Submitted to *Transportation Science*
manuscript (Please, provide the mansucript number!)

A Benders Decomposition Approach for the Symmetric TSP with Generalized Latency arising in the design of Semi-flexible Transit Systems

Fausto Errico
CIRRELT and Département de génie de la construction, École de technologie supérieure de Montréal, fausto.errico@etsmtl.ca

Teodor Gabriel Crainic
CIRRELT and Département de management et technologie, École des sciences de la gestion, Université du Québec à Montréal
teodorgabriel.crainic@cirrelt.ca

Federico Malucelli
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, federico.malucelli@polimi.it

Maddalena Nonato
School of Engineering, Università degli Studi di Ferrara, maddalena.nonato@unife.it

We present the Symmetric TSP with Generalized Latency, TSP-GL, a new problem arising in the planning of the important class of Semi-flexible transit systems. The TSP-GL can be seen as a very challenging variant of the Symmetric TSP (S-TSP), where the objective function combines the usual cost of the circuit with a routing component accounting for the passenger travel times. The main contributions of the paper include the formulation of the problems in terms of multicommodity flows, the study of its mathematical properties, and the introduction of a Branch-and-Cut approach based on Benders reformulation taking advantage of properties that relating the feasible region of the TSP-GL and the S-TSP polyhedron. An extensive computational experimentation compares a number of variants of the proposed algorithm, as well as a commercial solver. These experiments show that the method we propose outperforms significantly a well-known commercial solver, and obtains good-quality solutions to realistically-sized instances within short computational times.

Key words: Symetric TSP with generalized latency, Semi-flexible transit, Traveling Salesperson Problem, Latency, Benders Decomposition, Branch-and-Cut

1. Introduction

The term *Semi-fexible Transit Systems* (*STSs*) identifies a family of public transit systems particularly suited to medium/low demand zones and periods providing demand-responsive service within the framework of regularly-scheduled transit. Thus, a STS operates along a basic path made up of a series of stops according to a fixed, predetermined timetable, similarly to traditional transit. Yet, part of the STS service is flexible responding to user requests for service at optional locations. The fundamental idea behind such systems is that the regularity of the service is by itself a valuable property of public transportation because it helps users to plan their trips, facilitates integration with other transportation modes, makes it possible to access the service without booking, etc. At the same time, STSs aim to inject flexibility into operations by considering "additional" time for the vehicle to perform its route, time that can then be used to possibly deviate from the basic path to operate services in a demand-responsive framework.

The first STSs were introduced in the early '90s and the number of STS implementations consistently increased over the years. Recent statistics report that some 40% of the transit companies in North America adopt some kind of STSs (see Koffman 2004, Potts et al. 2010, for a detailed review of STS experiences in North America and a broad analysis and discussion of their economic and social impacts). Planning STS operations turns out to be an extremely complex activity because such services, combining characteristics of traditional and on-demand systems, require both a service-design phase and user-request-dependent adjustments of vehicle routes and schedules at operational level. A comprehensive literature review on methodological aspects related to STS planning can be found in Errico et al. (2013).

As underlined in Errico et al. (2013), there is a significant gap between the methodological development required by STS planning and the current methods available in literature. The scope of this paper is to contribute to filling this gap. More specifically, we focus on STS tactical planning and provide an efficient methodology for the optimization of one of the core problems arising in this context, the *Symmetric TSP with Generalized Latency* (*TSP-GL*). Errico et al. (2011) described the way the TSP-GL arises within and interacts with the broader STS tactical planning problem. Thus, they experimentally underlined the importance of passenger travel times as a critical element of the service-quality evaluation from the passenger perspective. In this respect, the TSP-GL plays a fundamental role allowing for the simultaneous optimization of passenger travel times and company operating costs. Errico et al. (2011) did not provide a methodology to optimize the TSP-GL. This is the primary scope of the present paper.

The TSP-GL can be seen as a variant of the *Symmetric TSP* (*S-TSP*), where a more complex objective function accounts for passenger travel times, and can be described as follows. As in the S-TSP, we are given a complete undirected graph with non negative costs on the edges and a Hamiltonian circuit has to be found. In addition, we are given an origin-destination demand matrix, where each entry specifies the amount of demand to be routed between the corresponding origin and destination nodes of the graph. We are also given the traversing times or unit routing costs for every edge and for both directions. Therefore, besides finding a Hamiltonian circuit, we must also find the routing of the demand on the circuit. Consequently, the objective function of the TSP-GL accounts for two components, the first considering the cost of the edges in the Hamiltonian circuit (*installation cost*), while the second accounts for the overall time spent to traverse the required portion of the circuit by each demand unit (*routing cost* or *latency*).

It is this characterizing objective function that makes the TSP-GL a very challenging problem. On the one hand, the TSP-GL is at least as difficult as the TSP as it reduces to a S-TSP when the latency component is removed from the objective function. On the other hand, the TSP-GL is computationally very challenging and harder than the TSP due to the very strong interconnections among the problem decisions. Two families of problems known in the literature are related to the TSP-GL: The Fixed-Charge Network Design Problem (FNDP) and several variants of the TSP. Nonetheless, the specific structure of the TSP-GL does not allow for a successful adaptation of existing algorithms for the above mentioned problems and new ad hoc solution methods are required to address practical size instances.

We propose a multicommodity flow based formulation for the TSP-GL as an extension of the traditional S-TSP formulation where new variables are used to express the specific objective function of the TSP-GL. This formulation has the advantage of being compact, with a polynomial number of variables and constraints with respect to the dimension of the problem. This formulation also presents some drawbacks, however. In spite of its compactness, the solution of the LP relaxation requires very long computing times, and the lower bounds obtained in this way are quite loose. These reasons motivated us to investigate a solution approach that combines Benders decomposition, implicit enumeration principles, and the generation of advanced valid inequalities.

Traditional Benders decomposition requires to iteratively solve a *master problem*, which is a relaxation of the original problem, and a (dual) *subproblem* used to generate inequalities improving the current solution of the master problem. The inequalities generated may be either *optimality* or *feasibility* cuts. Differently from the traditional scheme where Benders cuts are only generated at integer master solutions, we consider an alternative scheme in this paper, where integrality constraints of the master problem are relaxed, and integrality is recovered by embedding the Benders master problem into a Branch and Cut framework. We call such an approach the *Benders-Branch-and-Cut* (*BBC*) algorithm. With respect to the traditional Benders decomposition scheme, the BBC algorithm is particularly advantageous for the TSP-GL because, by projecting flow variables and relaxing integrality, it allows to take full advantage of the relations between TSP-GL and S-TSP polyhedra.

Starting from the Benders reformulation, we consider three alternative families of inequalities to be deployed as feasibility cuts. We prove mathematical relations among the lower bounds provided by each family. We implemented the corresponding versions of the BBC algorithm and experimentally compared their efficiency. All BBC algorithms widely outperform a state-of-the-art MIP solver applied to the initial multicommodity flow formulation. As a second step, we refined the most efficient BBC algorithm by implementing a number of techniques known to improve the performance of the traditional Benders decomposition. We experimentally compared such refinements and the resulting best choices were used to address a set of newly-created benchmark instances derived from the well-known TSPLIB library. It turns out that the refined BBC algorithm is able to optimality solve instances with up to 42 nodes, while providing a feasible solution within 8.8% of the optimality in 5 hours of computing time for the most difficult 101-node instance.

The reminder of this paper is organized as follows. We introduce the TSP-GL multicommodity flow based formulation and survey works related to the TSP-GL in Section 2. We then, Section 3, recall the main Benders-decomposition ideas and apply Benders reformulation to the TSP-GL. Section 4 describes the general BBC algorithm, motivates our choices, and examines three versions of the BBC algorithm together with their mathematical properties. We report the details of the experimental campaign, computational results and analyses in Section 5, and draw our conclusions in Section 6.

2. Problem statement and related works

The tactical planning of a STS line consists of three major interrelated activities: 1) selecting transit stops to be visited according to a regular, predetermined, timetable (*compulsory stops*); 2) establishing a generalized ordering among transit stops in the service area and 3) determining suitable timetables for compulsory stops (*master schedule*). Errico et al. (2011) proposed several decomposition strategies for this tactical planning problem and the TSP-GL arose as a core problem related to activity 2). In such a context, one needs to define the main components of the line configuration providing a sequence among compulsory stops and the service region to be serviced between consecutive compulsory stops. In defining such a general ordering of the transit line, it is important to account for operations costs as well as for the level of service offered to users and, specifically, for the time spent by users on the vehicle.

As a consequence, the TSP-GL problem involves two levels of decisions. On the one hand, a set of edges defining a Hamiltonian circuit must be selected. On the other hand, the demand has to be routed on the circuit and the routing direction is an issue. This suggests the use of a mixed graph since both edges and directed arcs are used.

Consider a complete mixed graph $G = (N, E \cup A)$, where $N = \{1, \ldots, n\}$ is the set of nodes, $E = \{[i, j] : i, j \in N, i < j\}$ is the set of edges, and $A = \{(i, j) : i, j \in N, i \neq j\}$ is the set of directed arcs. A *design* cost $\bar{c}_e > 0$ is associated to each edge $e = [i, j] \in E$. Moreover, the amount of demand

d_{hk} is given for each origin and destination node pair (h,k). Let D denote the set of pairs having non-zero demand, that is $D = \{(h,k) : h,k \in N, d_{hk} > 0\}$. A routing cost $q_{ij}^{hk} \geq 0$ is associated to every arc $(i,j) \in A$ and every demand $(h,k) \in D$.

The problem consists in finding a subset $X \subset E$ of edges defining a Hamiltonian circuit in G and a routing of every demand pair (h,k) coherently with the circuit. The objective function to be minimized is the sum of the design costs of the edges in X and the total routing cost on the arcs traversed by every demand pair (h,k). Notice that no capacity restriction is considered in the definition of the TSP-GL.

2.1. Multicommodity flow-based formulation

The most natural way to formulate the problem is to use an integer multicommodity flow-based model. For each edge $e = [i,j] \in E$, we introduce a binary variable x_e with value 1 if the edge $[i,j]$ is in the circuit X and 0 otherwise. Given a circuit, every demand pair (h,k) must be routed along one of the two possible paths from h to k in the circuit, depending on the direction. Because there is no capacity restriction, all the demand units will follow the shortest path. Therefore, we introduce multicommodity unit flow variables for each commodity $(h,k) \in D$ and for each arc $(i,j) \in A$: f_{ij}^{hk} assumes value 1 if demand from h to k travels on arc (i,j) and 0 otherwise.

We use the following notation to simplify the formulation: given a subset of nodes $S \subset N$, $\delta(S) \subset E$ denotes the set of edges having exactly one endpoint in S and $\delta(i)$ is used as a shorthand of $\delta(\{i\})$. Similarly, $\delta^+(S) \subset A$ ($\delta^-(S) \subset A$) denotes the set of arcs having tail in S and head in $N \setminus S$ (head in S and tail in $N \setminus S$, respectively). We use $\delta^+(i)$ ($\delta^-(i)$) as a shorthand of $\delta^+(\{i\})$ ($\delta^-(\{i\})$). Given a subset $E^* \subseteq E$ and a subset $A^* \subseteq A$, we write $x(E^*)$ for $\sum_{e \in E^*} x_e$ and $f^{hk}(A^*)$ for $\sum_{(i,j) \in A^*} f_{ij}^{hk}$, respectively.

In its general form, the TSP-GL may be formulated as follows:

$$\min \ (1-\alpha)\sum_{e \in E} \bar{c}_e x_e + \alpha \sum_{(i,j) \in A} \sum_{(h,k) \in D} q_{ij}^{hk} f_{ij}^{hk} \tag{1}$$

$$x(\delta(i)) = 2 \qquad \forall\, i \in N \tag{2}$$

$$f^{hk}(\delta^+(i)) - f^{hk}(\delta^-(i)) = \begin{cases} 1 & \text{if } i = h, \\ -1 & \text{if } i = k, \\ 0 & i \neq h,k \end{cases} \qquad \forall\, (h,k) \in D,\ \forall i \in N \tag{3}$$

$$f_{ij}^{hk} \geq 0 \qquad \forall\, (i,j) \in A,\ \forall (h,k) \in D \tag{4}$$

$$x_e - f_{ij}^{hk} \geq 0 \qquad \forall e = [i,j] \in E, \forall (h,k) \in D \tag{5}$$

$$x_e - f_{ji}^{hk} \geq 0 \qquad \forall e = [i,j] \in E, \forall (h,k) \in D \tag{6}$$

$$x_e \in \{0,1\} \qquad \forall\, e \in E \tag{7}$$

The objective function (1) accounts for the two components: the sum of design costs and the latency, where α is a suitable tradeoff parameter. Constraints (2) impose that the selected edges incident to each node must be exactly two, thus they must form a cycle cover of G. Constraints (3) are the multicommodity flow balance constraints stating that one unit of flow must be sent from h to k for every $(h,k) \in D$. Constraints (5) and (6) enforce the coherence between x and f variables, that is, any commodity (h,k) cannot travel on an arcs (i,j) or (j,i) if the corresponding edge $e = [i,j]$ is not in the circuit. Observe that constraints (5) and (6) could be alternatively expressed in the apparently stronger form:

$$x_e - f_{ij}^{hk} - f_{ji}^{hk} \geq 0 \qquad \forall\ e = [i,j] \in E, \forall\ (h,k) \in D. \tag{8}$$

There is no practical advantage in using this alternative expression, however, because no optimal solution of the linear relaxation of (1)-(7) would have f_{ij}^{hk} and f_{ji}^{hk} strictly positive at the same time. We keep consequently the original formulation.

Observe that for generality sake we chose to model the routing of demand by unit-flow variables and let the routing cost coefficients in the objective function account for the application-specific interpretation of such costs. For STS, routing costs stand for the time spent on-board by travellers, our model aiming to find the best compromise between the cost of operating and the displeasure felt by travellers due to too long delays. Routing cost may be set to represent different measures of the delay, e.g., a unit cost for each (h,k), the sum of the delays over all pairs (h,k), the average delay, etc. In the transit application used in the experimental phase of this study (Section 5), routing costs q_{ij}^{hk} are set to $c_{ij}\dfrac{d_{hk}}{\sum_{(s,t)\in D} d_{st}}$, where c_{ij} is the travel time from location i to location j. Therefore, the objective function component $\sum\limits_{(i,j)\in A}\sum\limits_{(h,k)\in D} q_{ij}^{hk} f_{ij}^{hk}$ measures the average passenger travel time.

Finally note that our formulation does not avoid the presence of subtours. Subtours are made infeasible by the combination of coherence constraints and flow balance constraints if D is dense enough. The only case when the presence of subtours is possible is when D defines in G more than one connected component. In this case, in order to generate Hamiltonian circuits we must introduce explicit cutset inequalities. Usually, the demand matrix D in transit line design is dense enough to avoid this occurrence. In our investigation, we focus on the case of matrices D inducing a single connected component.

2.2. Works related to the TSP-GL

The TSP-GL can be seen as a specialized version of the FNDP consisting in finding a set of links (and capacities) on which flow demands can be routed, minimizing design or routing cost or both. In the case of the TSP-GL, links have infinite capacities and the network topology is constrained to be a Hamiltonian circuit. The FNDP is a very general framework able to model a wide variety of problems and for which several exact methods have been proposed. To cite a few Benders decomposition (see Costa 2005, for a complete review), Lagrangean based techniques (e.g., Crainic et al. 2001, Holmberg and Yuan 2000, Holmberg and Hellstrand 1998, Kliewer and Timajev 2005, Sellmann et al. 2002) and polyhedral approaches (e.g., Atamtürk 2002, Barahona 1996, Bienstock and Günlük 1994, Günlük 1999, Leung and Magnanti 1989, Ortega and Wolsey 2003, Raack et al. 2011).

More specifically, the Uncapacitated FNDP (UFNDP) has been addressed in Magnanti et al. (1986) by Benders decomposition. The authors applied a preprocessing phase based on a dual-ascent procedure that allowed them to eliminate design variables and showed that the selection of Benders cuts is a key element for the success of the approach. Holmberg and Hellstrand (1998) proposed a Lagrangean relaxation of the flow conservation constraints and embedded such a relaxation scheme into a Branch and Bound algorithm. Ortega and Wolsey (2003) considered a particular case of the UFNDP where one origin only is allowed and proposed a Branch and Cut algorithm. The so-called *dicut* inequalities showed to be very effective in their approach.

The TSP-GL also recalls a similar problem, called Generalized Minimum Latency Problem, introduced in Errico (2008). In the Generalized Minimum Latency Problem, we are given a complete directed graph $G = (N, A)$ were directed arcs are associated with a design cost and a routing cost. As in TSP-GL, origin-destination demand coefficients d_{hk} are given. The problem consists in determining a Hamiltonian directed cycle and in routing the demand on the cycle so that the sum of the design costs of the selected arcs and the flow routing costs is minimized. The main difference with respect to TSP-GL is that the cycle is directed, design costs are not necessarily symmetric, and demand routing must follow the direction of the arcs of the cycle.

Two important problems studied in the literature can be seen as special cases of the Generalized Minimum Latency Problem depending on how the demand is specified. One is the Minimum Latency Problem (MLP), also known in literature as the Traveling Deliveryman Problem or Traveling Repairman problem. A repairman has to visit a given set of customers, who issued a request before the repairman departure. Starting from a depot, the repairman has to visit all customers minimizing the total customer waiting time that depends on the distance traveled by the repairman before he/she serves them. The MLP can be modeled as a particular case of Generalized Minimum Latency Problem where, supposing that the depot is in node 1, $d_{1k} = 1$ for $k = 2, \dots, n$ and 0 for all the other origin/destination pairs. Moreover the design costs are equal to zero for all arcs. The other problem is the Multicommodity TSP which is very similar to the MLP problem with the exception that the demand coefficients between node 1 and the others can assume any value greater than 0 rather than being unitary.

Lucena (1990) formulates the MLP using a time dependent TSP and obtains lower and upper bounds via dynamic programming and Lagrangean relaxation for instances with up to 30 nodes. More recently, Méndez-Díaz et al. (2008) proposed an evolution of the previous approach exploiting the polyhedral structure of the problem and increasing the size of the solved instances to 40 nodes. Different lower and upper bounds obtained using cumulative matroids and Lagrangean decomposition were proposed by Fischetti et al. (1993). The Multicommodity TSP is studied in Sarubbi and Luna (2007) and Sarubbi et al. (2008) where a classical Benders decomposition and a Quadratic Assignment approach are proposed. Recently, Ortiz-Astorquiza et al. (2015) studied a particular case of the TSP-GL that does not account for design costs.

We observe that Benders decomposition has been adopted for several problems related to the TSP-GL and this motivated our study. However, as detailed in Section 4, the particular structure of the TSP-GL does not allow for a straightforward extension of the known methods. We exploit the structure to develop an efficient cutting plane algorithm based on Benders decomposition and polyhedral properties of the TSP-GL, and embedd it into an implicit enumeration scheme.

3. Benders decomposition and TSP-GL

Benders decomposition was introduced in Benders (1962) as a method to solve mixed integer programs. It applies in particular to structured problems where we can identify a subset of "complicating" variables for which, when fixed, the problem reduces to an easy one. In our case, it is easy to notice that if x variables defining the Hamiltonian circuit are fixed, the resulting restricted problem reduces to a set of independent minimum cost flow problems without capacity constraints. Let us analyze the Benders reformulation for the TSP-GL and illustrate the classical Benders decomposition.

Given a vector $\bar{x} \in \{0,1\}^{|E|}$, and denoting $z(\bar{x}) = \sum_{(h,k)\in D} z^{hk}(\bar{x})$ the total contribution of the latency component, the TSP-GL reduces to solving a so-called *PrimalSubproblem* (PS) that may be decomposed into independent one-commodity minimum cost flows, one for each origin-destination pair, where

$$z^{hk}(\bar{x}) = \min \sum_{(i,j)\in A} q_{ij}^{hk} f_{ij}^{hk} \tag{9}$$

$$f^{hk}(\delta^+(i)) - f^{hk}(\delta^+(i)) = \begin{cases} 1 & i = h, \\ -1 & i = k, \\ 0 & \forall i \in N \setminus \{h,k\} \end{cases} \qquad \forall\, i \in N \tag{10}$$

$$-f_{ij}^{hk} \geq -\bar{x}_e \qquad \forall\, i,j \in N,\ i<j,\ e=[i,j] \tag{11}$$

$$-f_{ji}^{hk} \geq -\bar{x}_e \qquad \forall\, i,j \in N,\ i<j,\ e=[i,j] \tag{12}$$

$$f_{ij}^{hk} \geq 0 \qquad \forall\,(i,j) \in A. \tag{13}$$

Observe that the strong duality holds for every flow subproblem. Hence, we can equivalently consider the *Dual Subproblems* (DS)

$$z^{hk}(\bar{x}) = \max \quad \rho_h^{hk} - \rho_k^{hk} - \sum_{e \in E} \bar{\lambda}_e^{hk} \bar{x}_e \tag{14}$$

$$\rho_i^{hk} - \rho_j^{hk} - \lambda_{ij}^{hk} \leq q_{ij}^{hk}, \qquad \forall\, (i,j) \in A \tag{15}$$

$$\lambda_{ij}^{hk} \geq 0 \qquad \forall\, (i,j) \in A \tag{16}$$

$$\rho_i^{hk} \gtreqless 0 \qquad \forall\, (i,j) \in A \tag{17}$$

where $\bar{\lambda}_e := \lambda_{ij} + \lambda_{ji}, \quad \forall e = [i,j] \in E$.

Let us denote $Q = \{(\lambda, \rho) \in \mathcal{R}^{(|A|+|N|)|D|} \mid (\lambda, \rho) \text{ satisfy } (15), \ldots, (17)\ \forall (h,k) \in D\}$. Given that for a generic x, $z(x)$ is obtained as a pointwise maximum of affine functions and that Q has a finite number of extreme points and rays, the TSP-GL can be expressed by Benders reformulation:

$$\min \quad \alpha\, \eta + (1-\alpha) \sum_{e \in E} \bar{c}_e x_e \tag{18}$$

$$\eta + \sum_{e \in E} \sum_{(h,k) \in D} \bar{\lambda}_e^{hk} x_e \geq \sum_{(h,k) \in D} (\rho_h^{hk} - \rho_k^{hk}) \qquad \forall (\lambda, \rho) \in extr(Q) \tag{19}$$

$$\sum_{e \in E} \sum_{(h,k) \in D} \bar{\lambda}_e^{hk} x_e \geq \sum_{(h,k) \in D} (\rho_h^{hk} - \rho_k^{hk}) \qquad \forall (\lambda, \rho) \in rays(Q) \tag{20}$$

$$x(\delta(i)) = 2 \qquad \forall\, i \in N \tag{21}$$

$$x_e \in \{0,1\} \qquad \forall e \in E \tag{22}$$

where $extr(Q)$ and $rays(Q)$ are the set of extreme points and rays of Q respectively.

Notice that the problem (18)-(22) does not contain f variables, but the number of constraints (19) and (20) (called *optimality* and *feasibility* cuts, respectively) is very high and this suggests the use of a cutting plane approach. In fact Benders decomposition is nothing but a cutting plane algorithm applied to the Benders reformulation. The classic version of the algorithm considers a relaxation of the Benders reformulation, called *Master Problem* (MP), with only a small subset of optimality and feasibility cuts. The master problem, which is itself a MIP, is solved. If the resulting DS is unbounded, then a feasibility cut is added. If DS is bounded, either the current solution is optimal or it is possible to identify a violated optimality cut. The process then iterates.

4. Three Benders-based algorithms

We now describe three algorithms addressing the TSP-GL based on Benders decomposition. In Section 4.1 we present the main motivations and formalize the common algorithmic scheme, called Benders-Branch-and-Cut (BBC). The three versions of the BBC differ on the type of generated feasibility cuts. In Section 4.2, we describe a basic version of the algorithm (BBC1) based on the classical Benders reformulation (18)-(22) and prove some mathematical properties. In Section 4.3, we underline the common points between TSP-GL and S-TSP and describe two algorithms that take advantage of such similarities (BBC2 and BBC3).

4.1. General algorithmic approach and motivations

The classic version of the Benders decomposition (CBD) requires the solution of a MIP at *every* master iteration. The drawback is that the complexity of such MIPs rapidly increases with the number of added constraints, which usually spoil the structure of the original problem, and convergence to optimality soon becomes impractical, except for small instances. For this reason, research focused on obtaining Benders cuts in alternative ways so as to reach a better convergence. One of the most popular techniques was proposed by McDaniel and Devine (1977). Starting from the observation that the feasible region of a dual Benders subproblem does not depend on the particular master problem solution, the authors deduce that any extreme point or ray of a dual subproblem, independently on how it was obtained, can be used to generate valid Benders cuts. Such a property implies, for example, that the solution obtained by solving the *linear relaxation* of the MP can be used to obtain valid Benders cuts, with the advantage of being much easier to solve than the corresponding integer version. McDaniel and Devine (1977) proposed a Modified Benders Decomposition (MBD) where the initial cuts were obtained by applying the Benders scheme to the linear relaxation of the MP. When certain conditions applied, the authors proposed to switch to CBD. MBD has been successfully applied in several works, e.g., Rei et al. (2009), Cordeau et al. (2001, 2000).

Observe that stopping the continuous phase of the MBD when no Benders cuts can be found corresponds to solving the linear relaxation of the initial formulation via Benders decomposition. It is actually possible to further modify MBD to progressively recover integrality by embedding the linear relaxation of the MP into a branch-and-bound scheme, instead of switching to the CBD. We name such a solution scheme Benders-Branch-and-Cut algorithm (BBC). Although the BBC has been already adopted in some works, as for example in Sridhar and Park (2000), Rodríguez-Martín and Salazar-González (2008), Ljubić et al. (2012), its popularity is rather limited with respect to CBD or MBD. However, Naoum-Sawaya and Elhedhli (2012) found that the BBC was up to seven time faster than CBD with Pareto-optimal cuts (for details on Pareto-optimal cuts, see Magnanti and Wong 1981) when applied to capacitated facility location problem.

The adoption in our work of the BBC framework instead of CBS or MBS plays a major role in the success of our algorithm. This, for the following two main reasons: 1) similarly to what Naoum-Sawaya and Elhedhli (2012) observed, our preliminary attempts to solve TSP-GL by CBD and MBD turned out to be successful only for very small instances of TSP-GL, even if a certain number of additional algorithmic improvements were implemented (Pareto-optimal cuts, heuristic generation of Benders cuts, warm-restart, etc.) and 2) in Section 4.3 we show that exploiting the polyhedral relation between the TSP-GL and the S-TSP considerably improves the algorithmic performances. Even if CBD and MBD can partially take advantage of these relations, only the BBC allows for their full exploitation.

In order to handle the enumeration, the BBC algorithm makes use of a list of candidate problems, which is originally empty. The first problem (the root node) initially consists of a linear relaxation of MP where only degree constraints (2) are considered together with a set of optimality cuts obtained heuristically. This problem is iteratively solved in a cutting plane scheme where separation routines look for violated feasibility or optimality cuts. It should be noticed that, differently from traditional branch-and-cut, the current problem optimizes a function that is a lower bound of the actual cost of a solution. Consequently, the cost of a solution is only known when the corresponding optimality cut has been generated. This implies that the sequence of costs of the relaxed solutions is not necessary non-decreasing. Once the current problem optimality is reached, the algorithm performs the usual branching and fathoming operations.

4.2. BBC1: The basic version

The BBC1 algorithm implements the BBC scheme and adopts the feasibility cuts defined by (20). Let us denote z_1 the value of the optimal solution of the linear relaxation of the MP in BBC1, and z_{MF} the value of the linear relaxation of the multicommodity flow formulation (1)-(7). It is known that the classical Benders reformulation (18)-(22) is equivalent to the original problem in terms of lower bounds provided by their linear relaxation, hence $z_1 = z_{MF}$.

The natural extension to the full space of any extreme ray of $Q^{hk} = \{(\lambda^{hk}, \rho^{hk}) \in \mathcal{R}^{|A|+|N|} \mid (\lambda^{hk}, \rho^{hk}) \text{ satisfy } (15), \ldots, (17)\}$ for any given $(h,k) \in D$ is a ray of Q itself (for a formal proof about sets of ray generators of cones with diagonal block structure see Padberg and Sung 1991). Moreover, recall that the Benders subproblem decomposes into smaller independent subproblems, one per commodity. Constraints (20) can be consequently substituted by the following disaggregated version:

$$\rho_h^{hk} - \rho_k^{hk} - \sum_{e \in E} \bar{\lambda}_e^{hk} x_e \le 0 \quad \forall (\lambda^{hk}, \rho^{hk}) \in extr(Q^{hk}),\ \forall (h,k) \in D. \tag{23}$$

Let us denote z_{CI} the value of the optimal linear relaxation of (18), (19), (21), (22) and the following Cutset Inequalities:

$$x(\delta(S)) \ge 1 \quad \forall S \subset V. \tag{24}$$

Proposition 1 in Appendix shows that $z_1 = z_{CI}$, i.e., the two formulations provide the same lower bound.

From the algorithmic point of view, BBC1 separates inequalities (23) by solving a minimum cost flow algorithm for each commodity, using a network simplex algorithm, for example. As soon as a violated inequality is found, the separation algorithm is stopped and the corresponding cut is added.

4.3. BBC2 and BBC3: Exploiting the Hamiltonian circuit structure

The formulation (1)-(7) does not explicitly exploit the particular Hamiltonian circuit structure of the design part of TSP-GL. An important property that plays a central role for BBC2 and BBC3 is that the projection of (1)-(7) on the x-space corresponds to the polyhedron of the S-TSP. Consequently, every valid inequality for the S-TSP is also valid for the projection of the TSP-GL on the x-space. There has been an extensive research effort to study the S-TSP polyhedron and a number of families of facet-defining inequalities have been found (see, for example, Gutin and Punnen 2002). These considerations suggest to exploit such knowledge about the S-TSP polyhedron to possibly strengthen the lower bound given by the linear relaxation of the MP. To this purpose, consider the well-known family of S-TSP facet-defining inequalities called *Subtour Elimination Constraints* (*SEC*):

$$x(\delta(S)) \geq 2 \quad \forall S \subset V, \tag{25}$$

and let us call P_{MF} the polyhedron of the linear relaxation (1)-(7), and P_{SEC} the polyhedron defined by P_{MF} with the addition of SEC inequalities (25). Clearly, $P_{SEC} \subseteq P_{MF}$, and it is possible to prove that such an inclusion is tight. Consequently, inequalities (25) can be exploited to improve the lower bound provided by BBC1.

Let P_{FAC} stand for the polyhedron defined by including an additional set of families of S-TSP facet-defining inequalities into P_{SEC}. Because the added inequalities are facet defining for the S-TSP and because the projection on the x-space of P_{MF} is the S-TSP polyhedron, it follows that P_{FAC} is strictly included in P_{SEC}. We may therefore conclude that the following hierarchy of bounds holds true:

Corollary 1. *Denote z_{SEC} and z_{FAC} the optimal value of minimizing the objective function* (1) *over polyhedra P_{SEC} and P_{FAC}, respectively. Then, $z_1 = z_{MF} \leq z_{SEC} \leq z_{FAC}$.*

Notice that the improvement of z_{MF} by adding S-TSP cuts to the original formulation (1)-(7) is far from being practical. In fact, as we will point out in Section 5.2, solving the linear relaxation of the original formulation is extremely time consuming and solution techniques requiring the iterated solution of such a problem cannot be efficient. Similarly, S-TSP cuts can be adopted, in a CBD context, to improve the convergence of the integer MP. However, such a problem rapidly becomes intractable as the number of cuts increases. In this sense, the BBC framework, allowing to solve an integer relaxation of the MP, can fully exploit cutting plane generation and in particular the Hamiltonian structure of the network design in the TSP-GL.

BBC2 and BBC3 implement the general BBC algorithm and differ in the families of S-TSP cuts generated. In the following, we briefly list inequalities and the adopted separation algorithms.

4.3.1. BBC2. BBC2 is similar to BBC1, but adopts inequalities (25) as feasibility cuts instead of (20). Denoting z_2 the optimal value of the linear relaxation provided by algorithm BBC2, it is clear that $z_2{=}z_{SEC}$. For the separation of inequalities (25), BBC2 adopts the exact method described in Padberg and Rinaldi (1990).

4.3.2. BBC3. In addition to constraints (25) introduced by BBC2, BBC3 considers a variety of additional families of S-TSP facet-defining inequalities such as comb inequalities and local cuts. Let us denote z_3 the optimal value of the linear relaxation provided by algorithm BBC3. From Corollary 1, we can clearly deduce that $z_3 \geq z_2$.

Recall that a *comb* C is a subset of N and is specified by a *handle* $H \subseteq N$ and an odd number of disjoint *teeth* $S_1, \ldots, S_k \subseteq N$ such that each tooth has at least one node in H and at least one node not in H. For every comb C, the corresponding inequality is:

$$x(\delta(H)) + \sum_{j=1}^{k} x(\delta(T_j)) \geq 2k + 1. \tag{26}$$

It is not known whether the general form of comb inequalities admits a polynomial-time algorithm. However, polynomial algorithms or fast heuristics for specific comb inequalities have been proposed. BBC3 implements both a heuristic and an exact algorithm described in Padberg and Rao (1982) for *blossoms*, i.e., combs whose teeth have only two nodes. BBC3 also implements a heuristic separation of comb inequalities starting from *blocks*, i.e., maximal two-connected components of the residual graph induced by the relaxed solution. Comb inequalities from blocks are separated following the algorithm in Naddef and Thienel (2002). Finally, BBC3 implements separation procedures for *local* cuts. Such cuts do not follow a predefined template (as subtour elimination or comb inequalities), but are obtained by suitably shrinking nodes of the residual graph induced by a fractional solution and mapping violated inequalities back in the original graph representation. Details on local cuts can be found in Applegate et al. (2007), Chapter 11.

5. Computational experience

We perform three experimentation campaigns with the scope of 1) comparing lower bounds and their computational efficiency, 2) selecting the overall best algorithm, 3) testing the selected algorithm on suitable benchmark instances. More specifically, the first campaign aims at identifying which algorithm among BBC1, BBC2, and BBC3 gives the best performance in terms of root bounds and computing times. For completeness, we also compare BBC algorithms with a state-of-the-art LP solver applied to (1)-(7). Since there is computational evidence of the superiority of BBC3, in the second campaign we test several possible implementation options applied to BBC3 with the scope of identifying a unique algorithmic setting that averagely outperforms the others. We call the resulting algorithm BBC3+. In the third experimental campaign, we test BBC3+ on a new set of benchmark instances suitably derived from the well-known TSPLIB library. Results show that benchmark instances with up to 42 nodes are solved to optimality and, in the worst case, a feasible solution within 8.2% of the optimal value is obtained on a 101-nodes instance.

We initiate the presentation with a description of how instance sets were generated. We then proceed to report on the three experimental campaigns.

5.1. Instance sets

We built two instance sets, each with a specific purpose, TSPGL1 to compare algorithms and algorithmic options (first two campaigns), and TSPGL2 to benchmark BBC3+ in the third campaign. The construction criteria were mostly the same for all sets, and were inspired by two main principles: 1) be as general as possible; 2) be representative of our main application context, the design of semi-flexible transit lines. Thus, for statistical robustness, TSPGL1 includes a wide variety of problem types, with several randomly-generated instances for each type. Given that TSPGL2 provides the first benchmark instance set for the TSP-GL, it was built upon the widely used TSPLIB benchmark library and includes one instance only for each problem type. For both instance sets, the problem dimension range reflects typical situations found in our reference application.

Each instance in TSPGL1 is characterized by a set of nodes and a demand matrix D. The nodes are uniformly generated in a square of side length 100 and the graph is complete. c_{ij} is given by the Euclidean distance between nodes i and j, while c_e is set equal to twice the distance between the two endpoints. This choice comes from the reference application of the TSP-GL, i.e., the design of semi-flexible transit lines, (see Errico et al. 2011, for details), where two vehicles are considered to travel the transit line in opposite directions. As mentioned in Section 2, the routing costs q_{ij}^{hk} on the arcs, that model passenger travel times, are set to $c_{ij}\dfrac{d_{hk}}{\sum_{(s,t)\in D} d_{st}}$, where $\dfrac{d_{hk}}{\sum_{(s,t)\in D} d_{st}}$ is the contribution of demand (h,k). Therefore $\displaystyle\sum_{(i,j)\in A}\sum_{(h,k)\in D} q_{ij}^{hk} f_{ij}^{hk}$ measures the average passenger travel time.

As for demand matrices, we considered two different classes, reflecting the typical application context in which a few nodes, called *poles*, concentrate most of the transit demand. In particular, we considered the following scenarios:

- **C**: Complete and Polarized: The demand matrix is complete and each d_{hk} follows a uniform distribution on a given interval. However, a few poles are randomly picked among the nodes and approximately half of the total demand volume has origin *and* destination among poles.

• **S**: Sparse and Polarized: As in C, but the demand matrix is sparse and the number of destinations associated with a given node is limited to 4 and chosen randomly.

As specified in Errico et al. (2011) each of the above demand matrix classes corresponds to a different approach in the design of a semi-flexible transit line, where C and S account for more or less aggregated data, respectively.

Notice that, for completeness, we initially considered also the following two set of demand matrices:

• Complete and Uniform: Demand matrix is complete and d_{hk} are generated uniformly on a given interval.

• Sparse and Uniform: As above, but the demand matrix is sparse as the number of destinations associated with a given node is limited to 4 and chosen randomly.

The results obtained on these instances were not qualitatively different form the corresponding polarized ones.

TSPGL1 considers networks with the following number of nodes: from 15 to 40 included, by steps of 5, and 50, 60, 80, and 100. We also considered three possible values of α in the objective function: 0.25 (emphasis on design), 0.75 (emphasis on latency), 0.5 (fairness). Finally, for each network dimension, demand setting, and objective emphasis, we generated 10 instances considering different random seeds for the generation of random data. As a result, TSPGL1 is composed of 720 different instances.

The building process for TSPGL2 is quite similar to TSPGL1, with a few important differences. First of all, the networks are deduced from the TSPLIB library instead of being generated randomly. We selected all the TSPLIB instances of the S-TSP with at most 101 cities. As in the case of TSPGL1, we considered two demand scenarios for each network, C and S. Differently from TSPGL1, TSPGL2 only considers $\alpha = 0.5$ in the objective function and one only representative instance is considered for each network/demand/emphasis combination. The resulting 48 instances are available online (TSPGL2 2012).

5.2. Comparing lower bounds and efficiency

In this section, we compare root lower bounds and efficiency provided by algorithms BBC1, BBC2 and BBC3. For completeness, we considered also the solution of the linear relaxation of the original formulation (1)-(7) by a commercial solver, more specifically, we used the dual simplex solver in CPLEX 12.3.0.0, which gave better performances with respect to other algorithms. In the following, MF will denote these results.

We implemented the BBC algorithm in the C++ programming language. The master problem was solved by the MIP solver available in CPLEX 12.3.0.0 with Concert Technology. The dual simplex was used as linear relaxation solver. Separation procedures were implemented as instantiations of several user-cutcallback available in Concert Technology. In particular, for BBC1 the Benders subproblem was solved via Network Simplex Optimizer implemented in CPLEX 12.3.0.0. S-TSP separation procedures in BBC2 and BBC3, i.e., subtour elimination, heuristic and exact blossom, combs from blocks, and local cuts were all implemented by suitably adapting to our codes the separation procedures of the well-known S-TSP solver Concorde, version 03-12-19, coded in C programming language (Applegate et al. 2003). Details on the latter set of separation procedures can be found in Applegate et al. (2007). Experiments were executed on a Dual-Core AMD Opteron(tm) Processor 2218, with 4G DDR2 RAM memory on SunOS 5.10.

Instance			GAP	%Closed GAP		Times (s)			
dimension	D. type	α	BBC1	BBC2	BBC3	CPLEX	BBC1	BBC2	BBC3
15	C	0.25	3.4	73.6	78.3	5.4	0.9	0.9	0.9
20	C	0.25	3.2	82.3	90.6	40.1	5.0	3.3	4.5
25	C	0.25	3.3	69.1	85.3	265.9	17.1	9.6	12.9
30	C	0.25	3.5	62.8	80.3	1309.0	46.2	28.4	44.1
35	C	0.25	3.5	68.2	89.0	2872.3	75.8	50.5	57.4
15	C	0.5	2.8	66.8	69.1	8.4	2.4	1.8	2.4
20	C	0.5	3.5	52.5	61.1	69.4	13.4	7.9	11.7
25	C	0.5	4.3	45.3	57.8	486.9	39.7	20.2	36.3
30	C	0.5	4.9	33.7	50.4	1827.6	115.1	69.4	101.5
35	C	0.5	4.4	41.7	62.7	8779.2	277.8	157.8	178.7
15	C	0.75	3.4	25.9	29.6	16.2	8.5	5.4	11.4
20	C	0.75	6.6	20.9	26.0	98.7	32.3	26.5	49.4
25	C	0.75	7.8	15.4	23.3	862.6	109.4	77.7	109.1
30	C	0.75	9.2	13.2	19.7	4749.9	340.2	245.6	301.8
35	C	0.75	8.0	17.2	23.8	17495.8	742.8	494.9	470.2
15	S	0.25	3.4	74.4	77.2	5.3	0.6	0.5	0.5
20	S	0.25	3.2	80.0	89.9	22.1	2.7	1.8	2.5
25	S	0.25	3.4	67.9	84.0	149.2	7.7	4.3	6.6
30	S	0.25	3.5	62.7	79.8	552.4	22.2	12.7	18.9
35	S	0.25	3.5	67.9	88.0	1318.1	30.9	23.8	24.3
15	S	0.5	2.9	67.4	65.9	6.8	1.6	1.1	1.6
20	S	0.5	3.7	53.5	61.7	38.7	6.5	4.3	6.6
25	S	0.5	4.4	44.2	56.2	203.3	19.5	10.2	18.4
30	S	0.5	5.0	34.6	50.6	738.4	48.4	31.4	48.9
35	S	0.5	4.6	41.6	60.9	2339.8	90.6	54.0	77.1
15	S	0.75	3.5	28.1	31.6	11.8	5.2	3.1	7.9
20	S	0.75	6.9	23.3	28.1	50.9	16.6	14.7	33.5
25	S	0.75	8.5	16.3	22.9	276.2	51.4	33.7	70.9
30	S	0.75	9.5	13.4	19.7	1306.0	145.0	100.0	160.9
35	S	0.75	8.4	17.4	23.9	3397.3	275.7	189.5	270.3

Table 1 Comparison among CPLEX and BBC1, BBC2, and BBC3 on small instances

We report in Table 1 the numerical results of the comparison of MF with BBC1, BBC2, and BBC3 on small instances of the TSPGL1 set. Recalling that TSPGL1 includes 10 random samples for each instance type, rows in Table 1 report average results for each instance type. In the first three columns the instance type is identified by indicating the number of nodes, demand type, and value of α. The following column reports the relative percentage gap between the value of the optimal solution and the lower bound of the linear relaxation provided by z_{MF}. Given that, as observed in Section 4.2, $z_{MF} = z_1$, there is no need to report the lower bound quality of BBC1. The next two columns display the average gap percentage closed by z_2 and z_3 with respect to z_{MF}. The last four columns report the average computing times in seconds.

We observe that the lower bound provided by BBC1 (equivalent to MF) is the worst and has a gap ranging from about 3% to no more than 10%. This gap increases with size of the instances and is also affected by the value of α. When α is higher, i.e., when the travel time component of the objective function is greater, the gap increases, while when the design component dominates, i.e., when the instances are closer to a TSP problem, the bound is tighter. Note also that the gap of BBC1 is not affected by the type (C/S) of the demand matrix.

As for the computing times, there is a huge difference between the commercial software and BBC1. CPLEX times dramatically increase with the size and also testify that instances with high values of α are more difficult. Moreover, there is a great impact on computing times due to the demand matrix type. Dense instances (C) require much higher times with respect to the sparse ones (S). This is mainly due to the number of variables of the model. Such a difference in computing times is mitigated using BBC1 instead of CPLEX. Besides requiring much smaller times, BBC1 reveals also to be more stable with respect to all parameters.

Considering the other two bounds, we may observe a clear improvement in terms of gap and computing time. The gap closed by BBC2 with respect to BBC1 ranges in average from 80% for the easy instances to 13% for the more difficult ones. Notice also that, BBC2 computing times are shorter that those of BBC1. BBC3 reaches better bounds closing a larger amount of gap and has computing times comparable with BBC1 and sometimes smaller for big instances, thus showing a better trend with respect to the instance size.

It is interesting to notice how the TSP-GL behaves differently than the general FNDP family. In the latter problems in fact, the more important the flow component is with respect to the design component, the easier the problem becomes. This is normally explained noticing that the linear part of the problem (the flows) becomes predominant with respect to the non linear one (the design). For the TSP-GL the situation is different because the Hamiltonian structure of the network topology seems to emphasize the non-linearity of the problem when flow costs are higher. From the computational view point, we observed for BBC algorithms that the number of optimality cuts generated is much higher for high values of α. This suggests that the optimal solution of the master problem is located towards the interior of the feasible region of the projection in the x-subspace of the TSP-GL polyhedron. This would also explain why optimal solutions of

Instance			GAP	GAP %Impr.		Times (sec)			#Time limit		
dimension	D. type	α	BBC1	BBC2	BBC3	BBC1	BBC2	BBC3	BBC1	BBC2	BBC3
40	C	0.25	761.8	2.6	3.4	166	95	132	0	0	0
50	C	0.25	848.1	3.3	4.1	599	351	469	0	0	0
60	C	0.25	916.6	3.1	4.0	1354	1079	1177	0	0	0
80	C	0.25	1040.8	3.1	4.0	7266	4951	6417	0	0	0
100	C	0.25	1148.0	3.4	4.3	17937	16392	16219	9	7	6
40	C	0.5	538.2	2.3	2.9	509	393	353	0	0	0
50	C	0.5	597.0	2.9	3.6	2060	1505	1297	0	0	0
60	C	0.5	640.7	2.6	3.4	6466	4105	3340	1	0	0
80	C	0.5	726.7	2.5	3.3	17604	15836	15906	8	6	4
100	C	0.5	793.7	3.3	4.1	18000	18000	18000	10	10	10
40	C	0.75	309.3	1.7	2.1	1371	1088	1062	0	0	0
50	C	0.75	339.9	2.2	2.6	5378	4805	4237	0	0	0
60	C	0.75	358.9	1.9	2.4	15634	11760	10576	4	0	0
80	C	0.75	401.8	2.5	3.2	18000	18000	18000	10	10	10
100	C	0.75	414.0	5.8	7.0	18000	18000	18000	10	10	10
40	S	0.25	761.3	2.7	3.5	1860	38	53	1	0	0
50	S	0.25	847.6	3.3	4.1	203	116	164	0	0	0
60	S	0.25	915.7	3.1	4.0	492	345	404	0	0	0
80	S	0.25	1040.6	3.1	4.0	2477	1398	1632	0	0	0
100	S	0.25	1147.4	3.4	4.3	8105	5896	6916	0	0	0
40	S	0.5	537.3	2.4	3.0	1975	144	154	1	0	0
50	S	0.5	596.0	3.0	3.6	637	484	484	0	0	0
60	S	0.5	639.2	2.8	3.5	1525	1154	1269	0	0	0
80	S	0.5	726.3	2.5	3.4	7561	4886	5026	0	0	0
100	S	0.5	796.1	3.0	3.8	17932	17490	15872	9	9	5
40	S	0.75	308.0	1.8	2.2	494	375	485	0	0	0
50	S	0.75	338.3	2.3	2.7	1681	1448	1526	0	0	0
60	S	0.75	356.9	2.1	2.6	6429	3905	3849	1	0	0
80	S	0.75	405.1	1.8	2.5	17988	16162	16057	9	4	4
100	S	0.75	433.3	3.2	3.9	18000	18000	18000	10	10	10

Table 2 Comparison among BBC1, BBC2 and BBC3 on bigger INST1 instances. Time limit: 5h

the linear relaxation are more fractional, with consequently worse lower bounds, and why S-TSP facet-defining inequalities are less helpful for high α.

We performed a second experimental test where we compared BBC1, BBC2 and BBC3 on instances of TSPGL1 with a higher number of nodes, and limiting the computing time to 5 hours. In this second test, we did not compare with MF because the commercial solver usually failed due to memory limits. Also, differently from the previous experimentation, we do not know the values of the optimal solutions of the integer problems. We consequently compared the relative improvements of lower bound values instead of comparing the relative gaps with the optimum. Results are reported in Table 2, where the first three columns identify the instance type by specifying the number of nodes, the demand type and the value of α, respectively. The next column reports the average value of BBC1, on the 10 considered random seeds, while the following two columns report the average relative improvement of BBC2 and BBC3 over BBC1, respectively, computed as $100(z_i - z_1)/z_1,\ i = 2, 3$. The next three columns represent the average computing times, while the last three columns report the number instances for which the algorithm stops due to time limit reasons.

Table 2 confirms the trends observed in Table 1. Perhaps the most important fact underlined by the results in Table 2 is that BBC3 consistently outperforms BBC1 and BBC2 in terms of lower bounds. Its computing times are comparable with BBC2, and are even better on harder instances, as it can be deduced by comparing how many times BBC3 hits the time limit with respect to BBC1 and BBC2.

Summarizing, we can conclude that BBC algorithms clearly outperform MF solved by a state-of-the art linear programming solver. Overall, BBC3 favorably compares with BBC1 and BBC2 because it yields better lower bounds and competitive computing times. For this reason, the following experimental campaigns focus on BBC3 only.

5.3. Implementation refinements and details

The efficiency and effectiveness of Benders decomposition-based approaches depend on the quality of the implementation. Recall, however, that the main goal of this paper was not to review all the many acceleration strategies proposed in the literature for CBD or to implement all of them for the BBC3. We rather operated a selection among these techniques and do not present techniques that either seemed not suitable to our problem or proved inefficient in preliminary experiments. For example, we do not show results for Pareto-optimal cuts (Magnanti and Wong 1981) because of poor performance in our case, and we did not implemented lifting techniques for low-density cuts (see, for example, Saharidis et al. 2010, Saharidis and Ierapetritou 2010, 2013) given that our Benders cuts are generally very dense.

This section aims to evaluate the impact of the selected implementation options or refinements on the performance of BBC3. Given the high number of algorithmic combinations, we decided to perform the tests on a restricted but representative subset of TSPGL1. In particular, we only considered instances with 40, 50 and 60 nodes. In Section 5.3.1, we discuss the generation of the initial sets of Benders cuts and heuristic strategies. In Section 5.3.2, we consider the generation of disaggregated optimality cuts (multi-cut). In Sections 5.3.3 and 5.3.4, we compare alternative separation and branching strategies.

5.3.1. Initial cuts and heuristics. Heuristic methods might improve the convergence of the BBC algorithm for at least two reasons: 1) They can provide good solutions in early stages of the algorithm and this can reduce the size of the branch-and-bound tree. 2) Heuristic solutions can be used to generate optimality cuts in an easy way without having to solve a specific separation procedure. This can be of some help both in the starting phase of the algorithm, to provide good initial cuts and starting solutions, and during the algorithm.

We implemented a simple local search based on the traditional two-opt exchange move for the S-TSP. Observe in fact that any Hamiltonian circuit can be easily extended to obtain a feasible

TSP-GL solution. In particular, given a solution of the TSP-GL, we apply a two-opt procedure to the circuit defined by the x variables, update the optimal flows and evaluate the new costs. Such a procedure can be applied at several points of the algorithm. We implemented and compared the following variants of the BBC3 algorithm:

- **h1**: No initial solution or cut is provided except for the obvious $\eta \geq 0$ and no heuristic is performed.
- **h2**: We provide exactly one optimality cut to the initial MP by solving the S-TSP with edge costs given by the design costs $\bar{c}_e, \quad e \in E$. Such a solution also initializes the algorithm.
- **h3**: The initialization is as in h2. In addition, the modified two-opt local search procedure is applied to the initial solution to generate alternative solutions. The best γ solutions so obtained are used to generate optimality cuts. The best found solution initializes the algorithm.
- **h4**: The initialization is as for h3. The modified two-opt local search is applied every time an integer solution is found during the algorithm. The best γ solutions are used to generate optimality cuts and if improving solutions are found, the incumbent is updated.

Results for $\gamma = 5$ are summarized in Tables 3, 4, and 5 reporting separately the effect of instance size, demand type and value of α. In the three tables, the first column reports the feature whose variation effects are examined. Columns two to five report information about the best upper bound obtained during the solution process. In particular, many experiments failed in obtaining a feasible solution when using h1, thus the second column reports the percentage of such instances. We report in Column three, the average value of the upped bound obtained by h2. Columns four and five report the average relative improvements of h3 and h4 over h2, respectively, computed as $100(ub_2 - ub_i)/ub_2, \ i = 3,4$. The remaining columns report the average computing times of h1, h2, h3, and h4, respectively.

Feat.	Upper Bounds				Times (sec.)			
dimension	h1 % no solution	h2	h3 %Impr.	h4 %Impr	h1	h2	h3	h4
40	30.0	573.8	0.2	0.4	10086.1	9370.0	9369.9	9372.9
50	45.0	645.1	0.3	0.4	13386.9	12475.3	12477.2	12470.3
60	61.7	692.4	0.3	0.4	15229.3	14040.3	14051.8	13960.8

Table 3 Comparison among heuristic strategies by instance dimension.

Feat.	Upper Bounds				Times (sec.)			
D. type	h1 % no solution	h2	h3 %Impr.	h4 %Impr	h1	h2	h3	h4
C	50.0	637.1	0.3	0.4	13421.9	12390.9	12396.4	12351.6
S	41.1	637.2	0.3	0.4	12379.6	11532.9	11536.2	11517.7

Table 4 Comparison among heuristic strategies by demand type

Feat.	Upper Bounds				Times (sec.)			
α	h1 % no solution	h2	h3 %Impr.	h4 %Impr.	h1	h2	h3	h4
0.25	3.3	883.1	0.0	0.0	4814.8	2544.7	2552.1	2453.8
0.5	38.3	636.9	0.0	0.0	15838.3	15289.5	15291.8	15294.5
0.75	95.0	391.3	0.8	1.1	18000.0	18000.0	18000.0	18000.0

Table 5 Comparison among heuristic strategies by α.

We observe some advantages in adopting version h4. Indeed, in average it provides better upper bounds and computing times with respect to the other versions. This is due to the fact that, when h4 attains optimality, it is faster, and when it stops due to the time limit, it gives a better upper bound.

5.3.2. Optimality cuts and multicut generation. As mentioned in the general description of the algorithm, optimality Benders cuts of the form (19) can be separated by solving a minimum cost flow problem for each commodity. Moreover, since at every master iteration all the subproblems share the same network, we can exploit sensitivity analysis to more efficiently obtain the solution of one subproblem given the solution of another.

It is moreover possible to exploit the decomposability of the Benders subproblem to produce a disaggregated version of the optimality cuts. Let η^{hk} be the contribution of the commodity (h,k) to the objective function, where $\sum_{(h,k)\in D}\eta^{hk}=\eta$. Substituting in (18), we obtain:

$$\min \alpha \sum_{(h,k)\in D} \eta^{hk} + (1-\alpha)\sum_{e\in E}\bar{c}_e x_e, \tag{27}$$

and the disaggregated cuts are then

$$\eta^{hk} + \sum_{e\in E}\bar{\lambda}_e^{hk} x_e \geq \rho_h^{hk} - \rho_k^{hk} \qquad \forall (h,k)\in D, \tag{28}$$

for every commodity (h,k). Although disaggregated cuts can accelerate the convergence of the Benders algorithm, every time optimality cuts are separated, a potentially high number of cuts are added to the MP and this can make the MP considerably harder. For this reason, several authors proposed strategies to partially re-aggregate the optimality cuts (see, e.g., Contreras et al. 2011).

We implemented and compared several versions of the optimality cut generation. In all versions, the separation of the optimality cut is performed as described above, and the difference is in the different level of aggregation. At every separation of the optimality cut:

- **o1**: Only one optimality cut (19) is added.
- **o2**: For every origin destination (h,k), a cut (27) is added.
- **o3**: We aggregate the optimality cuts of o2 by origin. More specifically, we consider η^h to be the contribution to the objective function of all commodities with origin in h, and we set $\sum_{h\in N}\eta^h=\eta$. Substituting in (18), we obtain:

$$\min\ \alpha \sum_{h\in N}\eta^h + (1-\alpha)\sum_{e\in E}\bar{c}_e x_e, \tag{29}$$

and the corresponding partially disaggregated optimality cuts

$$\eta^h + \sum_{k\in N\setminus\{h\}} \sum_{e\in E} \bar{\lambda}_e^{hk} x_e \geq \sum_{k\in N\setminus\{h\}} \rho_h^{hk} - \rho_k^{hk} \qquad \forall h \in N. \tag{30}$$

Results are summarized in Tables 6, 7, and 8. The fist column of the three tables gives the instance feature examined by the table. The second column reports the root gap values which, as expected, are the same for all versions. In the next three columns, we report average computing times to reach optimality at the root node. The next three columns report average gaps obtained within five hours of computing time. The last three columns report the average computing times.

Feat.	root % gap	root time (sec.)			final % gap			total time (sec.)		
dimension	o1, o2, o3	o1	o2	o3	o1	o2	o3	o1	o2	o3
40	5.1	300.2	108.0	101.3	3.5	3.6	3.1	9370.0	9420.2	8600.8
50	6.1	1168.9	350.7	288.3	5.1	5.2	4.8	12475.3	12934.1	12343.2
60	6.1	2876.7	814.2	620.3	5.4	5.5	5.3	14040.3	14397.3	13777.4

Table 6 Comparison among Multicut strategies by instance dimension.

Feat.	root % gap	root time (sec.)			final % gap			total time (sec.)		
D. type	o1, o2, o3	o1	o2	o3	o1	o2	o3	o1	o2	o3
C	5.7	2124.4	710.8	484.1	4.7	4.9	4.5	12390.9	13162.7	11864.3
S	5.9	772.8	137.9	189.2	4.6	4.6	4.3	11532.9	11338.4	11283.3

Table 7 Comparison among Multicut strategies by demand type.

Feat.	root % gap	root time (sec.)			final % gap			total time (sec.)		
α	o1, o2, o3	o1	o2	o3	o1	o2	o3	o1	o2	o3
0.25	1.1	298.9	187.2	168.2	0.0	0.1	0.0	2544.7	3575.1	2205.1
0.5	4.1	898.7	352.9	294.4	2.5	2.7	2.1	15289.5	15162.5	14474.1
0.75	12.1	3148.2	732.8	547.2	11.5	11.5	11.1	18000.0	18000.0	18000.0

Table 8 Comparison among Multicut strategies by α.

We observe that o2 and especially o3 are much faster than o1 in obtaining root optimality. However, when looking at the overall solution process, this advantage is weaker. We observe that option o3 remains the best one, both for the gaps between lower and upper bounds, and the average computing times.

5.3.3. Separation strategies. We aimed to keep separation strategies as simple as possible. At every separation round, the algorithms check for violated inequalities according to a predefined sequence. If violated inequalities are found, the search stops, all found inequalities are added to MP, and MP is reoptimized. When invoked again, the separation procedure will restart from the beginning. The two considered sequences are:

1. **s1**: subtour elimination, heuristic blossom, exact blossom, combs from block, local cuts, optimality cuts.
2. **s2**: subtour elimination, optimality cuts, heuristic blossom, exact blossom, combs from block, local cuts.

The difference between s1 and s2 is in the relative position of optimality cuts in each sequence: in s2 optimality cuts are only generated when all feasibility cuts are satisfied, while in s2, optimality cuts are separated when subtour inequalities are satisfied. In fact, when subtours elimination are satisfied, the network flow problem to separate optimality cuts is always feasible. The idea behind s2 is to avoid spending time trying to refine the fractional solution in a suboptimal region of the solution space.

Results are shown in Tables 9, 10 and 11. As in the previous sections, the first column presents the instance feature. Columns two and three present the average gaps obtained at root by s1 and s2, respectively, and Columns four and five their computing times. Similarly, Columns six and seven report the average final gaps and, Columns 8 and 9 the average computing times of s1 and s2, respectively.

Feat.	root % gap		root time (sec.)		final % gap		final time (sec.)	
dimension	s1	s2	s1	s2	s1	s2	s1	s2
40	5.1	5.3	300.2	279.3	3.5	3.5	9370.0	9557.2
50	6.1	6.3	1168.9	1094.5	5.1	5.1	12475.3	12681.3
60	6.1	6.3	2876.7	2729.4	5.4	5.4	14040.3	14559.2

Table 9 Comparison between Separation Strategies by instance dimension.

Feat.	root % gap		root time (sec.)		final % gap		final time (sec.)	
D. type	s1	s2	s1	s2	s1	s2	s1	s2
C	5.7	5.9	2124.4	2028.4	4.7	4.7	12390.9	12696.3
S	5.9	6.0	772.8	707.1	4.6	4.6	11532.9	11835.6

Table 10 Comparison between Separation Strategies by instance type.

Even though s1 and s2 should provide exactly the same results at the root node, we note that s1 yields better gaps than s2. The actual difference in lower bounds is due to the numerical instability of s2 that caused early exiting of the separation procedure. For this reason, we excluded s2 from further consideration, even if s1 is slower than s2 at the root.

Feat.	root % gap		root time (sec.)		final % gap		final time (sec.)	
α	s1	s2	s1	s2	s1	s2	s1	s2
0.25	1.1	1.5	298.9	214.0	0.0	0.0	2544.7	3309.9
0.5	4.1	4.2	898.7	638.4	2.5	2.5	15289.5	15442.3
0.75	12.1	12.2	3148.2	3250.8	11.5	11.5	18000.0	18000.0

Table 11 Comparison between Separation Strategies by α.

5.3.4. Branching strategy. Let $(\bar{f}, \bar{x})$ denote the optimal fractional solution of the current node. We considered the following two branching alternatives:

- **b1**: Branch on the variable x_e closest to 0.5,
- **b2**: Branch on the variable x_e maximizing

$$min\{\bar{x}_e, 1-\bar{x}_e\}(c_e + \sum_{(h,k)\in D} d_{hk}\bar{f}_{ij}^{hk}) \tag{31}$$

where $e = [i, j]$.

Strategy b1 is one of the most common ones, while strategy b2 tries to reroute the maximum possible amount of flow, with the hope of causing a stronger change in the subproblem's objective value.

Results are reported in Tables 12, 13, and 14 where the first column identifies the instance feature, Columns two and three represent the average final gaps, Columns four and five the average computing times.

Tests mostly show that the considered branching strategies perform quite similarly in terms of gaps and computing times. However, b2 occasionally attains better gaps than b1 while showing comparable computing times. Consequently we chose b2 in our final experimentation.

Feat.	final % gap		final time (sec.)	
dimension	b1	b2	b1	b2
40	3.5	3.5	9370.0	9347.8
50	5.1	5.0	12475.3	12762.3
60	5.4	5.4	14040.3	13953.8

Table 12 Comparison between branching strategies: by dimension

Feat.	final % gap		final time (sec.)	
D. type	b1	b2	b1	b2
C	4.7	4.6	12390.9	12464.2
S	4.6	4.6	11532.9	11578.4

Table 13 Comparison between branching strategies: by demand density

Feat.	final % gap		final time (sec.)	
α	b1	b2	b1	b2
0.25	0.0	0.0	2544.7	2758.6
0.5	2.5	2.3	15289.5	15252.9
0.75	11.5	11.5	18051.5	18000.0

Table 14 Comparison between branching strategies: by α.

5.4. Final experiments

In accordance with the results of the experiments on the specific implementation details, we specialized the BBC3 algorithm by adopting heuristic h4, multicut generation o3, separation strategy s1, and branching rule b2. The resulting algorithm, called BBC3+, was tested on the 48 benchmark instances of TSPGL2. Results are shown in Table 15. The first column indicates the name of the original TSPLIB instance where digits contained in the name give the number of nodes. The second column reports the demand type. Columns three and four report the percentage gap at the root node and the corresponding computing time. Similarly, Columns five and six report the final gap and the overall computing time. A sign "*" in the last column indicates that the algorithm did not prove optimality within 5 hours of computing time.

We observe that BBC3+ was able to optimally solve 18 of the 48 instances. The largest solved instance is *dantzig42* with complete demand matrix. The hardest considered instances turned out to be *eil101* with sparse demand matrix. BBC3+ was not able to prove optimality for such an instance, but the lower bound guarantees that the best feasible solution value is within 8.2 percent of the optimal solution. The average best feasible solution value computed on the set of problems not solved to optimality is within 4.6 percent of the optimal solution in the worst case.

As mentioned in Section 5.1, the considered instance dimension range in TSPGL2 is representative of typical situations in our reference application. We observe that from such a viewpoint, BBC3+ can be also used as a very good heuristic, given the quite small optimality gaps provided on instances not solved to optimality.

6. Conclusions

We presented a new optimization problem, the Symmetric TSP with Generalized Latency problem, TSP-GL, arising in the planning activity of Semi-fexible Transit Systems, an important class of emerging transit systems. This problem can be seen as a very challenging variant of the S-TSP where both installation costs and latency are considered in the objective function. These characteristics makes the problem extremely challenging even for small problem instances.

We proposed a multicommodity flow formulation and derived a particular branch-and-cut algorithm based on Benders reformulation (BBC). This approach takes advantage of properties relating the feasible region of our problem with the S-TSP polyhedron. An extensive computational experimentation compared some variants of the proposed algorithm and, for small instances only, a

		root		final	
Name	D. type	GAP	Time	GAP	Time
att48	C	2.9	345.0	2.1	*
att48	S	4.8	133.9	3.3	*
bayg29	C	3.6	65.5	0.0	3242.0
bayg29	S	5.2	25.9	0.0	9433.5
berlin52	C	3.7	460.9	2.8	*
berlin52	S	1.6	164.3	0.5	*
burma14	C	1.5	2.2	0.0	4.7
burma14	S	1.8	1.6	0.0	4.9
dantzig42	C	3.2	125.2	0.0	14482.4
dantzig42	S	3.8	50.2	0.0	5807.9
eil51	C	6.2	498.0	5.6	*
eil51	S	3.7	151.1	2.1	*
eil76	C	5.1	3617.3	5.1	*
eil76	S	6.4	1212.1	6.3	*
eil101	C	7.8	12770.2	7.8	*
eil101	S	8.3	4250.7	8.2	*
fri26	C	1.9	24.4	0.0	103.0
fri26	S	1.9	13.4	0.0	83.6
gr17	C	1.7	3.0	0.0	16.5
gr17	S	3.3	1.6	0.0	60.3
gr21	C	0.9	7.0	0.0	8.9
gr21	S	2.0	4.1	0.0	13.8
gr24	C	1.3	19.8	0.0	55.6
gr24	S	3.3	7.7	0.0	385.8
gr48	C	6.1	437.0	5.0	*
gr48	S	5.4	139.9	3.3	*
gr96	C	6.4	8468.7	6.1	*
gr96	S	5.1	3331.6	5.0	*
kroA100	C	3.0	7173.2	2.8	*
kroA100	S	4.3	2434.0	3.9	*
kroB100	C	6.1	9642.2	6.0	*
kroB100	S	4.5	3569.5	4.3	*
kroC100	C	4.9	9061.9	4.7	*
kroC100	S	4.0	2573.2	3.7	*
kroD100	C	4.4	7486.5	4.2	*
kroD100	S	4.5	3307.8	4.4	*
pr76	C	7.1	1973.5	7.0	*
pr76	S	4.2	704.7	3.8	*
rat99	C	3.9	4708.9	3.7	*
rat99	S	8.0	1821.4	7.9	*
rd100	C	5.2	9468.4	4.9	*
rd100	S	4.8	3088.0	4.6	*
st70	C	4.7	2149.1	4.7	*
st70	S	4.1	748.1	3.4	*
ulysses16	C	2.4	5.3	0.0	91.9
ulysses16	S	1.6	3.3	0.0	24.7
ulysses22	C	1.2	15.3	0.0	103.9
ulysses22	S	3.6	4.7	0.0	1936.6

Table 15 Final experimentation on TSPLIB-derived instances, $\alpha = 0.5$

commercial solver. A large set of instances was generated for this purpose, including a set of benchmark instances derived from TSPLIB that emphasizes the difficulty of the problem addressed and, in some cases, constitute an interesting challenge for future research. These experiments provided the means to identify the best-performing variant of the algorithm we propose and to show that not only it outperforms significantly a well-known commercial solver, but that it is able to obtain good-quality solutions to realistically-sized instances within short computational times.

Among possible future research avenues, beyond investigating the interaction of the BBC with other known accelerating strategies such as Inexact Benders Cuts (Zakeri et al. 2000) and Local Branching (Rei et al. 2009), we consider particularly promising the study of alternative relaxation strategies to further tighten the current lower bounds, by exploiting the inherent integrality of the Benders subproblems. Extending the proposed methodology to other problems in the TSP and transit planning areas is also an interesting challenge.

Appendix. Proof of Propositions 1

Proposition 1. *Let us denote z_{CI} the value of the optimal linear relaxation of* (18)*,* (19)*,* (21)*,* (22) *and the following Cutset Inequalities:*

$$x(\delta(S)) \geq 1 \quad \forall S \subset V. \tag{32}$$

Then, $z_1 = z_{CI}$

Proof. We prove the proposition by showing that inequalities (23) are equivalent to (32). To this purpose, we recall that Padberg and Sung (1991) provide a characterization of the extreme rays of the node-arc cone of a directed graph, i.e., the extreme rays of Q^{hk}. In particular, they prove that the extreme rays of Q^{hk} are positive multiple of vectors $(\lambda^{hk}, \rho^{hk})$ given by (i), (ii), (iii) where

$$(i) \quad \rho_i^{hk} = 0 \quad \forall i \in N, \qquad \lambda_{ij}^{hk} = \begin{cases} 1 & for\ exactly\ one\ i\ and\ one\ j \in N \\ 0 & otherwise \end{cases}$$

$$(ii) \quad \rho_i^{hk} = \begin{cases} 1 & for\ all\ i \in S \\ 0 & otherwise \end{cases}, \qquad \lambda_{ij}^{hk} = \begin{cases} 1 & for\ all\ i \in N \setminus S,\ j \in S \\ 0 & otherwise \end{cases}$$

$$(iii) \quad \rho_i^{hk} = \begin{cases} -1 & for\ all\ i \in S \\ 0 & otherwise \end{cases}, \qquad \lambda_{ij}^{hk} = \begin{cases} 1 & for\ all\ i \in S, j \in N \setminus S \\ 0 & otherwise \end{cases}$$

and $S \subseteq N, 1 \leq |S| \leq n-1$. The substitution of (i) in (23) gives the redundant inequality $x_e \geq 0,\ \forall e \in E$. Let us consider now extreme rays of the form (ii) and the case $h \in S,\ k \in N \setminus S$. Substituting in (23) one obtains the desired inequality. If both h and k belong to S or to $N \setminus S$ we obtain redundant equations. The case (iii) is similar to (ii). Because we assumed that the demand matrix D induces a single connected component, the proposition is proved. □

Acknowledgments

While working on this project, F. Errico was postdoctoral researcher at CIRRELT and École des sciences de la gestion, Université du Québec à Montréal. T.G. Crainic was the NSERC Industrial Research Chair in Logistics Management, ESG UQAM, and Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal, and the Department of Economics and Business Administration, Molde University College, Norway, and F. Errico was postdoctoral researcher with the Chair. Funding for this project was provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grant programs, by our partners CN, Rona, Alimentation Couche-Tard, the Ministry of Transportation of Québec, and by the Fonds de recherche du Qubec - Nature et technologies (FRQNT) through its Team Research Project program. We also gratefully acknowledge the support of the Canada Foundation for Innovation, the Québec Ministry of Education, UQAM and SUN/Azuris for our computing infrastructure, as well as that of Fonds de recherche du Qubec for the CIRRELT infrastructure grants.

References

Applegate, D.L., R.E. Bixby, V. Chvàtal, W.J. Cook. 2003. Concorde: Traveling Salesman Problem Solver. `http://www.tsp.gatech.edu/concorde.html`.

Applegate, D.L., R.E. Bixby, V. Chvàtal, W.J. Cook. 2007. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press.

Atamtürk, A. 2002. On capacitated network design cut-set polyhedra. *Mathematical Programming* **92** 425–437.

Barahona, F. 1996. Network design using cut inequalities. *SIAM Journal on Optimization* **6**(3) 823–837.

Benders, J.F. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numerishe Mathematik* **4**.

Bienstock, D., O. Günlük. 1994. Capacitated Network Design – Polyhedral Structure and Computation. *INFORMS Journal on Computing* **8** 243–259.

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

Cordeau, J.-F., F. Soumis, J. Desrosiers. 2000. A Benders Decomposition Approach for the Locomotive and Car Assignment Problem. *Transportation Science* **34**(2) 133–149.

Cordeau, J.-F., F. Soumis, J. Desrosiers. 2001. Simultaneous Assignment of Locomotives and Cars to Passenger Trains. *Operations Research* **49**(4) 531–548.

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

Crainic, T. G., A. Frangioni, B. Gendron. 2001. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics* **112**(1-3) 73 – 99.

Errico, F. 2008. The design of flexible transit systems: models and solution methods. Ph.D. thesis, Politecnico di Milano, Milano, Italy.

Errico, F., T. G. Crainic, F. Malucelli, M. Nonato. 2013. A survey on planning semi-flexible transit systems: Methodological issues and a unifying framework. *Transportation Research Part C: Emerging Technologies* **36** 324 – 338.

Errico, F., T.G Crainic, F. Malucelli, M. Nonato. 2011. The Design Problem for Single-line Demand Adaptive transit Systems. CIRRELT-2011-65, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal.

Fischetti, M., G. Laporte, S. Martello. 1993. The Delivery Man Problem and Cumulative Matroids. *Operations Research* **41**(6) 1055–1064.

Günlük, O. 1999. A branch-and-cut algorithm for capacitated network design problems. *Mathematical Programming* **86** 17–39.

Gutin, G., A.P. Punnen. 2002. *The traveling salesman problem and its variations*. Springer Science & Business Media.

Holmberg, K., J. Hellstrand. 1998. Solving the Uncapacitated Network Design Problem by a Lagrangean Heuristic and Branch-and-Bound. *Operations Research* **46**(2) 247–259.

Holmberg, K., D. Yuan. 2000. A Lagrangian Heuristic Based Branch-and-Bound Approach for the Capacitated Network Design Problem. *Operations Research* **48**(3) 461–481.

Kliewer, G., L. Timajev. 2005. Relax-and-Cut for Capacitated Network Design. Gerth Brodal, Stefano Leonardi, eds., *Algorithms - ESA 2005*, *Lecture Notes in Computer Science*, vol. 3669. Springer Berlin / Heidelberg, 47–58.

Koffman, D. 2004. Operational Experiences with Flexible Transit Services. Transportation Research Board.

Leung, J. M. Y., T. L. Magnanti. 1989. Valid inequalities and facets of the capacitated plant location problem. *Mathematical Programming* **44** 271–291.

Ljubić, I., P. Putz, J.-J. Salazar-González. 2012. Exact approaches to the single-source network loading problem. *Networks* **59**(1) 89–106.

Lucena, A. 1990. Time-dependent Traveling Salesman Problem - The Deliveryman case. *Networks* **20**(6) 753–763.

Magnanti, T. L., P. Mireault, R. T. Wong. 1986. Tailoring Benders decomposition for uncapacitated network design. *Netflow at Pisa*, *Mathematical Programming Studies*, vol. 26. Springer Berlin Heidelberg, 112–154.

Magnanti, T.L., R.T. Wong. 1981. Accelerating Benders Decomposition: Algorithmic Enhancement and Model Selection Criteria. *Operations Research* **29**(3).

McDaniel, D., M. Devine. 1977. A Modified Benders' Partition Algorithm for Mixed Integer Programming. *Management Science* **24**(3) 312–319.

Méndez-Díaz, I., P. Zabala, A. Lucena. 2008. A new formulation for the Traveling Deliveryman Problem. *Discrete Appl. Math.* **156**(17) 3223–3237.

Naddef, D., S. Thienel. 2002. Efficient separation routines for the symmetric traveling salesman problem I: General tools and comb separation. *Mathematical Programming* **92** 272–255.

Naoum-Sawaya, J., S. Elhedhli. 2012. An interior-point Benders based branch-and-cut algorithm for mixed integer programs. *Annals of Operations Research* 1–2310.1007/s10479-010-0806-y.

Ortega, F., L. A. Wolsey. 2003. A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem. *Networks* **41**(3) 143–158.

Ortiz-Astorquiza, C., I. Contreras, G. Laporte. 2015. The minimum flow cost hamiltonian cycle problem: A comparison of formulations. *Discrete Applied Mathematics.* **187** 140–154.

Padberg, M, M. R. Rao. 1982. Odd minimum cut-set and *b*-matchings. *Mathematics of Operations Research* **7** 62–80.

Padberg, M., G. Rinaldi. 1990. An Efficient Algorithm for the Minimum Capacity Cut Problem. *Mathematical Programming* **47** 19–36.

Padberg, M., T. Sung. 1991. An analytical comparison of different formulations of the travelling salesman problem. *Mathematical Programming* **52**(1) 315–357.

Potts, J. F., M. A. Marshall, E. C. Crockett, J. Washington. 2010. A Guide for Planning and Operating Flexible Public Transportation Services. Transportation Research Board.

Raack, C., A. M.C.A. Koster, S. Orlowski, Roland Wessäly. 2011. On cut-based inequalities for capacitated network design polyhedra. *Networks* **57**(2) 141–156.

Rei, W., J.-F. Cordeau, M. Gendreau, P. Soriano. 2009. Accelerating Benders Decomposition by Local Branching. *INFORMS Journal on Computing* **21**(2) 333–345.

Rodríguez-Martín, I., J.-J. Salazar-González. 2008. Solving a capacitated hub location problem. *European Journal of Operational Research* **184**(2) 468 – 479.

Saharidis, G.K.D., M.G. Ierapetritou. 2010. Improving Benders decomposition using Maximum Feasible Subsystem (MFS) cut generation strategy. *Computers & chemical engineering* **34**(8) 1237–1245.

Saharidis, G.K.D., M.G Ierapetritou. 2013. Speed-up Benders decomposition using Maximum Density Cut (MDC) generation. *Annals of Operations Research* **210**(1) 101–123.

Saharidis, G.K.D., M. Minoux, M.G. Ierapetritou. 2010. Accelerating Benders method using covering cut bundle generation. *International Transactions in Operational Research* **17**(2) 221–237.

Sarubbi, J., H.P. Luna. 2007. The Multicommodity Traveling Salesman Problem. *Proceedings of International Network Optimization Conference*. Spa, Belgium.

Sarubbi, J., G. Miranda, H.P. Luna, G Mateus. 2008. A Cut-and-Branch algorithm for the Multicommodity Traveling Salesman Problem. *IEEE international conference on Service Operations and Logistics, and Informaticsi*. IEEE/SOLI, Bejing, 1806–1811.

Sellmann, M., G. Kliewer, A. Koberstein. 2002. Lagrangian Cardinality Cuts and Variable Fixing for Capacitated Network Design. *Proceedings of the 10th Annual European Symposium on Algorithms*. ESA '02, Springer-Verlag, London, UK, UK, 845–858.

Sridhar, V., J. S. Park. 2000. Benders-and-cut algorithm for fixed-charge capacitated network design problem. *European Journal of Operational Research* **125**(3) 622 – 632.

TSPGL2. 2012. The Symmetric Traveling Saslesman Problem with Generalized Latency: the TSPGL2 instance set. `http://w1.cirrelt.ca/~errico/TSPGL2.zip`.

Zakeri, G., A.B. Philpott, D.M. Ryan. 2000. Inexact cuts in Benders decomposition. *SIAM Journal on Optimization* **10**(3) 643–657.