ij}^r \leq W^r d_{ij} \qquad \forall (i,j) \in E \\
& \mu_{ij}^r \geq 0 \qquad \forall (i,j) \in E \\
& \lambda_i^r \in \mathbb{R} \qquad \forall i \in N.
\end{aligned}$$

As previously mentioned, the set of extreme rays of DSP_r, obtained when it is unbounded, indexes the feasibility cuts of MP while the set of extreme points, obtained from the optimal solution of DSP_r, indexes the optimality cuts. The MP is of the following form:

$$\begin{aligned}
\text{minimize} \quad & \sum_{r\in R} z_r \\
\text{subject to} \quad & z_r \geq \lambda_{d_r}^r - \lambda_{o_r}^r - \sum_{(i,j)\in E} \mu_{ij}^r y_{ij} \qquad \forall r \in R, (\lambda,\mu)_r \in \Theta_r \qquad (10) \\
& 0 \geq \bar{\lambda}_{d_r}^r - \bar{\lambda}_{o_r}^r - \sum_{(i,j)\in E} \bar{\mu}_{ij}^r y_{ij} \qquad \forall r \in R, (\bar{\lambda},\bar{\mu})_r \in \Phi_r \qquad (11) \\
& \sum_{(i,j)\in E} y_{ij} = |N| - 1 \\
& y \in \{0,1\}^{|E|},
\end{aligned}$$

where Θ_r and Φ_r represent the set of extreme points and extreme rays of DSP_r, respectively. Constraints (10) are Benders optimality cuts that estimate the communication cost of each request. Constraints (11) on the other hand, ensure that the MP solution contains a path from each origin/destination request.

Note that MP exploits the decomposability of the subproblems by adding one artificial variable for each commodity unlike the classical Benders reformulation which only uses one variable to aggregate this information. This leads to a better approximation of the transportation costs at each iteration which has been empirically shown to improve solution times (Magnanti et al., 1986; Contreras et al., 2011; Zetina et al., 2017).

4. An Exact Algorithm for the OCT

Benders decomposition has been successfully used to solve a variety of problems in network design (Geoffrion and Graves, 1974; Magnanti et al., 1986; Ran-

dazzo and Luna, 2001; Costa, 2005; Botton et al., 2013), scheduling (Muckstadt and Wilson, 1968), facility location (Magnanti and Wong, 1981; de Camargo et al., 2009; Contreras et al., 2011), and transportation (Cordeau et al., 2000, 2001a,b; Papadakos, 2009). Each of these use algorithmic enhancements and exploit the problem specific structure to successfully apply this method. The OCT is no exception and requires several additional enhancements, which are described in this section.

Since not all feasibility and optimality cuts are needed to solve MP, the *standard* Benders decomposition algorithm (Benders, 1962) proposes relaxing these constraints and solving the relaxed integer MP to optimality providing a dual bound. The obtained solution is then substituted into DSP to obtain violated Benders feasibility or optimality cuts and a primal bound. The cuts are then added to MP and solved again. This iterative process is repeated until the gap between the dual and primal bound is within a desired threshold.

One of the major drawbacks of this *standard* algorithm is that an integer MP needs to be solved at each iteration and there is no simple way to reoptimize when adding additional constraints from one iteration to the next. To overcome this difficulty, *modern* implementations of Benders decomposition consider the solution of the Benders reformulation with a branch-and-cut algorithm, in which Benders cuts are separated not only at integer points but also at fractional points at the nodes of a single enumeration tree (see, for instance Fischetti et al., 2017; Zetina et al., 2017; Ortiz-Astorquiza et al., 2017). We use this modern approach to develop an exact algorithm for the OCT.

In addition, we use the following strategies to speedup the convergence of our branch-and-cut algorithm: i) we use subtour elimination constraints and combinatorial bounds to strengthen the formulation, ii) we employ a heuristic at the nodes of the enumeration tree to efficiently explore the solution space, iii) we exploit the structure of the OCT to efficiently generate non-dominated Benders cuts in the primal space, iv) we use a tailored branching rule for faster exploration, and v) we fine-tune the branch-and-cut parameters. The following sections provide details on each of the applied enhancements.

4.1. Feasibility cuts, cutset inequalities, and subtour elimination constraints

Although the extreme rays of DSP needed to define feasibility cuts for the master problem are readily obtained from any general purpose MIP solver, it may be preferable that these be substituted by a better tailored set of constraints that are obtained via efficient combinatorial algorithms. In this light, we analyze two families of constraints that suit this purpose: *cutset inequalities* and *subtour elimination constraints*.

Cutset inequalities are ubiquitous in the network design literature as a tool for enforcing connectivity in a network. Their conceptual simplicity and effectiveness has contributed to their widespread use for many network design problems. These inequalities are of the following form

$$\sum_{(i,j)\in\delta(S)} y_{ij} \geq 1 \tag{12}$$

where $S \subset N$ and $\delta(S) = \{(i,j) \in E | i \in S, j \in N \backslash S\}$. Violated cutsets can be easily found by solving maximum flow/minimum cut problems with efficient combinatorial algorithms such as that of Edmonds and Karp (1972).

In the context of Benders decomposition, it is well-known that cutset inequalities are sufficient to guarantee the feasibility of Benders primal subproblems for both capacitated and uncapacitated multicommodity network design problems (Magnanti et al., 1986; Costa et al., 2009; Ortiz-Astorquiza et al., 2017; Zetina et al., 2017). Given that the OCT is a special case of the uncapacitated multicommodity network design, cutset inequalities can also be used in lieu of Benders feasibility cuts for our problem.

On the other hand, exploiting the fact that solutions to the OCT must be spanning trees, subtour elimination constraints (SECs) are also viable candidates as a substitute for Benders feasibility cuts. SECs used in formulations for spanning trees are known to provide stronger LP relaxations than those that use cutset inequalities (Magnanti and Wolsey, 1995). Interestingly, for integer solutions, SECs are equivalent to cutset inequalities and are thus separated using the same algorithms, however, instead of having the form (12), they are written as

$$\sum_{(i,j) \in E(S)} y_{ij} \leq |S| - 1, \tag{13}$$

where $E(S)$ represents all edges whose end points both lie within $S \subset N$. This equivalence does not hold for fractional solutions where SECs have been shown to be stronger than cutset inequalities.

Given these characteristics and the fact that they can be used to substitute Benders feasibility cuts, we use SECs in our Benders master problem to ensure feasibility of the primal subproblems. Our computational experiments show that using SECs leads to a tighter LP relaxation for MP than that obtained from P_{OCT}.

4.2. Combinatorial lower bounds

In the case of mixed integer programs with a minimization objective, the LP of any formulation provides a valid lower bound on the optimal solution value. In an enumerative framework such as branch-and-bound, the smallest solution value of all explored nodes provides a global lower bound. Global lower bounds are used to estimate the optimality gap during the execution and at the end of any exact algorithm, but may also be useful in a preprocessing step to evaluate a priori the difficulty of a given instance. We next present two global combinatorial lower bounds for the OCT, which were first proposed in Fernández et al. (2012) (see also, Luna-Mota, 2015). To the best of our knowledge, these bounds have never been used (or exploited) in an algorithmic context. They not only give an assessment of the difficulty of the problem, but at times obtain the optimal solution value of the mixed integer program.

The first combinatorial lower bound is valid for complete graphs with Euclidean distances that satisfy the triangle inequality. We observe that for this

case, at most $|N|-1$ requests will be served on their shortest path $D_G(i,j)$, i.e. by a direct link from origin to destination. Therefore, at least $|R|-|N|-1$ requests will have to use a path that is at least as long as the second shortest path $D^2_G(i,j)$. Let z represent the total communication cost of all requests on the spanning tree, therefore we have:

$$z \geq \min_{T\in\Omega(G)}\{\sum_{(i,j)\in T} W^{ij}D_G(i,j) + \sum_{(i,j)\notin T} W^{ij}D^2_G(i,j)\} \tag{14}$$

$$\iff z \geq \sum_{(i,j)\in E} W^{ij}D^2_G(i,j) + MST(d^*), \tag{15}$$

where $d^*_{ij} = W^{ij}(D_G(i,j) - D^2_G(i,j))$ and MST(d^*) represents the optimal solution value of the minimum spanning tree over the network with distance function d^*. Note that MST(d^*) accounts for the $|N|-1$ edges with the biggest difference between $D_G(i,j)$ and $D^2_G(i,j)$ provided they form a spanning tree.

This combinatorial bound is easily obtained in two steps. The first is to calculate the shortest distance for each request $(i,j)\in R$ on the original graph minus the edge joining these directly. The second is to obtain the minimum spanning tree on a topologically equivalent graph with modified distance function d^* using Kruskal or Prim's algorithm.

The second combinatorial lower bound does not require any assumptions on the distance function. It is calculated using the minimum spanning tree of the distance function d, MST(d), and the minimum communication tree with respect to the request R, MCT, to construct the lexicographically minimal sequences of edge lengths and edge flows which when combined appropriately provide a lower bound on the solution of the OCT.

Let $f_1 \leq f_2 \leq ... \leq f_{|N|-1}$ be the flows that traverse the edges of MCT and $d_1 \leq d_2 \leq ... \leq d_{|N|-1}$ be the lengths of the edges of MST(d), both sorted in increasing order. In addition, let $f^*_1 \leq f^*_2 \leq ... \leq f^*_{|N|-1}$ and $d^*_1 \leq d^*_2 \leq ... \leq d^*_{|N|-1}$ be the equivalent edge flows and edge lengths sequences, respectively, of the optimum communication spanning tree. The edge length sequence of MST(d) is the lexicographically minimal length sequence among all spanning trees of G, while the edge flow sequence of MCT is also lexicographically minimal among all equivalent flow sequences of spanning trees of G.

Since, these sequences are lexicographically minimal among all equivalent spanning trees, they are also lexicographically minimal with respect to the optimal OCT, i.e. $f_i \leq f^*_i \ \forall i \in 1..|N|-1$ and $d_i \leq d^*_i \ \forall i \in 1..|N|-1$. Thus, if $S_{|N|-1}$ denotes the set of all permutations of the indices $\{1,2,...,|N|-1\}$ then

$$z \geq \min_{\sigma\in S_{|N|-1}}\{\sum_{i=1}^{|N|-1} d_i f_{\sigma(i)}\} \tag{16}$$

$$\iff z \geq \sum_{i=1}^{|N|-1} d_i f_{|N|-i}, \tag{17}$$

where the equivalence between (16) and (17) comes from the rearrangement inequality (Bulajich Manfrino et al., 2009).

Both combinatorial bounds (15) and (17) can be obtained at low computational cost and provide insights on the difficulty of the underlying instance. For some instances, these combinatorial bounds are tight and play a significant role in the proof of optimality. These instances will be highlighted in Section 5.

4.3. In-tree Heuristics

Obtaining high quality solutions early in the branch-and-bound search often leads to smaller enumeration trees by providing an improved criterion for pruning and a guide for potential variables to branch on. If obtained before beginning the enumeration tree, these solutions can be used for variable elimination tests that reduce the problem size.

For the OCT, we exploit the fact that our search is limited to spanning trees to construct heuristic solutions at all nodes in the enumeration tree. We then use a well known local search heuristic proposed by Ahuja and Murty (1987) to improve this initial candidate solution. Both phases of our heuristic algorithm take into account the variables that have been fixed so far by the branching process of the enumeration tree. This decreases the problem size and leads to a faster computation time.

Let $Y^1(\rho)$ and $Y^0(\rho)$ denote the set of variables y that have been set to 1 and 0, respectively, at node ρ of the enumeration tree. In addition, let $\bar{y}(\rho)$ denote the solution of the LP relaxation at node ρ. We construct an initial solution by finding a maximum spanning tree with weights $\bar{y}(\rho)$ such that all edges in $Y^1(\rho)$ form part of the tree and edges in $Y^0(\rho)$ are not considered. This spanning tree is obtained by a modified version of Prim's algorithm for minimum spanning trees. Upon obtaining this initial candidate, we try to improve it by exploring a 1-edge-exchange neighbourhood. While a naïve algorithm may require $\mathcal{O}(n^5)$ operations to evaluate the elements of the 1-edge-exchange neighborhood of an arbitrary tree, Ahuja and Murty (1987) present an algorithm that exploits the natural structure of the problem to explore this neighbourhood in $\mathcal{O}(n^3)$ operations. For the sake of completeness, we next present a brief overview of the procedure and refer the reader to Ahuja and Murty (1987) for a detailed description.

The local search starts by pre-computing the distance matrix of T, $D_T = [d^T_{ij}]$, and the communication cost of T (z) in $\mathcal{O}(n^2)$ total time. After this global pre-processing step, the algorithm iterates over the $|N|-1$ edges of T applying the following process for each $e \in E$. Let $\delta_T(S)$ denote the cut associated with $T\backslash\{e\}$. For every $i \in N$, the amount of communication that vertex i needs to send from S to $N\backslash S$ or vice-versa is calculated as

$$W_i^{e_T} = \begin{cases} \sum\limits_{j \in N\backslash S} W^{ij} & \text{if } i = S \\ \sum\limits_{j \in S} W^{ij} & \text{if } i = N\backslash S, \end{cases}$$

and the total communication cost is

$$\xi_i^{e_T} = \begin{cases} \sum\limits_{j \in N \setminus S} W^{ij} d_{ij}^T & \text{if } i = S \\ \sum\limits_{j \in S} W^{ij} d_{ij}^T & \text{if } i = N \setminus S. \end{cases}$$

Finally, the total communication requirement that must traverse $\delta_T(S)$ is $W^{e_T} = \sum_{i \in V} W_i^{e_T}$. Having done all these calculations as a preprocessing step, for a given edge $e = (e_1, e_2) \in E$, $\forall i \in S$, $\forall j \in N \setminus S$, and current routing cost z, we can evaluate the communication cost of replacing e for (i, j) in constant time as $z - (\xi_{e1}^{e_T} + W^{e_T} d_e^T + \xi_{e2}^{e_T}) + (\xi_i^{e_T} + W^{e_T} d_{ij}^T + \xi_j^{e_T})$. Therefore, we are able to explore all solutions in the 1-exchange neighbourhood of a given tree T in $\mathcal{O}(n^3)$ time and keep that with the lowest communication cost.

4.4. Pareto-optimal cuts

Modern implementations of Benders decomposition use strengthened variants of Benders optimality cuts. These are obtained either by using combinatorial arguments for lifting coefficients as in Magnanti et al. (1986), in-and-out strategies as in Fischetti et al. (2017), or Pareto-optimal cuts as proposed by Magnanti and Wong (1981) and Papadakos (2008). These modified Benders cuts play a crucial rule in the convergence speed of the standard iterative Benders algorithm and the strength of the LP estimation in the case of our branch-and-cut algorithm.

Magnanti and Wong (1981) define cut dominance as follows. Given two cuts defined by dual solutions u and u' of the form $z \geq f(u) + yg(u)$ and $z \geq f(u') + yg(u')$, respectively, the cut defined by u dominates that defined by u' if and only if $f(u) + yg(u) \geq f(u') + yg(u')$ with strict inequality holding for some feasible y of MP. If a cut defined by u is not dominated by any other optimality cut, then this cut is said to be a Pareto-optimal Benders cut.

For the OCT, we first note that our subproblem is equivalent to that obtained when applying Benders decomposition to the *multicommodity uncapacitated fixed-charge network design problem* (MUFND) where requests are considered as commodities. Zetina et al. (2017) apply Benders decomposition to MUFND using Pareto-optimal cuts obtained by solving *minimum cost flow problems* as in Magnanti et al. (1986). We adopt this strategy for our subproblem to obtain Pareto-optimal cuts. In particular, we solve the following parametric minimum cost flow problem (MCF_r) to obtain Pareto-optimal cuts:

$$\text{minimize} \quad \sum_{(i,j) \in A} c_{ij}^r x_{ij}^r - DSP_r(\bar{y}) x_0 \tag{18}$$

$$\text{subject to} \quad \sum_{j \in N:(j,i) \in A} x_{ji}^r - \sum_{j \in N:(i,j) \in A} x_{ij}^r = \begin{cases} -(1 + x_0) & \text{if } i = o_r \\ 1 + x_0 & \text{if } i = d_r \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in N \tag{19}$$

$$x_{ij}^r + x_{ji}^r \leq y_{ij}^0 + x_0 \bar{y}_{ij} \qquad \forall (i,j) \in E \tag{20}$$

$$x_{ij}^r, x_{ji}^r \geq 0 \qquad \forall (i,j) \in E,$$

where y^0 is a core point as defined by Magnanti and Wong (1981) and $x_0 \in \mathbb{R}$.

The problem can be interpreted as that in which a rebate of $DSP_r(\bar{y})$ is given for each additional unit of the request routed on the network with demand and capacities defined by (19) and (20), respectively (Magnanti et al., 1986). Magnanti et al. (1986) show that any fixed value $x_0 \geq \sum_{(i,j)\in A} y^0_{ij}$ is optimal for MCF_r, therefore leaving only a minimum cost flow problem to be solved for each request $r \in R$. As a result of fixing x_0, it is no longer necessary to solve $DSP_r(\bar{y})$ since it is now multiplied by a constant in MCF_r. This observation allow us to save computational time by solving MCF_r directly as the separation problem rather than solving it as a complementary problem for Pareto-optimal Benders cuts. The corresponding dual variables of (19) and (20) are used to define the Pareto-optimal Benders cuts of the form (10).

4.5. Branching rule

Branching plays an important role in branch-and-bound algorithms. Poor branching choices lead to larger enumeration trees and longer solution times (Mitra, 1973). Three prevailing branching strategies are maximum infeasibility, pseudocost, and strong branching. The latter has been shown to produce significantly smaller enumeration trees at the expense of solving inexactly several LP problems before branching. Pseudocost branching on the other hand uses statistics accumulated while exploring the enumeration tree. Since there is not much information at the beginning, this strategy is vulnerable to making poor choices at the beginning. Maximum infeasibility simply branches on the variable with the most fractional value and is known to be no better than randomly selecting a variable to branch on. Finally, Achterberg et al. (2005) propose a hybrid of strong branching and pseudocost branching that addresses the drawbacks of both approaches. Computational experiments show this is a promising branching strategy for general mixed integer programs.

An important difference between a Benders reformulation and the generic mixed integer programs on which these strategies were tested is that at any point in time during Benders decomposition, there is not a complete description of the underlying problem, i.e. only a partial formulation is available. This poses an issue in particular to the use of strong branching where LP relaxations of the nodes are used to estimate the impact of candidate variables for branching. On the other hand, the reduced size of the LP problems in our branch-and-cut requires less computational time to evaluate the potential impact of branching candidates.

We propose a hybridized strategy that exploits the problem specific structure and uses strong branching as its alternative when the former does not render a clear candidate. Our primary branching rule considers the fact that our solution is a tree and seeks to construct one using the variables currently fixed at value one.

Let $\bar{y}(\rho)$ and $Y^1(\rho)$ be the current fractional master problem solution and the components whose value is one at node ρ, respectively. Note that $\sum_{ij\in Y^1(\rho)} \bar{y}_{ij} < N-1$, otherwise $\bar{y}$ would be an integer solution. Let $S(Y^1(\rho))_i$ denote the nodes

in connected components $i \in I(Y^1(\rho))$ of $Y^1(\rho)$. Our branching rule selects edge $(i,j) \in \delta(S(Y^1(\rho))_i, S(Y^1(\rho))_j)$, $i \neq j \in I(Y^1(\rho))$ that leads to the component with the highest number of nodes. If more than one candidate satisfying this condition exists, we select the one preferred according to strong branching.

4.6. Fine-tuning branch-and-cut parameters

As pointed out in Zetina et al. (2017), implementing Benders decomposition within one enumeration tree, requires additional fine-tuning of branch-and-cut parameters. They show that by properly selecting the core points and cutting strategies, one can save up to one order of magnitude of computing time. For the OCT, we observed three main contributors to the speed of our algorithm.

The first one is core point selection. After testing several alternative strategies for the selection of core points, based on promising and non-promising variables, we observed that the one performing best was that where the selection of core points was based on the current incumbent solution. Our corepoint y^0 is defined as follows:

$$y^0_{ij} = \begin{cases} 1 & \text{if } (i,j) \in T_{best} \\ 0.3 & \text{otherwise,} \end{cases}$$

where T_{best} is the current incumbent solution. It is evident that this core point is always feasible for the primal subproblem since there will exist a path for each origin/destination pair given that it contains T_{best} which is a spanning tree.

The second one is root node preprocessing. We try to leave the root node with a lower bound close to the LP relaxation of the Benders reformulation. Its importance is two-fold. The first is that it serves as an initial global lower bound that will increase as the tree grows and new child nodes are created and explored. As mentioned in Section 4.1, our master problem LP relaxation is tighter than that of the arc-based formulation. To take advantage of this in our algorithm, we add Benders cuts for fractional solutions until the lower bound has improved at least once and at least three consecutive iterations have passed without improvement.

The second important reason for leaving the root node with a solution close to that of the LP relaxation is that the Benders cuts generated prior to obtaining the solution provide an improved description of the master problem polytope. However, to avoid keeping unnecessary information, we filter them and keep only 30% of the Benders cuts, namely those with the least slack at the last LP relaxation solution. These cuts are used to declare the initial Benders master problem in our solver. They are used to infer improved lower bounds, general valid inequalities and bound strengthening.

The last important fine-tuning aspect in our algorithm is the cutting and separation frequency. Despite leaving the root node with a good description of the underlying polytope, there still exists a risk of not properly estimating the LP relaxations of child nodes in the enumeration tree. On the other hand, adding too many cuts, leads to excessively large LPs that must be solved at other nodes. To circumvent this, we control the cutting frequency by only solving the

subproblems at nodes whose depth in the enumeration tree is divisible by five and by only adding violated inequalities as local cuts instead of global cuts. In addition, at each candidate node, the separation procedure is applied at most three times.

Putting together all the above elements, our algorithm works as follows. We first obtain heuristic solutions such as the minimum spanning tree and the optimum requirement spanning tree and add the corresponding Pareto-optimal Benders cuts to warm start the solution process. We then calculate the combinatorial lower bounds to assess the difficulty of the instance. If the desired optimality gap of 0.01% is not yet achieved, we begin the root node preprocessing. We then define the problem with the Benders cuts kept from our preprocessing phase, which defines the initial MP of our branch-and-cut. Then, at each node of the enumeration we solve the corresponding LP relaxation, execute our heuristic, and add Pareto-optimal cuts locally to avoid large underlying LPs.

5. Computational Experiments

We have performed extensive computational experiments to evaluate the efficiency of our proposed algorithmic framework and the effect of the implemented enhancements. Our analyses focus on: the performance of the combinatorial lower bounds, the strength of our Benders MP and the efficiency of our proposed branch-and-cut algorithm compared to the state-of-the-art and a general-purpose solver solving the arc-based formulation.

We use the benchmark Berry, Palmer and Raidl instances from the literature (Palmer, 1994; Palmer and Kershenbaum, 1995; Rothlauf, 2009). These instances consist of complete graphs with requests between each origin/destination pair. We also use the instances of Contreras et al. (2010b) that range from 10 to 50 nodes with requests solicited between approximately fifty percent of origin/destination pairs. Finally, we have generated 20 new instances ranging from 60 to 100 nodes, also with approximately fifty percent of origin/desination pairs.

All algorithms were coded in C using the callable library for CPLEX 12.7.1. The separation and addition of SECs and Benders optimality cuts has been implemented via lazycallbacks and usercutcallbacks. For a fair comparison, all use of CPLEX was limited to one thread and the traditional MIP search strategy. Experiments were executed on an Intel Xeon E5 2687W V3 processor at 3.10 GHz under Linux environment.

5.1. Combinatorial bounds

The combinatorial bounds proposed in Section 4.2 are all calculated in less than a second of CPU time and give an indication of the difficulty of the instances. Given that the Raidl instances do not satisfy the triangle inequality, we focus on the Berry and Palmer instances. Table 1 shows the objective value of the optimum integer solution (Optimum), the LP relaxation optimum of the arc-based formulation, the MCT-MST bound, the second shortest path bound (D^2), and the % deviation of the best combinatorial bound from the integer optimum (% best dev).

Table 1: Combinatorial bound performance

Name	$\|N\|$	Optimum	LP	MCT-MST	D^2	% best dev
Berry	6	534.00	528.00	374.00	488.00	8.61
	35	16,915.00	16,915.00	16,167.00	16,915.00	-
	35u	16,167.00	14,010.67	16,167.00	13,106.00	-
Palmer	6	693,180.00	693,180.00	656,877.00	633,874.00	5.24
	12	3,428,509.00	3,305,964.48	2,824,224.00	2,827,752.00	17.52
	24	1,086,656.00	1,086,656.00	901,510.00	1,086,656.00	-

We note that the bounds are tight for three of the five instances. The second shortest path is the best performing for all except for Berry 35u where MCT-MST not only outperforms the second shortest path, but is also better than the LP relaxation. For the Raidl instances, the MCT-MST has an average deviation from the optimum value of 57.24%, giving testimony to the difficulty of these instances.

5.2. Linear programming relaxations

As mentioned in Section 4.1, the substitution of Benders feasibility cuts for subtour elimination constraints has the additional advantage of providing a tighter LP relaxation than the arc-based formulation. Table 2 shows the LP relaxation of the arc-based formulation (LP 4-index), the LP relaxation at our root node (LP MP), and the LP gap calculated as LP Gap= $(\text{Opt} - \max(\text{LP arc-based}, \text{LP MP}))/\text{Opt}$ where Opt denotes the optimal solution value.

We first note that while the Palmer and Berry instances have a tight LP bound, the same no longer holds for the Raidl set. In fact, we note that as the size of the instances increases, the average LP gap increases with the highest being 9.48% for Raidl 40d. This supports the indication posed in the previous section on the difficulty of these instances. Second, we note that the six bold-faced instances in Table 2 have a stronger LP bound with our Benders MP using subtour elimination constraints than with the arc-based formulation.

5.3. Solution times

This section summarizes the results of the experiments done to assess the computational efficiency of our proposed algorithm. We compare our algorithm's performance with that of the state-of-the-art MIP solver CPLEX 12.7.1 solving the arc-based formulation. We also compare our results with those reported in Tilk and Irnich (2016).

Results are presented in three parts. The first one refers to the results on the Berry, Palmer and Raidl instance classes while the second part summarizes the performance on the instances presented in Contreras et al. (2010a). The final part of our experiment was performed on a new set of larger size instances generated with the code used in Contreras et al. (2010a). The interested reader is referred to that paper for further details on how the instances have been

Table 2: Linear programming relaxations

Name	$\|N\|$	LP arc-based	LP MP	% LP Gap
Berry	6	528.00	528.00	1.12
	35	16,915.00	16,915.00	-
	35u	14,010.67	**14,187.16**	-
Palmer	6	693,180.00	693,180.00	-
	12	3,305,964.48	3,301,299.23	3.57
	24	1,086,656.00	1,086,656.00	-
Raidl	10	53,643.00	53,643.00	0.06
	20	155,006.30	155,006.30	1.63
	50	747,476.27	747,338.44	7.36
	75	1,500,604.65	**1,507,453.31**	n.a.
	100	2,166,764.50	1,839,134.00	n.a.
Contreras	10a	70,954.00	70,954.00	0.28
	10b	38,059.00	38,059.00	-
	10c	29,113.00	29,113.00	-
	10d	38,892.00	38,892.00	0.78
	20a	85,810.50	85,810.50	4.09
	20b	94,935.67	94,932.95	1.45
	20c	98,785.17	**98,900.50**	3.52
	20d	87,154.33	87,154.33	0.34
	30a	222,590.50	222,590.50	2.48
	30b	244,704.00	244,704.00	1.96
	30c	203,723.06	203,722.32	2.55
	30d	213,357.20	213,357.20	2.65
	40a	346,642.67	346,635.06	1.11
	40b	278,745.50	**279,536.91**	4.28
	40c	273,956.38	273,929.71	4.61
	40d	314,754.83	314,754.83	9.48
	50a	438,462.13	**439,016.62**	4.33
	50b	468,673.84	**468,900.11**	7.54
	50c	363,830.01	363,788.29	8.35
	50d	461,613.32	461,554.18	7.53

generated. Unless otherwise stated, all experiments are given a time limit of two hours.

Table 3 shows the optimum solution values, and the times in seconds of our algorithm and CPLEX branch-and-cut for the classic Berry, Raidl, and Palmer instances. The 75 and 100-node Raidl instances timed out for both algorithms having an optimality gap of 10.58% and 45.1% with our algorithm and CPLEX, respectively.

As shown in Table 3, the Raidl 50-node instance was solved to proven optimality in less than one hour of computing time. CPLEX, however, is unable to solve it within the allocated time. To the best of our knowledge, it is the first time that an exact algorithm solves this instance to proven optimality. We also highlight that we are also able to solve the Berry 35u instance to proven optimality. This comes as a result of the global lower bound calculated by our MCT-MST procedure which proved the optimality of our optimum requirement

Table 3: Solution times- Classic Instances

Name	$\|N\|$	Optimum	Benders (s)	CPLEX (s)
Berry	6	534.00	0.01	0.01
	35	16,915.00	0.02	0.60
	35u	16,167.00	0.02	time
Palmer	6	693,180.00	-	0.01
	12	3,428,509.00	0.45	3.64
	24	1,086,656.00	-	0.08
Raidl	10	53,674.00	0.01	0.08
	20	157,570.00	0.23	7.36
	50	806,864.00	3,171.95	time

spanning tree solution. Like with the Raidl 50 instance, we are not aware of any other exact algorithm able to solve to proven optimality the Berry 35u instance.

In Table 4 we present the optimum solution values, and the computing times in seconds of our Benders algorithm, CPLEX and the computing times reported in Tilk and Irnich (2016) for their column-and-row generation method for the extensive arc-based formulation. We realize that a different version of the CPLEX software is used. However, we report them as a benchmark for instances that are currently solvable by ad hoc exact algorithms.

Table 4: Solution times- Contreras instances

		CPU time (s)		
$\|N\|$	Optimum	Benders	CPLEX	Tilk & Irnich
10a	71,156.00	0.02	0.07	0.07
10b	38,059.00	0.01	0.01	0.02
10c	29,113.00	0.01	0.01	0.02
10d	39,197.00	0.01	0.05	0.05
20a	89,474.00	0.24	3.66	4.19
20b	96,333.00	0.15	2.48	2.62
20c	102,505.00	0.23	6.14	3.4
20d	87,452.00	0.11	1.15	0.27
30a	228,247.00	1.89	87.37	101.9
30b	249,607.00	1.67	129.72	162.14
30c	209,062.00	2.00	246.43	111.47
30d	219,170.00	1.32	193.61	106.29
40a	350,542.00	3.33	521.28	115.08
40b	292,047.00	7.18	time	time
40c	287,198.00	6.75	time	time
40d	347,715.00	194.32	time	time
50a	458,881.00	129.24	time	time
50b	507,142.00	605.81	time	time
50c	396,966.00	42.04	time	time
50d	499,184.00	3,221.28	time	time

Our branch-and-cut algorithm outperforms the other two approaches by a significant margin. It is at least ten times and up to 100 times faster than the others. Moreover, for the first time it proves optimality for the seven elusive 40

and 50-node instances of Contreras et al. (2010a). We attribute this speedup to the ability of our algorithm to explore nodes in a significantly shorter time due to the reduced size of the underlying LP problems. In addition, our strengthened Benders cuts provide an accurate estimation of the underlying LP of the nodes which, combined with our branching rules, lead to shorter solution times.

To test the limits of our algorithm, we have generated 20 instances of between 60 and 100 nodes. Table 5 shows the optimum or best-known solution value, the computing time required to calculate it and the percentage optimality gap at the end of a 24 hour time limit. We note that our algorithm is able to solve all 60-node instances and obtains reasonable percentage gaps for the remaining instances.

Table 5: Solution times- new instances

$\|N\|$	Optimum/Best	Time (s)	% gap
60a	558,837.00	1,415.69	-
60b	646,174.00	422.99	-
60c	665,518.00	15,938.51	-
60d	503,685.00	20,388.01	-
70a	692,837.00	time	6.49
70b	849,031.00	time	4.74
70c	829,570.00	time	6.97
70d	631,192.00	time	2.28
80a	825,605.00	time	7.24
80b	951,069.00	time	7.06
80c	883,122.00	time	1.27
80d	855,935.00	time	7.04
90a	1,065,823.00	time	11.02
90b	1,107,918.00	time	8.36
90c	1,107,435.00	time	10.15
90d	1,080,571.00	time	13.04
100a	1,222,478.00	time	7.20
100b	1,564,725.00	time	9.54
100c	1,255,077.00	time	9.92
100d	1,233,536.00	time	10.72

These results showcase the advantages of applying a Benders-based branch-and-cut algorithm for network design problems. Some of the key ingredients that led to these results are: core point selection, an efficient separation algorithm to obtain Pareto-optimal cuts, a customized branching rule, and fine-tuning the branch-and-cut parameters. By considering these, we were able to improve the performance of our algorithm from being able to solve the same instances as CPLEX to the results presented in Tables 3, 4, and 5.

6. Conclusion

In this paper, we have provided a novel exact algorithm for the OCT that is up to 100 times faster and is able to solve larger instances than the state-of-the-art. The proposed algorithm uses a novel Benders reformulation for OCT that is tighter than the best known formulation in the literature and takes advantage of problem specific structure to obtain Pareto-optimal Benders cuts efficiently, guarantee feasibility in fractional solutions, select branching variables and obtain high quality solutions during the enumeration process. Finally, a new testbed of larger instances has been presented to be used for benchmarking in future research. Our results support the use of a Benders-based branch-and-cut for network design emphasizing on the importance of fine-tuning the algorithm's parameters for faster solution times.

Acknowledgments

The research of the first two authors was partially funded by the Canadian Natural Sciences and Engineering Research Council [Grant 418609-2012]. The research of the last two authors was partially funded by the Spanish Ministry of Economy and Competitiveness and ERDF funds [Grant MTM2015-63779-R (MINECO/FEDER)]. This support is gratefully acknowledged.

References

Achterberg, T., Koch, T., Martin, A., 2005. Branching rules revisited. Operations Research Letters 33 (1), 42–54.

Ahuja, R. K., Magnanti, T. L., Orlin, J. B., 1993. Network Flows: Theory, Algorithms, and Applications. Prentice-Hall, New Jersey, U.S.A.

Ahuja, R. K., Murty, V. V. S., 1987. Exact and heuristic algorithms for the optimum communication spanning tree problem. Transportation Science 21 (3), 163–170.

Benders, J.-F., 1962. Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4, 238–252.

Botton, Q., Fortz, B., Gouveia, L., Poss, M., 2013. Benders decomposition for the hop-constrained survivable network design problem. INFORMS Journal on Computing 25 (1), 13–26.

Bulajich Manfrino, R., Gómez Ortega, J. A., Valdez Delgado, R., 2009. Inequalities - A Mathematical Olympiad Approach. Birkhäuser Basel.

Contreras, I., Cordeau, J.-F., Laporte, G., 2011. Benders decomposition for large-scale uncapacitated hub location. Operations Research 59 (6), 1477–1490.

Contreras, I., Fernández, E., Marín, A., 2010a. Lagrangean bounds for the optimum communication spanning tree problem. TOP 18 (1), 140–157.

Contreras, I., Fernández, E., Marín, A., 2009. Tight bounds from a path based formulation for the tree of hub location problem. Computers and Operations Research 36 (12), 3117 – 3127.

Contreras, I., Fernández, E., Marín, A., 2010b. The tree of hubs location problem. European Journal of Operational Research 202 (2), 390 – 400.

Cordeau, J.-F., Soumis, F., Desrosiers, J., 2000. A Benders decomposition approach for the locomotive and car assignment problem. Transportation Science 34 (2), 133–149.

Cordeau, J.-F., Soumis, F., Desrosiers, J., 2001a. Simultaneous assignment of locomotives and cars to passenger trains. Operations research 49 (4), 531–548.

Cordeau, J.-F., Stojković, G., Soumis, F., Desrosiers, J., 2001b. Benders decomposition for simultaneous aircraft routing and crew scheduling. Transportation Science 35 (4), 375–388.

Costa, A. M., 6 2005. A survey on Benders decomposition applied to fixed-charge network design problems. Computers & Operations Research 32 (6), 1429–1450.

Costa, A. M., Cordeau, J.-F., Gendron, B., 2009. Benders, metric and cutset inequalities for multicommodity capacitated network design. Computational Optimization and Applications 42 (3), 371–392.

de Camargo, R. S., de Miranda Jr., G., Luna, H. P. L., 2009. Benders decomposition for hub location problems with economies of scale. Transportation Science 43 (1), 86–97.

Edmonds, J., Karp, R. M., 1972. Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the Association for Computing Machinery 19 (2), 248–264.

Fernández, E., Luna-Mota, C., Hildenbrandt, A., Reinelt, G., Wiesberg, S., 2012. A compact formulation for the optimum communication spanning tree. In: International Symposium on Mathematical Programming.

Fernández, E., Luna-Mota, C., Hildenbrandt, A., Reinelt, G., Wiesberg, S., 2013. A flow formulation for the optimum communication spanning tree. Electronic Notes in Discrete Mathematics 41 (Supplement C), 85 – 92.

Fischer, T., Merz, P., 2007. A memetic algorithm for the optimum communication spanning tree problem. In: Proceedings of the 4th International Conference on Hybrid Metaheuristics. HM'07. Springer-Verlag, Berlin, Heidelberg, pp. 170–184.

Fischetti, M., Lancia, G., Serafini, P., 2002. Exact algorithms for minimum routing cost trees. Networks 39 (3), 161–173.

Fischetti, M., Ljubić, I., Sinnl, M., 2017. Redesigning Benders decomposition for large-scale facility location. Management Science 63 (7), 2146–2162.

Geoffrion, A. M., Graves, G. W., 1974. Multicommodity distribution system design by Benders decomposition. Management Science 26 (8), 855–856.

Gomory, R. E., Hu, T. C., 1961. Multi-terminal network flows. Journal of the Society for Industrial and Applied Mathematics 9 (4), 551–570.

Hu, T. C., 1974. Optimum communication spanning trees. SIAM Journal on Computing 3 (3), 188–195.

Kerivin, H., Mahjoub, A. R., 2005. Design of survivable networks: A survey. Networks 46 (1), 1–21.

Kruskal, J. B., 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society 7 (1), 48–50.

Luna-Mota, C., 2015. The optimum communication spanning tree problem. Ph.d. dissertation, Universitat Politècnica de Catalunya - Barcelona Tech.

Magnanti, T. L., Mireault, P., Wong, R., 1986. Tailoring Benders decomposition for uncapacitated network design. In: Gallo, G., Sandi, C. (Eds.), Network Flow at Pisa. Mathematical Programming Studies, volume 26. pp. 112–154.

Magnanti, T. L., Raghavan, S., 2005. Strong formulations for network design problems with connectivity requirements. Networks 45 (2), 61–79.

Magnanti, T. L., Wolsey, L. A., 1995. Chapter 9 optimal trees. In: Network Models. Vol. 7 of Handbooks in Operations Research and Management Science. Elsevier, pp. 503 – 615.

Magnanti, T. L., Wong, R. T., 1981. Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. Operations Research 29 (3), 464–484.

Mitra, G., 1973. Investigation of some branch and bound strategies for the solution of mixed integer linear programs. Mathematical Programming 4 (1), 155–170.

Muckstadt, J., Wilson, R., 1968. An application of mixed-integer programming duality to scheduling thermal generating systems. Power Apparatus and Systems, IEEE Transactions on PAS-87 (12), 1968–1978.

Ortiz-Astorquiza, C., Contreras, I., Laporte, G., 2017. An exact algorithm for multi-level uncapacitated facility location. S*ubmitted to* T*ransportation* S*cience*.

Palmer, C., 1994. Two algorithms for finding optimal communications spanning trees. Tech. rep., Thomas J. Watson IBM Research Center.

Palmer, C. C., Kershenbaum, A., 1995. An approach to a problem in network design using genetic algorithms. Networks 26 (3), 151–163.

Papadakos, N., 2008. Practical enhancements to the Magnanti-Wong method. Operations Research Letters 36 (4), 444–449.

Papadakos, N., 2009. Integrated airline scheduling. Computers and Operations Research 36 (1), 176–195, part Special Issue: Operations Research Approaches for Disaster Recovery Planning.

Papadimitriou, C. H., Yannakakis, M., 1991. Optimization, approximation, and complexity classes. Journal of Computer and System Sciences 43 (3), 425 – 440.

Peleg, D., Reshef, E., 1998. Deterministic polylog approximation for minimum communication spanning trees. In: Larsen, K. G., Skyum, S., Winskel, G. (Eds.), Automata, Languages and Programming: 25th International Colloquium, ICALP'98 Aalborg, Denmark, July 13–17, 1998 Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 670–681.

Prim, R. C., Nov 1957. Shortest connection networks and some generalizations. The Bell System Technical Journal 36 (6), 1389–1401.

Randazzo, C., Luna, H., 2001. A comparison of optimal methods for local access uncapacitated network design. Annals of Operations Research 106 (1-4), 263–286.

Rothlauf, F., 2007. Design and applications of metaheuristics. habilitation, Universitat Mannheim.

Rothlauf, F., 2009. On optimal solutions for the optimal communication spanning tree problem. Operations Research 57 (2), 413–425.

Rothlauf, F., Goldberg, D. E., 2002. Representations for Genetic and Evolutionary Algorithms. Physica-Verlag.

Sharma, P., Mar 2006. Algorithms for the optimum communication spanning tree problem. Annals of Operations Research 143 (1), 203–209.

Soak, S.-M., 2006. A new evolutionary approach for the optimal communication spanning tree problem. IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences E89-A (10), 2882–2893.

Tilk, C., Irnich, S., 2016. Combined Column-and-Row-Generation for the Optimal Communication Spanning Tree Problem. Working Papers 1613, Gutenberg School of Management and Economics, Johannes Gutenberg-Universität Mainz.

Tutte, W. T., 1946. On hamiltonian circuits. Journal of the London Mathematical Society s1-21 (2), 98–101.

Wu, B. Y., 2002. A polynomial time approximation scheme for the two-source minimum routing cost spanning trees. Journal of Algorithms 44 (2), 359 – 378.

Wu, B. Y., Chao, K.-M., Tang, C. Y., 2000a. Approximation algorithms for some optimum communication spanning tree problems. Discrete Applied Mathematics 102 (3), 245 – 266.

Wu, B. Y., Chao, K.-M., Tang, C. Y., 2000b. A polynomial time approximation scheme for optimal product-requirement communication spanning trees. Journal of Algorithms 36 (2), 182 – 204.

Wu, B. Y., Lancia, G., Bafna, V., Chao, K.-M., Ravi, R., Tang, C. Y., 2000c. A polynomial-time approximation scheme for minimum routing cost spanning trees. SIAM Journal on Computing 29 (3), 761–778.

Zetina, C. A., Contreras, I., Cordeau, J.-F., 2017. Exact algorithms for the multicommodity uncapacitated fixed-charge network design problem. Tech. Rep. CIRRELT 2017-69.