Chapter 5

Delay-constrained Steiner Tree problem

The problem we want to deal with in this chapter is the minimum Steiner Tree with Delay constraints which has been proved to be an NP–Complete problem. In Multicast problems, indeed, one of the crucial aspects can be the Quality of Service requirement, in particular in communications not only the costs should be minimized but a time limit warranty in the reception of the forwarded messages should be considered. For this reason, we address here a Delay-constrained version of the Steiner Tree problem [51] that may find immediate application in Ad-Hoc wireless networks introducing, also in this context, the Quality of Service requirements ([37], [43]). We present several valid MIP formulations in section 5.2 comparing the respective LP relaxation (in section 5.5). In section 5.6 we describe some preprocessing procedures to reduce the size of the problems. We present exact procedures for solving the problems and some computational results in section 5.7 and 5.9 respectively.

5.1 Introduction and Related works

The Steiner Tree problem is an NP-Hard problem with a long history ([29], [41]) and in the last 20 years it has been well studied and solved ([2], [47], [57], [69]), since several practical problems can be modelled as a Steiner Tree problem. Recently some variants of the classical Steiner Tree problem have been taken into account on the influence of new problems in communications with the introduction of the Quality of Service (QoS) requirement or with the restriction on the maximum degree of the nodes (Degree-constrained Steiner Tree problem). The pure Steiner Tree problem (see Definition 1.5.1) on the graph $G = (V, E)$ is the problem of finding a tree with the minimum total cost connecting a required set of nodes R, subset of V, making possibly use of the other nodes of the graph. The Steiner Tree problem can be extended taking into account the concept of the QoS requirement. Indeed, it could be useful and appreciable in practice to guarantee the connection of a source with the nodes in R within a time limit. In particular in communication networks, messages sent by a source towards all the members of a multicast group can be required to be delivered within a maximum delay ([49], [66]). Naturally, the QoS constraints and, specifically, the maximum delay constraints impose a restriction on an acceptable multicast tree. Only recently, the Delay-constrained Steiner Tree problem has been object of study, specially, with the developments of the multimedia technology. In fact, real-time applications need to transmit information within a certain amount of time and so a message generated by one source of the network has to reach a set of target devices for delivering the same information in a fixed delay limit.

Many heuristics for solving the problem have been proposed for both static and dynamic networks ([49], [50], [79], [80]). Kompella *et al.* in [49] present greedy heuristics where they find a spanning tree of the closure graph of the constrained shortest path between the source and the required nodes, while Sriram *et al.* in [79] propose two algorithms for sparse and static communication groups divided into two phases: the first computes all the possible shortest paths from the source to each terminal respecting the maximum delay requirement and the second uses these paths for constructing the multicast tree. Zhu *et al.* [89] propose a heuristic based on a feasible search optimization method that starts with the minimum delay tree and then decrease the costs of the delay-bounded tree. An integer programming formulation together with an exact solution technique can be found in [65] by Noronha *et al.*. In Tseng *et al.* [80], a genetic algorithm and a mixed integer formulation for the Delay and Degree-constrained Broadcast problem is presented, whereas a simulated annealing method is proposed in [50] for a distributed multicast routing in Delay-constrained Steiner Tree problem. The problem of the QoS in a Minimum Energy Multicast problem in wireless Ad-Hoc networks has been already considered and mixed integer programming formulations for the QoS-MPM problem have been proposed in [37] and [43].

5.2 Mixed integer programming formulations

Let $G = (V, E)$ be an undirected graph. With each edge $e = \{i, j\} \in E$, two nonnegative real numbers are associated: the cost c_e and a delay del_e which represents the time needed to run along the edge e. The directed graph associated with $G = (V, E)$ is denoted by $G = (V, A)$, where the set A is the set of the directed arcs (i, j) and (j, i) corresponding to the undirected edge $e = \{i, j\} \in E$. We suppose that both the costs and the delays are symmetric, i.e. for every (i, j) and (j, i) in A we have $c_{(i,j)} = c_{(j,i)}$ and $del_{(i,j)} = del_{(j,i)}$. For simplifying the notation we write c_{ij} and del_{ij}

instead of $c_{(i,j)}$ and $del_{(i,j)}$, respectively. A source node s and a set R of destinations are selected among the elements of V; all the other nodes of the network (different from the source an not belonging to R) are the Steiner nodes.

The Delay–constrained Minimum Steiner Tree problem consists in finding a tree T connecting the source s with every terminal node in R (possibly making use of the Steiner nodes) with the minimum total cost $c(T)$, while respecting a fixed maximum delay $\Delta \in \mathbb{R}^+$. For each $t \in R$, if $P_{(s,t)}$ is a feasible path connecting the source s to the terminal t, then it must hold:

$$\sum_{(i,j)\in P_{(s,t)}} del_{ij} \leq \Delta.$$

Given a path $P_{(i,j)}$ from i to j, we denote by $Del(P_{(i,j)})$ the sum of the delays of the arcs of $P_{(i,j)}$:

$$Del(P_{(i,j)}) := \sum_{(k,h)\in P_{(i,j)}} del_{kh}.$$

In order to model the problem the state link variables y are introduced. For each arc $(i,j) \in A$, the boolean variable y_{ij} indicates whether or not the arc (i,j) belongs to the arborescence T connecting the source with the destinations, i.e.

$$y_{ij} := \begin{cases} 1 & \text{if } (i,j) \in T, \\ 0 & \text{otherwise.} \end{cases}$$

In the following subsections, we present four different mixed integer programming formulations for the minimum Steiner Tree problem with Delay constraints.

5.2.1 F_1: Degree-constrained Minimum Spanning Tree formulation with Delay constraints

As done in [57] for the Steiner Tree problem, the first formulation finds a Degree-constrained Minimum Spanning Tree T_0 respecting the Delay constraints on a modified network $G_0 = (V_0, A_0)$ obtained introducing another node 0 in the graph $G = (V, A)$. The set V_0 is the set of all the elements of V with the addition of the node 0, that is, $V_0 := V \cup \{0\}$ and the set A_0 is the set of all the arcs in A and of all the arcs $(0, i)$ with $i \in V \setminus R$, that is, $A_0 := A \cup \{(0, i) : i \in V \setminus R\}$. All the new directed arcs $(0, i) \in A_0 \setminus A$ have costs c_{0i} and delays del_{0i} equal to zero. On the graph $G_0 = (V_0, A_0)$, we want to find the Degree-Delay-constrained Minimum Spanning Arborescence T_0 such that the new node 0 is directly connected to the source and all the Steiner nodes $i \in V \setminus (R \cup \{s\})$ adjacent to 0 have degree 1 (i.e. if the arc $(0, i) \in T_0$, then for every (j, i) or (i, k) belonging to A neither (j, i) nor (i, k) are in the arborescence T_0) and all the required nodes are reached within the maximum time limit Δ.

Moreover, with each node of the graph $i \in V$ is associated a continuous variable t_i which represents the time when the node i is reached in the arborescence from s to each terminal in R. These variables are bounded to take positive values not greater than Δ i.e.:

$$t_i \in [0, \Delta] \qquad \forall i \in V \setminus \{s\},$$

and naturally $t_s := 0$.

The formulation, that we refer to as F_1, can be expressed as follows:

$$\min \sum_{(i,j)\in A} c_{ij} y_{ij} \tag{5.1}$$

$s.t.$

$$\sum_{(i,j)\in\delta^-(j)} y_{ij} = 1 \qquad \forall\, j \in V \tag{5.2}$$

$$\sum_{(i,j)\in\delta^+(i)} y_{ij} \geq 1 - y_{0i} \qquad \forall\, i \in R^c \tag{5.3}$$

$$y_{0j} + y_{ij} + y_{ji} \leq 1 \qquad \forall\, j \in R^c \setminus \{s\}, (j,i) \in \delta^+(j) \tag{5.4}$$

$$y_{0s} = 1 \tag{5.5}$$

$$t_i - t_j + M_{ij} y_{ij} + \alpha_{ji} y_{ji} \leq M_{ij} - del_{ij} \qquad \forall\, (i,j) \in A \tag{5.6}$$

$$0 \leq t_i \leq \Delta \qquad \forall\, i \in V \setminus \{s\} \tag{5.7}$$

$$t_s = 0 \tag{5.8}$$

$$y_{ij} \in \{0,1\} \qquad \forall\, (i,j) \in A_0. \tag{5.9}$$

Constraints (5.2) together with constraints (5.9) build a spanning arborescence rooted at 0 in G_0: in every feasible solution there is exactly one arc of the graph incoming in each node of V. Constraints (5.3) together with (5.4) are the requirements on the degree of the Steiner nodes and constraint (5.5) forces the new node 0 to be connected to the source in G_0. Finally, constraints (5.7) and (5.8) are the time limitation constraints for the time variable t_i. For each $(i,j) \in A$, M_{ij} and α_{ji} are suitable parameters that will be defined in section 5.3 where constraints (5.6) will be analysed.

5.2.2 F_2: Delay-constrained Steiner Tree formulation with directed cuts

The following formulation is a directed cut formulation for the Steiner Tree problem [87] with the addition of the delay constraints. Even in this formulation, with each node of the graph $i \in V \setminus \{s\}$ is associated a continuous variable $t_i \in [0, \Delta]$ and t_s is set to zero. We refer to the formulation as F_2:

$$\min \sum_{(ij)\in A} c_{ij} y_{ij} \tag{5.10}$$

$s.t.$

$$\sum_{(i,j)\in\delta^+(S)} y_{ij} \geq 1 \qquad \forall S \subset V, s \in S, R \cap S^c \neq \emptyset \tag{5.11}$$

$$\sum_{(j,i)\in\delta^-(i)} y_{ji} \leq \sum_{(i,j)\in\delta^+(i)} y_{ij} \qquad \forall\, i \in V \setminus (R \cup \{s\}) \tag{5.12}$$

$$y_{ij} + y_{ji} \leq 1 \qquad \forall\,(i,j) \in A \tag{5.13}$$

$$t_i - t_j + M_{ij} y_{ij} + \alpha_{ji} y_{ji} \leq M_{ij} - del_{ij} \qquad \forall\,(i,j) \in A \tag{5.14}$$

$$0 \leq t_i \leq \Delta \qquad \forall\, i \in V \setminus \{s\} \tag{5.15}$$

$$t_s = 0 \tag{5.16}$$

$$y_{ij} \in \{0,1\} \qquad \forall\,(i,j) \in A. \tag{5.17}$$

Constraints (5.11) are the directed cut constraints, for each cutset S separating the source from some required nodes in R, there should be at least one outgoing arc. The classical directed cut formulation did not consider the Flow–Balance constraints (5.12) introduced in [47] in order to strengthen the original formulation. These constraints force each Steiner node with one incoming arc to have at least one outgoing arc. Moreover, constraints (5.15), (5.16) and (5.17) are the variable domain restrictions and again constraints (5.14) will be considered in section 5.3.

5.2.3 F_3: Multicommodity Flow formulation

The following formulation F_3 is a generalization of the Multicommodity Flow formulation for the minimum Steiner Tree problem [87] including the delay constraints. For each required node $k \in R$ and arc $(i,j) \in A$, the variable x_{ij}^k takes value one if the arc (i,j) is in the directed path connecting the source to k, zero otherwise.

$$\min \sum_{(i,j)\in A} c_{ij} y_{ij} \tag{5.18}$$

$$s.t.$$

$$\sum_{(i,s)\in A} x_{is}^k - \sum_{(s,i)\in A} x_{si}^k = -1 \qquad \forall\, k \in R \tag{5.19}$$

$$\sum_{(i,j)\in A} x_{ij}^k - \sum_{(j,i)\in A} x_{ji}^k = 0 \qquad \forall k \in R,\ \forall\, j \in V \setminus \{k, s\} \tag{5.20}$$

$$\sum_{(i,k)\in A} x_{ik}^k - \sum_{(k,i)\in A} x_{ki}^k = 1 \qquad \forall\, k \in R \tag{5.21}$$

$$0 \le x_{ij}^k \le y_{ij} \qquad \forall (i,j) \in A,\ \forall\, k \in R \tag{5.22}$$

$$\sum_{(i,j)\in A} del_{ij}\, x_{ij}^k \le \Delta \qquad \forall\, k \in R \tag{5.23}$$

$$y_{ij} \in \{0,1\} \qquad \forall\, (i,j) \in A. \tag{5.24}$$

The variable x_{ij}^k represents the quantity of commodity k flowing through the arc (i,j). Constraints (5.19), (5.20) and (5.21) are the flow conservation constraints that guarantee that there is a flow of one unit outgoing from the source and incoming in each node of R. Constraints (5.23) are the delay constraints, whereas constraints (5.22) are the relation between the x and y variables.

5.2.4 F_4: Multi-cut formulation

The following formulation F_4 is a multi-cut formulation with delay constraints. Even in this formulation, we introduce variables x_{ij}^k that are defined as in formulation F_3.

$$\min \sum_{(ij)\in A} c_{ij} y_{ij} \tag{5.25}$$

$s.t.$

$$\sum_{(i,j)\in\delta^+(S)} x_{ij}^k \geq 1 \qquad \forall\, k \in R,\ \forall\, S \subset V,\ s \in S,\ k \in S^c \tag{5.26}$$

$$0 \leq x_{ij}^k \leq y_{ij} \qquad \forall\,(i,j) \in A,\ \forall\, k \in R \tag{5.27}$$

$$\sum_{(i,j)\in A} del_{ij}\, x_{ij}^k \leq \Delta \qquad \forall\, k \in R \tag{5.28}$$

$$y_{ij} \in \{0,1\} \qquad \forall\,(i,j) \in A. \tag{5.29}$$

Constraints (5.26) force the existence of an arc for each cut (S, S^c) separating the source from each element of R. The remaining constraints have the same meaning of formulation F_3: (5.28) are the delay constraints, (5.27) are the relation between the x and y variables and (5.29) are the variable domain constraints.

5.3 Cumulative-delay constraints

Constraints (5.6) and (5.14) of formulations F_1 and F_2, respectively, are at the same time subtour–elimination constraints and cumulative-delay constraints.

The classical Miller-Tucker-Zemlin constraints (MTZ, see e.g. [58]) have been introduced for providing a polynomial formulation for the Traveling Salesman problem (TSP). In our case, these constraints that include the cumulative delays can be expressed as:

$$t_i - t_j + del_{ij} \leq M_{ij}(1 - y_{ij}) \qquad \forall (i,j) \in A. \tag{5.30}$$

For these constraints, if the variable y_{ij} takes value one, then the value of t_j is forced to the value of t_i plus the delay value on the arc (i,j), if $y_{ij} = 0$, then constraints (5.30) are fulfilled just defining a sufficiently big value of M_{ij}. This value has to make the inequality $t_i - t_j \leq M_{ij} - del_{ij}$ redundant whenever $y_{ij} = 0$ and so it suffices to set $M_{ij} := \Delta + del_{ij}$.

A possible improvement that can be performed is to lift [26] the constraints (5.30) adding a nonnegative term $\alpha_{ji} y_{ji}$, with a sufficiently big value of α_{ji}, namely $\alpha_{ji} := \Delta - del_{ji}$, so that constraints (5.30) become:

$$t_i - t_j + M_{ij} y_{ij} + \alpha_{ji} y_{ji} \leq M_{ij} - del_{ij} \qquad \forall (i,j) \in A. \tag{5.31}$$

If variable $y_{ji} = 0$, then the added term does not give any contribution, if the variable y_{ji} takes value one, then $y_{ij} = 0$ in view of (5.4) or (5.13). Using the inequality (5.31) applied to the arc $(j,i) \in A$ and setting y_{ji} to 1 in (5.31), it is easy to see that the value of t_i is forced to the value of t_j plus the delay value on the arc (j,i).

5.4 Improved cumulative delay constraints

It is possible to strengthen the coefficients M_{ij} and α_{ji} of constraints (5.31) and the lower and upper bounds for (5.7) and (5.15).

The delays on the arcs can define, for every nodes $i \in V \setminus \{s\}$, a time window during which the communication should be received and forwarded by the nodes in order to respect the maximum delay Δ on the nodes of R. A message forwarded by the source s can not reach any node i of the network in a time that is lower than the shortest path value considering the delays as costs. For every node $i \in V$, we denote by $\lambda_i \in \mathbb{R}^+$ the value of the shortest path between s and i with the delays as costs: $\lambda_i = \min\{Del(P) : P \text{ is an } s{-}i \text{ path}\}$. The cumulated delay t_i at the node i should be greater than or equal to λ_i and obviously, if $\lambda_i > \Delta$ for a required node $i \in R$, the Delay-constrained Steiner Tree problem is infeasible.

Moreover, we can reduce the upper bound for the variables t_i associated with a Steiner node i. Indeed, a Steiner node i is in any feasible solution and, hence, in a feasible arborescence T only if there exists a destination $t \in R$ such that $t_i + Del(P_{(i,t)}) \leq \Delta$, where $P_{(i,t)}$ is the path from i to t in the arborescence T. For this reason, if we denote by ζ_i the value of the Shortest Path from i to the nearest destination in R with the delays as costs, the variables t_i must be at most equal to μ_i, where $\mu_i := \Delta - \zeta_i$. If $i \in R$, then obviously $\mu_i = \Delta$.

Constraints (5.7) and (5.15) become, thus:

$$\lambda_i \leq t_i \leq \mu_i \qquad \forall i \in V \setminus \{s\} \tag{5.32}$$

Naturally, these new extremes of the time window $[\lambda_i, \mu_i]$ can be used to perform a first delay-based preprocessing (see section 5.6) so that all the Steiner nodes with an empty time window can be eliminated from the graph since they will never be in a feasible solution respecting the maximum delay Δ (see Proposition 5.6.2).

Furthermore, with the introduced limitations on the values of the variables t_i, and after eliminating the Steiner nodes with an empty time window, the coefficients M_{ij} and α_{ji} of constraints (5.31) can be lowered.

Remark 5.4.1. For each $(i, j) \in A$ in constraints (5.31) the coefficients M_{ij} and α_{ji} can be set to $M_{ij} := \mu_i - \lambda_j + del_{ij}$ and $\alpha_{ji} := \mu_i - \lambda_j - del_{ji}$ respectively. Indeed, let $(i, j) \in A$. If $y_{ij} = y_{ji} = 0$, then constraint (5.31) becomes $t_i - t_j \leq M_{ij} - del_{ij}$ which is easy to see that is always fulfilled. If $y_{ij} = 1$, then in view of constraints (5.4) or (5.13), it holds that $y_{ji} = 0$ and so constraint (5.31) is:

$$t_i - t_j + M_{ij} \leq M_{ij} - del_{ij},$$

so that $t_j \geq t_i + del_{ij}$. If $y_{ji} = 1$, then $y_{ij} = 0$ and so constraint (5.31) becomes:

$$t_i - t_j + \alpha_{ji} \leq M_{ij} - del_{ij}.$$

Substituting the value of α_{ji} and M_{ij}, we have:

$$t_i - t_j + \mu_i - \lambda_j - del_{ji} \leq \mu_i - \lambda_j.$$

This last constraint with the addition of constraint (5.31) for the arc $(j, i) \in A$ force t_i to assume the value $t_j + del_{ji}$.

5.5 Comparison of LP relaxations

In this section, given a set $S \subset V$, we denote by $\delta^-_G(S)$ and by $\delta^-_{G_0}(S)$, respectively, the set of the arcs of the graph $G = (V, A)$ and of the graph

$G_0 = (V_0, A_0)$ incoming in S. In an analogous way, $\delta^+_G(S)$ and $\delta^+_{G_0}(S)$ are, respectively, the set of the arcs of the graph $G = (V, A)$ and of the graph $G_0 = (V_0, A_0)$ outgoing from S.

Proposition 5.5.1. *The value of an optimal solution of the linear relaxation of F_2 is not smaller than the value of an optimal solution of the linear relaxation of formulation F_1.*

Proof. First of all we need to augment formulation F_2 with the variables associated with the arcs $(0, i)$ with $i \in R^c$ in order to compare the optimal values of the linear relaxations of the two formulations. Given an optimal solution (y^*, t^*) for the linear relaxation of formulation F_2, as in [69], we define $\overline{y}_{ij} := y^*_{ij}$ for each $(i, j) \in A$ and for the arcs $(0, j)$ with $j \in R^c$ we set $\overline{y}_{0j} := 1 - \sum_{(i,j)\in\delta^-_G(j)} y^*_{ij}$. The solution $(\overline{y}, t^*)$ is still an optimal solution for the linear relaxation of F_2 since the costs associated with the arcs $(0, j)$ with $j \in R^c$ are zero. We should show that the augmented optimal solution $(\overline{y}, t^*)$ is a feasible solution for the linear relaxation of F_1.

Since (y^*, t^*) is an optimal solution for the linear reaxation of F_2 and the costs of the arcs are nonnegative, it follows that:

$$\sum_{(j,s)\in\delta^-_G(s)} y^*_{js} = 0.$$

Constraint (5.5) is fulfilled by $\overline{y}$, since:

$$\overline{y}_{0s} = 1 - \sum_{(j,s)\in\delta^-_G(s)} y^*_{is} = 1.$$

In [69] it is shown that the variables y^* verify the following two inequalities:

$$\sum_{(k,j)\in\delta^-_G(j),k\neq i} y^*_{kj} \geq y^*_{ji} \quad \forall j \in V \setminus \{s\},\ (j, i) \in A, \tag{5.33}$$

and

$$\sum_{(i,j)\in\delta_G^-(j)} y_{ij}^* \leq 1 \quad \forall j \in V\setminus\{s\}. \tag{5.34}$$

Let $j \in R^c\setminus\{s\}$ and $(j,i) \in \delta_{G_0}^+(j)$, in view of constraints (5.33) it follows that:

$$\overline{y}_{0j} + \overline{y}_{ij} + \overline{y}_{ji} = 1 - \sum_{(k,j)\in\delta_G^-(j)} y_{kj}^* + y_{ij}^* + y_{ji}^* = 1 - \sum_{(k,j)\in\delta_G^-(j),\ k\neq i} y_{kj}^* + y_{ji}^* \leq 1,$$

and, hence, constraints (5.4) are fulfilled.

Furthermore, let $k \in R$, in view of constraints (5.11) with $S = V\setminus\{k\}$ and of constraints (5.34), it holds that:

$$1 \leq \sum_{(i,k)\in\delta_G^+(S)} y_{ik}^* = \sum_{(i,k)\in\delta_G^-(k)} y_{ik}^* = \sum_{(i,k)\in\delta_{G_0}^-(k)} \overline{y}_{ik} \leq 1$$

and, hence, constraints (5.2) with $k \in R$ are fulfilled. Let now $k \in V\setminus R$, then

$$\sum_{(i,k)\in\delta_{G_0}^-(k)} \overline{y}_{ik} = \sum_{(i,k)\in\delta_G^-(k)} y_{ik}^* + \overline{y}_{0k} = \sum_{(i,k)\in\delta_G^-(k)} y_{ik}^* + 1 - \sum_{(i,k)\in\delta_G^-(k)} y_{ik}^* = 1.$$

Finally we have to prove that

$$\sum_{(i,j)\in\delta_{G_0}^+(i)} \overline{y}_{ij} \geq 1 - \overline{y}_{0i} \quad \forall i \in R^c.$$

Let $i \in R^c$, for the constraints (5.12) it holds:

$$\sum_{(i,j)\in\delta_{G_0}^+(i)} \overline{y}_{ij} = \sum_{(i,j)\in\delta_G^+(i)} y_{ij}^* \geq \sum_{(j,i)\in\delta_G^-(i)} y_{ji}^* = 1 - \overline{y}_{0i}.$$

The other constraints of formulation F_1 are obviously verified and, therefore, $(\overline{y}, t^*)$ is feasible for the linear relaxation of F_1. $\square$

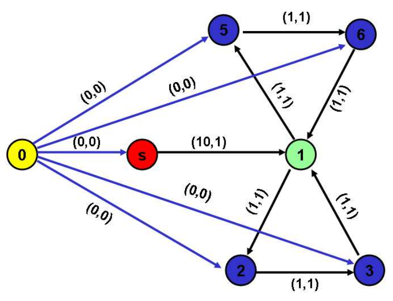

Figure 5.1: Example of an optimal solution of the linear relaxation of F_1 which is infeasible for the linear relaxation of F_2

The example used in [69] and reported in Figure 5.1, can be used to show that there exist Delay-constrained Steiner Tree problems in which the optimal solution of the linear relaxation of F_1 is not feasible for the linear realaxation of F_2.

Example 5.5.1. Consider the graph in Figure 5.1, where $R = \{1\}$ and $\Delta = 10$. The delay constraints in this case are redundant for defining any optimal solution. The solution in the variables y: $y_{02} = y_{03} = y_{15} = y_{56} = y_{61} = \frac{1}{3}$, $y_{12} = y_{23} = y_{31} = y_{05} = y_{06} = \frac{2}{3}$, $y_{0s} = 1$ is an optimal solution for the linear relaxation of the formulation of F_1, but it is not a feasible solution for the linear relaxation of the formulation F_2, since if $S = \{s\}$, then $\sum_{(i,j)\in\delta^+(S)} y_{ij} = 0$.

Proposition 5.5.2. *Formulation F_3 is better than formulation F_4.*

Proof. We have to prove that every feasible solution for the linear relaxation of formulation F_3 is feasible for the linear relaxation of F_4 and that there exist Delay-constrained Steiner Tree problems in which a feasible solution for the linear relaxation of F_4 is not feasible for the linear relaxation of F_3 (see Definition 1.1.4). Let (y^*, x^*) be a feasible solution for the linear relaxation of formulation F_3. The only constraints that should be checked are constraints (5.26). Let $k \in R$, $S \subset V$ such that $s \in S$ and $k \in S^c$, the value of the flow from s to k is 1, so that:

$$\sum_{(i,j)\in\delta^+(S)} x_{ij}^k - \sum_{(i,j)\in\delta^-(S)} x_{ij}^k = 1;$$

thus

$$\sum_{(i,j)\in\delta^+(S)} x_{ij}^k = 1 + \sum_{(i,j)\in\delta^-(S)} x_{ij}^k \geq 1.$$

A case in which a feasible solution for the linear relaxation of F_4 is not feasible for F_3 in given in the Example 5.5.2. □

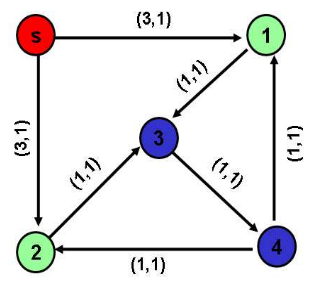

Figure 5.2: Example of a feasible solution for the linear relaxation of F_4 that is not feasible for the linear relaxation of F_3

Example 5.5.2. Consider the graph in Figure 5.2 which is another example proposed in [69]. Suppose $R := \{1, 2\}$ and $\Delta := 10$. The solution $y_{s1} = y_{s2} = y_{13} = y_{23} = y_{34} = y_{41} = y_{42} = x_{s1}^1 = x_{s2}^1 = x_{13}^1 = x_{23}^1 = x_{34}^1 = x_{41}^1 = x_{42}^1 = x_{s1}^2 = x_{s2}^2 = x_{13}^2 = x_{23}^2 = x_{34}^2 = x_{41}^2 = x_{42}^2 = \frac{1}{2}$, is a feasible solution for the linear relaxation of the formulation of F_4, but it is not feasible for the linear relaxation of the formulation F_3 since the constraint (5.20) relative to node 3 is not fulfilled.

5.6 Preprocessing

Preprocessing plays a very useful role in solving combinatorial and integer programming problems. This technique, indeed, reduces the size of the problems by means of logical implications, producing equivalent problems. The preprocessing performed in our problem is based on the fulfilment of the time windows request and on an adaptation of the known preprocessing techniques (see Proposition 1.5.1) used to reduce the size of the graph in the pure Steiner Tree problem; because of the presence of the delay on the arcs, if we want to contract certain edges, we need to store the delays. For this reason, we introduce $m_i \in \mathbb{R}$ for each $i \in V$ and initially we set m_i to zero for all $i \in V$.

5.6.1 Degree-delay preprocessing

Until no more reduction can be performed in the graph, the following tests for reducing the size of the problem are executed:

Proposition 5.6.1 (Degree one test). *For every node $i \in V$*

(i) *if i is a Steiner node and $|\delta(i)| = 1$, then i is eliminated from the graph together with the edge incident in i;*

(ii) *if $|\delta(i)| = 1$, $i \in R$ and $\delta(i) = \{\{i, j\}\}$, then $\{i, j\}$ is contracted, the cost c_{ij} is stored to be added to the optimal solution and if $del_{ij} > m_j$, the values of μ_j and m_j are updated: $\mu_j := \mu_i + m_j - del_{ij}$ and $m_j := del_{ij}$, respectively.*

For every node $i \in V$, the time windows $[\lambda_i, \mu_i]$ is empty if the time required to reach the node i from the source s is greater than the residual time to reach the nearest (in terms of delays) required node.

Proposition 5.6.2 (Non–empty time windows). *For every node $i \in V$*

(i) *If $\lambda_i > \mu_i$ and i is a Steiner node, then i can be removed from the graph together with all its incident edges.*

(ii) *If $\lambda_i > \mu_i$ and i is a required node, then the Minimum Steiner Tree problem with the delay constraints is infeasible.*

Proof. (i) Suppose on the contrary that an optimal solution contains a Steiner node i with $\lambda_i > \mu_i$ and let t be the nearest terminal from i using the delays as cost. As i is a node belonging to the optimal solution, there exists on the support of this solution at least a path $P_{s,\bar{t}}$ from the source to a terminal $\bar{t} \in R$ passing through i. The total delay along the path $P_{(s,\bar{t})}$ is such that:

$$Del(P_{(s,\bar{t})}) = \sum_{(i,j)\in P_{(s,\bar{t})}} del_{ij} \geq \lambda_i + Del(P_{(i,\bar{t})}) \geq \lambda_i + \zeta_i \geq \lambda_i - \mu_i + \Delta > \Delta,$$

which is a contradiction.

(i) Suppose on the contrary that there exists a feasible solution, since the shortest path value from the source to the required node i with the delays as costs is greater than Δ for each path $P_{(s,i)}$ in the graph it holds: $Del(P_{(s,i)}) \geq \lambda_i > \Delta$ which is a contradiction.

□

Proposition 5.6.3 (Adjacent time request). *For every edge $\{i, j\} \in E$, if $\lambda_i + del_{ij} > \mu_j$ and $\lambda_j + del_{ji} > \mu_i$, then the edge $\{i, j\}$ can be eliminated from the graph.*

Proof. Suppose on the contrary that an optimal solution contains the arc (i,j) with $\lambda_i + del_{ij} > \mu_j$ (the same holds for (j,i) with $\lambda_j + del_{ji} > \mu_i$). Let t be the nearest required node from j using the delays as costs. As j is a node belonging to the optimal solution, there exists on the support of this solution at least a path $P_{(s,\bar{t})}$ from the source to a terminal $\bar{t}$ passing through j. The total delay along the path $P_{(s,\bar{t})}$ is such that

$$\begin{aligned} Del(P_{(s,\bar{t})}) = \sum_{(i,j)\in P_{s,\bar{t}}} del_{ij} &\geq \lambda_i + del_{ij} + Del(P_{(j,\bar{t})}) \geq \lambda_i + del_{ij} + \zeta_i \\ &\geq \lambda_i + del_{ij} - \mu_i + \Delta > \Delta, \end{aligned}$$

which is a contradiction. □

The degree two test is analogous to the test of the Steiner Tree problem (see Proposition 1.5.1), but a further condition on the delays must be inserted in order to respect the maximum delay at the required nodes.

Proposition 5.6.4 (Degree two test). *If $i \in V$ is a Steiner node with $\delta(i) = \{\{i,k\},\{j,i\}\}$,*

(i) *if $\{k,j\} \notin E$, then the edges $\{k,i\}$ and $\{i,j\}$ are substituted by a new edge $\{k,j\}$ with cost $c_{kj} = c_{ki} + c_{ij}$ and delay $del_{kj} = del_{ki} + del_{ij}$ and i can be eliminated.*

(ii) *if $\{k,j\} \in E$, if $c_{ki} + c_{ij} > c_{kj}$ and $del_{ki} + del_{ij} > del_{kj}$, then i can be eliminated from the graph together with the edges $\{i,k\}$ and $\{j,i\}$.*

(ii) *if $\{k,j\} \in E$, $c_{ki} + c_{ij} \leq c_{kj}$ and $del_{ki} + del_{ij} \leq del_{kj}$, then node i is removed from the graph together with its incident edges and the edge $\{k,j\}$ is given the cost $c_{kj} = c_{ki} + c_{ij}$ and the delay $del_{kj} = del_{ki} + del_{ij}$.*

All the formulations are defined on directed graphs, so that, another reduction can be done considering the orientation of the arcs.

Proposition 5.6.5 (Direct arcs test). *Every arc incoming in the source $(i, s) \in A$ and all the directed arcs (i, j) such that $\lambda_i + del_{ij} > \mu_j$ can be eliminated from the directed graph.*

Proof. Because of the nonnegativity of the costs all the arcs $(i, s) \in A$ can be removed from the graph and the rest follows as in Proposition 5.6.3. □

The delay-degree preprocessing consists in summary in these steps:

Step 1: Degree one test;

Step 2: Non–empty time windows test;

Step 3: Adjacent time request test;

Step 4: Degree two test;

Step 5: If at least one contraction or elimination has been executed go to Step 1 else go to Step 6;

Step 6: Consider the directed graph and perform the Direct arc elimination.

5.6.2 LP preprocessing

The LP preprocessing is based on Proposition 1.5.2, in fact, if we denote by z_{LP} the optimal value of the linear relaxation of the problem and by z_{UB} the value of the best known feasible solution of the problem, that is an upper bound for the solution, then Proposition 1.5.2 can be applied to fix the value of certain nonbasic variables.

If y^* is an optimal solution of the linear relaxation of the problem, then if $y^*_{ij} = 0$ its reduced cost $\overline{c}_{ij}$ is nonnegative. Using Proposition 1.5.2, if

$z_{LP} + \overline{c}_{ij} > z_{UB}$, then fixing the variable y^*_{ij} to one does not produce any improvement in the optimal value of the objective function, hence, the value of the variable y^*_{ij} is fixed to zero, which means that it is possible to eliminate the arc (i, j) from the graph.

Moreover, if $y^*_{ij} = 1$ in the optimal solution, the reduced cost $\overline{c}_{ij}$ is nonpositive and, using again Proposition 1.5.2, if $z_{LP} - \overline{c}_{ij} > z_{UB}$, then even in this case reducing to zero the value of y^*_{ij} does not make any improvement and so the variable y^*_{ij} is fixed to take value 1, thus, the arc (i, j) is always in an optimal solution of the IP problem.

5.7 Exact Solution strategies

In this section, we present the methods for solving the different formulations presented in section 5.2.

5.7.1 Algorithm for F_1

The Degree-constrained Minimum Spanning Tree formulation with Delay constraints has a polynomial number of constraints and can be directly solved by any mixed integer linear programming solver. The algorithm for its solution can be summarized as follows:

Step 0: Perform the Degree-delay preprocessing;

Step 1: Solve the linear relaxation of formulation F_1;

Step 2: Perform the LP preprocessing; if an edge is eliminated go to Step 0 else go to Step 3;

Step 3: Solve the MIP formulation F_1.

5.7.2 Algorithm for F_2

The drawback of formulation F_2 is represented by the sets of constraints (5.11) that are in an exponential number, but since only a small fraction of these constraints is saturated at optimality, we choose to solve the problem with an iterative approach. Namely, the initial constraint matrix is constituted by the constraints (5.12) and (5.13), by the Delay-constraints (5.14), by constraints (5.15) and (5.16) and by the cuts (5.11) generated by the subset $S := \{s\}$ and by the subsets S such that $|S^c| = 1$. For speeding the generation of constraints (5.11) up, we solve the linear relaxation of formulation F_1 with all the costs equal to 1, whose optimum value is indicated by β. An approximation of the minimum number of arcs in the solution of the Delay-constrained problem is given by $\lceil \beta \rceil$, hence we add to the initial constraint system the inequality:

$$\sum_{(i,j) \in A} y_{ij} \geq \lceil \beta \rceil \, . \tag{5.35}$$

The procedure of the algorithm can be formalized as follows:

Step 0: Perform the Degree-delay preprocessing;

Step 1: Let F_2' be the formulation F_2 with only the constraints (5.11) corresponding to $S = \{s\}$, and S such that $|S^c| = 1$ and including the new constraint (5.35);

Step 2: Solve F_2', and let $(\overline{y}, \overline{t})$ be the optimal solution;

Step 3: If $\overline{y}$ violates a constraint ctr_{12} of type (5.11) (the separation routine will be described later), then add ctr_{12} to F_2' and go to Step 2;

Step 4: Perform the LP preprocessing, if an edge is eliminated, then go to Step 0 else go to Step 5;

Step 5: Solve the MIP problem; let $(\overline{y}, \overline{t})$ be the optimal solution of F_2';

Step 3: If $\overline{y}$ violates a constraint ctr_{12} of type (5.11), then add ctr_{12} to F_2' and go to Step 5 otherwise the optimal solution has been found.

Notice that the procedure converges after a limited number of iterations.

Separation problem

Once a solution $(\overline{y}, \overline{t})$ of F_2' is available, the presence of violated inequalities of type (5.11) of F_2 not inserted into F_2' can be detected as follows. For each source-destination pair the maximum flow problem with y as capacities is solved. If a maximum flow value is less than 1, then the minimum capacity cut (S, S^c) is indentified and the corresponding constraint (5.11) is generated.

5.7.3 Algorithm for F_3

The Multicommodity Flow formulation F_3 may have a large number of variables but it does not have critical constraints that impose the use of a specific solution technique. Formulation F_3 is, thus, directly solved by any mixed integer linear programming solver. The pseudocode of the solution algorithm for F_3 is the same as in subsection 5.7.1.

5.7.4 Algorithm for F_4

An iterative approach is used for solving the problem with formulation F_4.

The initial constraint matrix is constituted by the constraints of F_4 except constraints (5.26), indeed, among constraints (5.26) only those generated by the subset $S := \{s\}$ and by the subsets S such that $|S^c| = 1$ are considered initially. For speeding the generation of constraints (5.26) up, we have solved the shortest path problem connecting the source to each destination $k \in R$ and we have computed β_k which represents the number of arcs of each s-k path. In order to make the generation of constraints (5.26) faster, for each node $k \in R$ we add to the initial constraint system the inequalities:

$$\sum_{(i,j)\in A} x_{ij}^k \geq \beta_k \qquad \forall k \in R. \tag{5.36}$$

The algorithm is the same as for formulation F_2 (see subsection 5.7.2), the only difference is in the separation procedure. Indeed, when a cut (S, S^c) is found, then all the constraints (5.26) for each $k \in S^c$ are generated at the same time, instead of the unique constraint generated for F_2.

5.8 Heuristic Solution

In order to make the LP preprocessing effective, a good heuristic that provides a feasible solution with a tight upper bound z_{UB} in a reasonable time should be considered. We compute the shortest paths that fulfil the delay constraints between the source and all required nodes and we select the path $\overline{P}(s, \overline{t})$ with the highest length. Till all the required nodes are connected to the source, at each step, the heuristic H_1 adds a new path that fulfils the maximum delay constraint with the lowest total cost from one of the nodes of the current tree (initially constituted by $\overline{P}(s,\bar{t})$) to one of the required nodes not yet connected to the source. This heuristic is fast but does not provide a tight upper bound. For the sake of reducing the gap between the value of the optimal integer solution and z_{UB}, we propose the heuristic H_1' in which we repeat K times the following procedure: we perturb the costs associated with the arcs, we perform the heuristic procedure H_1 and we consider the best obtained value z_{UB}.

The problem of finding the Shortest Path with capacity constraints has been proved to be NP-Hard in [38]. This type of problem has been widely studied and the case in which the capacity constraints are the delay constraints has been considered in [39]. The Delay-constrained Shortest Path problem can be solved in an exact way with a dynamic approach based on a generalization of Ford-Fulkerson and of Dijkstra algorithms ([44], [59]). An exact solution based on the Lagrangian relaxation has been proposed in [38]. Since we aim at using an efficient and fast heuristic, like in [59], we find an approximate solution of the Lagrangian relaxation of the Delay-constrained Shortest Path problem where the delay constraints are relaxed so that the relaxed problem can be solved by Dijkstra's algorithm.

5.8.1 Heuristic H_1

Given the graph $G = (V, A)$, we indicate by C the set of the required nodes connected to the source. All the Delay-constrained Shortest Paths $\overline{P}_{(s,t)}$ that connect the source to each $t \in R$ are computed, and it is selected the path $\overline{P}(s,\bar{t})$ with the greatest cost (length) whose cost is assigned to z_{UB}. The set C becomes, thus, $C := \{\bar{t}\}$ and we assign a zero cost to all the arcs of the path $\overline{P}_{(s,\bar{t})}$. Unless the set C coincides with R, we add a new node f to the graph G and we define the set A' of all the arcs of A with the addition of all the arcs (i, f) for each $i \in R \setminus C$, whose is associated a zero cost and a zero delay (the current graph is, thus, $G' = (V', A')$ with $V' = V \cup \{f\}$ and $A' = A \cup \{(i, f) : \forall i \in V \setminus C\}$); we solve the Delay-constrained Shortest Path problem between the source and the node f finding the path $P_{(s,f)}$; we update C adding the required node t such that $(t, f) \in P_{(s,f)}$ and we set to zero the costs of the arcs of $P_{(s,f)}$ that belong to A; finally we update the value z_{UB} adding the cost of the path $P_{(s,f)}$, that is, $z_{UB} := z_{UB} + c(P_{(s,f)})$ and we repeat the process. If C coincides with R the current value z_{UB} is the required upper bound.

The algorithm can be summarized as follows:

(Step 0:) Set $C := \emptyset$.

(Step 1:) Compute the approximated Delay-constrained Shortest Paths between the source and all the required nodes. Select $\overline{P}_{(s,\bar{t})}$ the path with the maximum cost (length).

(Step 2:) Set $z_{UB} := c(\overline{P}_{(s,\bar{t})})$, $C := C \cup \{\bar{t}\}$ and $c_{ij} := 0$ for all $(i, j) \in \overline{P}_{(s,\bar{t})}$.

(Step 3:) Add a node f to the graph $G = (V, A)$; define $V' = V \cup \{f\}$ and $A' := A \cup \{(i, f) : \forall i \in R \setminus C\}$; set $c_{if} = del_{if} = 0$ for all $(i, f) \in A'$.

(Step 4:) Compute the approximated Delay-constrained Shortest Path $P_{(s,f)}$ on the graph $G' = (V'A')$, find $\bar{t} \in R$ such that $(\bar{t}, f) \in P_{(s,f)}$.

(Step 5:) Set $z_{UB} := z_{UB} + c(P_{(s,f)})$, $C := C \cup \{\bar{t}\}$ and $c_{ij} := 0$ for all $(i, j) \in P_{(s,f)} \cap A$. If $C \subset R$, then go to step 3 else Stop.

5.8.2 Heuristic H_1'

In this heuristic, we perturb the cost associated with each arc (i, j) of the graph, that is, we generate a random number ϵ_{ij} in the interval $[0.5, 1.5]$ and we assign to the arc (i, j) the cost $\epsilon_{ij}c_{ij}$; we solve the problem of finding a feasible solution for the Delay constrained Steiner Tree problem with the perturbed costs with the procedure H_1 and we store the best obtained value of z_{UB}. We have seen on the basis of the experimental results that we can find the best gap between the optimal value of the Delay constrained Steiner Tree problem and the value z_{UB}, if we perturb the costs and solve the problem for $K = 500$ times.

5.9 Computational results

All the instances of the Delay-constrained Steiner Tree problem has been solved on an Opteron 246 machine with 2 GB RAM memory using the version 9.1 of Cplex as solver. We have set to 30 minutes the computational time limit. By NS we indicate the number of instances not solved within the time limit when the solution process is interrupted.

5.9.1 Description of the problem instances

To the best of our knowledge, no benchmark is available for the Delay constrained Steiner Tree problem in literature. We have, therefore, considered the problems proposed in the SteinLib library [48] for the pure Steiner Tree problem, in particular the problems of the class B and the first 10 instances of the class C. The instances of class B and C are randomly generated sparse graphs with edge weights between 1 and 10; for the class B, the size of the problems goes from graphs with $|V| = 50, |R| = 9, |E| = 63$ to graphs with $|V| = 100, |R| = 50, |E| = 200$, whereas for the considered instances of the class C the size of the problems goes from $|V| = 500, |R| = 5, |E| = 625$ to $|V| = 500, |R| = 250, |E| = 1000$. For the classical Steiner Tree problem these instances can be solved in few seconds with the local preprocessing or by efficient known algorithms. We have generated randomly the delays on the edges in such a way that they result correlated and non-correlated to the costs. In the first case a random number r is generated in the interval $[0.8, 1.2]$ and for each edge $\{i, j\}$ we set $del_{ij} = r * c_{ij}$, in the second case the delays are simply random values belonging to the interval $[1, 100]$. On the basis of the generated delays, we have computed the value MP which is the maximum among the shortest paths with the delays as costs between the source and each required node, then in the problems indicated with 0.1 we have set Δ to the value $\Delta := 1.1 * MP$ and in the problems indicated with 0.5 we have set Δ to $\Delta := 1.5 * MP$. With these choices none of the problems is infeasible. In the following tables, we indicate for example by B Ran 0.1 the set of the instances of the class B with delays non-correlated with the costs and with $\Delta = 1.1 * MP$ and with C Cor 0.5 the set of the instances of the class C with delays correlated with the costs and with $\Delta = 1.5 * MP$.

In columns *Gap*, we report the mean of the ratios $(OPT - LP)/OPT$ where OPT is the optimum value of the integer problem and LP is the optimum value of the linear relaxation of the problem. For each class of problems, we indicate with T the mean of the resolution times in seconds for the instances solved within the time limit and with T_{max} the maximum computational time. If certain instances in a class are not solved within 30 minutes, then T_{max} reports the number of not solved problems.

5.9.2 Performance of the different formulations

In Table 5.1, we report the gap between the value of the optimal integer solution of the instances and the value of an upper bound provided by the heuristic H_1'; *gap* is, indeed, the mean of the values $(z_{UB} - OPT)/OPT$.

Table 5.1: Gap for the heuristic H_1'

Problem	$gap \times 100$	*Problem*	$gap \times 100$
B Ran 0.1	1.28	C Ran 0.1	3.06
B Ran 0.5	0.66	C Ran 0.5	1.24
B Cor 0.1	0.28	C Cor 0.1	2.26
B Cor 0.5	0.20	C Cor 0.5	2.01

We use the value z_{UB} of the heuristic H_1' to perform the LP proprocessing of the problem.

In Table 5.2, we present the average gap and the computational time for the different algorithms of section 5.7. All the instances of the class B have been solved within the required time limit, whereas there are certain instances of the class C that are unsolved. Formulation F_1 is the fastest among all the other formulations even if the optimal value of the linear relaxation of the problems are always the worst with respect to the lower bounds provided by the other formulations. Moreover, F_3 is the formulation with the closest optimal value of the linear relaxation to the optimal integer value, but for example 6 over the 10 instances of the different problems of the class C are not solved in the time limit.

Regarding to formulations F_2 and F_4, one provides a better gap (F_4), but the other solve the problems in a lower time (F_2), but just using the MIP solver for solving the instances none of the two's has interesting behaviours if compared with F_1 and F_3.

Table 5.2: Average gap and computational times for the Delay-constrained Steiner Tree problem

	F_1			F_2		
Problem	$Gap \times 100$	T	T_{max}	$Gap \times 100$	T	T_{max}
B Ran 0.1	9.5	0.14	1.35	7.59	6.75	114.06
B Ran 0.5	6.54	0.73	3.81	3.85	4.77	45.24
B Cor 0.1	5.44	0.14	0.71	2.83	0.82	5.11
B Cor 0.5	3.81	0.42	2.14	1.02	1.55	18.17
C Ran 0.1	7.30	196.00	2NS	5.45	121.23	5NS
C Ran 0.5	5.84	82.52	3NS	2.72	38.89	5NS
C Cor 0.1	6.28	210.50	2NS	2.40	22.25	5NS
C Cor 0.5	2.47	94.73	1NS	0.04	88.80	6NS

	F_3			F_4		
Problem	$Gap \times 100$	T	T_{max}	$Gap \times 100$	T	T_{max}
B Ran 0.1	2.11	34.61	480.88	2.32	38.82	1106.0
B Ran 0.5	1.82	44.21	637.26	2.33	212.14	1270.8
B Cor 0.1	1.54	7.35	103.40	1.76	45.94	319.90
B Cor 0.5	0.72	3.22	44.03	0.74	47.96	359.88
C Ran 0.1	4.94	0.76	6NS	4.97	0.90	6NS
C Ran 0.5	3.74	1.75	6NS	1.75	6.31	6NS
C Cor 0.1	1.84	0.50	6NS	1.92	0.72	6NS
C Cor 0.5	0.00	0.87	6NS	0.00	0.53	6NS

5.9.3 Assessment of the different components

In this section, we highlight certain of the contributions of the different components that influence the solution of the instances. In particular, we report the percentage of reduction of the degree-delay preprocessing (see section 5.6.1), the gap and the computational time of all the algorithms in which the LP preprocessing has not been executed and the computational comparison of the usage of the lifted constraints (5.31) and of the unlifted constraints (5.30) for formulation F_1 and F_2.

Table 5.3: Gap and computational times for the algorithm without the LP preprocessing

	F_1			F_2		
Problem	$Gap \times 100$	T	T_{max}	$Gap \times 100$	T	T_{max}
B Ran 0.1	9.44	0.13	1.30	7.54	5.94	99.31
B Ran 0.5	6.37	0.77	2.34	3.83	4.46	43.22
B Cor 0.1	5.27	0.21	1.27	2.79	1.97	15.23
B Cor 0.5	3.52	0.45	2.69	0.97	0.83	17.52
C Ran 0.1	7.60	189.90	2NS	5.44	68.02	5NS
C Ran 0.5	5.84	82.52	3NS	2.48	127.14	4NS
C Cor 0.1	6.28	210.50	2NS	2.39	22.25	5NS
C Cor 0.5	2.46	94.73	1NS	0.04	88.80	6NS

	F_3			F_4		
Problem	$Gap \times 100$	T	T_{max}	$Gap \times 100$	T	T_{max}
B Ran 0.1	4.02	27.47	24.46	4.66	163.9	1210.2
B Ran 0.5	2.28	43.55	549.3	2.33	212.1	1270.8
B Cor 0.1	1.45	4.62	36.14	1.76	45.94	319.90
B Cor 0.5	0.64	2.72	21.18	0.74	47.96	359.88
C Ran 0.1	4.93	1.14	6NS	4.97	4.37	6NS
C Ran 0.5	1.39	315.30	5NS	2.04	7.57	6NS
C Cor 0.1	1.87	173.00	5NS	2.09	1.68	6NS
C Cor 0.5	0.47	0.54	6NS	0.00	0.55	6NS

In Table 5.4, we present the mean percentage of reduction of the num-

Table 5.4: Degree-delay preprocessing reduction

Problem	$\%n$	$\%m$	$\%arc$
B Ran 0.1	45.85	15.50	49.97
B Ran 0.5	38.32	13.87	31.99
B Cor 0.1	46.98	15.48	47.50
B Cor 0.5	34.06	12.25	28.78
C Ran 0.1	61.94	14.09	59.51
C Ran 0.5	51.32	12.65	45.06
C Cor 0.1	60.08	12.81	56.00
C Cor 0.5	51.94	12.65	45.49

ber of nodes, destinations and arcs performed only using the degree-delay preprocessing (the LP preprocessing is not performed in this case) for the different instances we have generated. If n is the original number of nodes and n' the number of nodes in the reduced problem in column $\%n$ we report the mean of the percentage of the values $(n-n')/n$ over all the instances belonging to the same class of problems (similarly for column $\%m$ and $\%arcs$). The number of nodes is almost halved and there is a consistent reduction on the number of arcs, the reduction is more effective on the class C than on the class B and the effect of the preprocessing based on the delay can be noticed in the higher percentage of reduction of the size of the problem when Δ is only ten percent more than the value that make the problem feasible ($\Delta = 1.1 * MP$). When only the delay-degree preprocessing is performed to reduce the size of the problem, the relations among the formulations in terms of gap and computational time do not change as it is easy to see in Table 5.3. In most of the problems the gap is slightly reduced. We have not reported here another table to show the solution time of the different algorithms on the original graph (that is on the graph where no preprocess-

Table 5.5: Improvement of the lifted constraints (5.31) with respect to the unlifted constraints (5.30)

		(5.31)		*(5.30)*	
Problem		$Gap \times 100$	T	$Gap \times 100$	T
B Ran 0.1	F_1	9.44	0.12	14.78	0.12
B Ran 0.5	F_1	6.37	0.77	13.60	1.06
B Cor 0.1	F_1	5.28	0.21	12.84	0.14
B Cor 0.5	F_1	3.52	0.45	12.12	0.47
B Ran 0.1	F_2	7.54	5.94	7.85	10.31
B Ran 0.5	F_2	3.83	4.46	3.93	14.73
B Cor 0.1	F_2	2.79	1.98	2.92	2.76
B Cor 0.5	F_2	0.97	1.77	0.98	2.57

ing is performed), because even some of the instances of the class B are not solved within the time limit.

In Table (5.5), we compare the impact of the lifted constraints (5.31) with respect to the unlifted constraints (5.30). For formulation F_1, the usage of constraints (5.31) strongly reduces the value of *Gap*, but they do not improve the solution time, whereas for formulation F_2 constraints (5.31) reduce the computational time, but do not decrease significantly the *Gap*.

Table 5.6: Computational time of the class B and C for the Steiner Tree problem

Problem	T	T_{max}
B	0.47	2.68
C	104.14	827.53

Finally, taking a sufficiently big value of Δ the delay constraints become redundant for the optimal solution and in this case we have solved the Steiner Tree problem on the graph reduced by using only the degree preprocessing; the mean and the maximal computational time for the class B and C with formulation F_1 are reported in Table 5.6. All the instances of the Steiner Tree problem are solved within the time limit.

5.10 Conclusions

In this chapter, in order to guarantee a Quality of Service in the communications, we have considered the Delay-constrained Steiner Tree problem. We have proposed four different formulations for modelling the problem together with a preprocessing based on the degree-delay characteristics and on the reduced costs properties, for reducing the size of the problems. The computational results, we have provided, suggest the usage of different techniques for solving those problems that are not solved so far. Another interesting problem to deal with is to apply the delay constraints to the wireless Ad-Hoc networks.