A Lagrangian approach for the Minimum Spanning Tree Problem with Conflicting Edge Pairs

Francesco Carrabs*[1] and Manlio Gaudioso†[2]

[1]Dipartimento di Matematica, Universitá di Salerno, Italia
[2]Dipartimento di Ingegneria Imformatica, Modellistica, Elettronica e Sistemistica, Universitá della Calabria, Italia

Abstract

This paper addresses the Minimum Spanning Tree Problem with Conflicting Edge Pairs, a variant of the classical Minimum Spanning Tree where, given a list of conflicting edges, the goal is to find the cheapest spanning tree with no edges in conflict. We adopt a Lagrangian relaxation approach together with a dual ascent and a subgradient procedure to find tight lower bounds on the optimal solution. The algorithm is also equipped with a heuristics approach which provides an upper bound by removing the conflicts from possible infeasible solutions met during the calculation of the lower bounds. The computational results, carried out on benchmark instances, show that the proposed algorithm finds the optimal solutions on several instances.

*fcarrabs@unisa.it (Corresponding author)
†manlio.gaudioso@unical.it

Moreover, the lower bounds it provides are much more accurate than ones provided by other Lagrangian approaches available in the literature and they are computed is much less time.

Keywords: Lagrangian approach; Conflict Constraints; Conflicting Edges; Spanning Tree;

1. Introduction

In this paper, we study the Minimum Spanning Tree Problem with Conflicting Edge Pairs (MSTC), a NP-Hard variant of the classical Minimum Spanning Tree problem. Given a graph G and a set of edge pairs in conflict, MSTC problem consists of finding the minimum spanning tree of G containing no conflicting edges.

The MSTC problem has a number of practical applications. For instance, it arises in the design of offshore wind farm networks [3]. In this context, it is often required to build spanning trees to connect a set of turbines. These connections should be carried out in the cheapest way and by avoiding the overlap of the cables [18]. By considering two overlapped cables as a conflicting pair, the problem is just MSTC. Another application concerns the crossing of road maps where some movements are forbidden [17]. For instances, in some point of the map can be forbidden to turn left or right and this constraint can be simulated by using conflict edges. Other applications concern the resolution of quadratic bottleneck spanning problem [26] and the installation of an oil pipeline system connecting various countries [8].

In perspective it would be applied to freight transportation network design, to model situations where single transportation *legs*, which are under control of the same shipper, cannot be simultaneously activated due to possible scarcity of resources.

The problem was introduced by Darmann et al. [8] [9]. In these works, the authors proved that MSTC is NP-Hard and that in some particular cases it can be solved in polynomial time. In [25] the authors presented a branch-and-cut approach based on two sets of valid inequalities. Moreover, they introduced a preprocessing technique to possibly reduce the instance size. Very recently, a branch-and-cut approach was introduced in [5], based on a new set of valid inequalities. It outperforms previous branch-and-cut methods. Several heuristics were introduced in [26], together with two exact algorithms based on Lagrangian relaxation. Since our approach too is Lagrangian, we refer to the results of such relaxation to evaluate the quality of the lower bounds produced by our algorithm. Although the Lagrangian relaxation we adopt is the same as in [26], we focus here on an *ad hoc* dual ascent procedure to tackle the Lagrangian dual, instead of a classic subgradient method as in [26]. Finally, a multi ethnic genetic algorithm proposed in [3] uses a specific fitness function aimed at minimizing the number of conflicts in case no zero-conflict solution exists. In addition, three local search procedures are applied to chromosomes of the population to improve the quality of the solutions. Since this heuristic performs best among those available in the literature, we will use it for comparison purposes in the sequel of the paper.

MSTC problem belongs to the class of optimization problems with conflict constraints. In the literature there exist several optimization models dealing with conflicting decisions. We recall here, among the others, the knapsack problem with conflict constraints [20], the bin packing problem with conflicts [24], the minimum cost perfect matching with conflict pair constraints [19] and the maximum flow problems with disjunctive constraints [21].

The MSTC can be classified as a special case of the Quadratic Minimum spanning Tree Problem (QMST) [7, 23]. Since in the QMST there is an additional cost into the objective function associated to the couple of edges

selected, to obtain a solution for the MSTC the idea is to associate a very high cost to all the edge pairs in conflict. In this way, by selecting edge pairs in conflict the objective function is penalized and then the search of a minimum spanning tree under these conditions should avoid the selection of edge pairs in conflict. When the solution of QMST does not contains edge pairs in conflict, a feasible solution for the MSTC is obtained. In [7], it has been proved that MSTC is NP-hard on fan-stars, fans, wheels and (k, n)-accordions graphs.

In this paper we propose a dual ascent and a subgradient procedure to compute lower bounds on the optimal solution of the MSTC problem. Moreover, we use a local search procedure to obtain upper bounds by removing the conflicts from the infeasible trees produced during the computation of the lower bounds. We devise an algorithm, based on these procedures, that is able to relatively quickly compute tight upper and lower bounds. The computational results, carried out on benchmark instances, demonstrate that our algorithm finds rather often the optimal solution. Moreover, the lower bounds it provides are better both in terms of quality and computation time than those obtained by similar Lagrangian approaches proposed in the literature.

The remainder of this paper is organized as follows. In Section 2 we introduce terminology and notation used throughout the paper, while in Section 3 a mathematical formulation for the MSTC is provided. In Section 4 we describe the Lagrangian relaxation and the dual ascent and subgradient procedures to compute the lower bounds and the local search procedure used to compute the upper bounds. In Section 5 we present our algorithm for the MSTC. Computational results as reported in Section 6. Finally, conclusions are provided in Section 7.

2. Notations and problem definition

Let $G = (V, E, P)$ be an undirected, edge weighted graph, where V is the set of n vertices, E is the set of m edges and $P \subseteq E \times E$ is the set of *conflict edge pairs*. Formally,

$$P = \{\{e_i, e_j\} : e_i, e_j \in E,\ e_i \textit{ and } e_j \textit{ are in conflict}\}.$$

For each $e_i \in E$, we denote by w_{e_i} its weight and by $\chi(e_i)$ the set of edges that are in conflict with it. Moreover, let $\mathbb{P}(e_i, E) = \{\{e_i, e_j\} \in P : e_j \in E\}$ be the set of conflict edge pairs containing the edge e_i and let $\zeta(E') = \cup_{e_i \in E'} \mathbb{P}(e_i, E')$ be the set of conflict edge pairs induced by $E' \subseteq E$. In Figure 1(a) is depicted a weighted graph $G(V, E, P)$ where $P = \{\{(b,c),(d,f)\}, \{(b,c),(c,f)\}, \{(b,e),(d,g)\}\}$. In this graph, we have that $\chi((b,c)) = \{(d,f),(c,f)\}$ and $\mathbb{P}((b,c), E) = \{\{(b,c),(d,f)\},\{(b,c),(c,f)\}\}$. Moreover, $\zeta((b,c),(b,e)) = P$.

A *spanning tree* $T(V_T, E_T)$ of G is a connected subgraph of G such that $V_T = V$, $E_T \subseteq E$ and $|E_T| = n-1$. The weight of T is denoted by $W(T)$ and it is given by the sum of edges weights in E_T. A tree T is *conflict–free* if and only if $|\zeta(E_T)| = 0$. The MSTC problem consists in finding the minimum weight conflict–free spanning tree of G .

In Figure 1(b) is shown the minimum spanning tree T of G. However, this tree is not a feasible solution for MSTC because the edges (b,c) and (d,f) (in red) and the edges (b,e) and (d,g) (in blue) are in conflict. Indeed $\zeta(E_T) = \{\{(b,c),(d,f)\},\{(b,e),(d,g)\}\}$. On the contrary, the tree depicted in Figure 1(c) is feasible for the MSTC because it is conflict–free.

3. The MSTC formulation

We introduce now a formulation for MSTC derived from the traditional Subtour Elimination formulation of the Minimum Spanning Tree problem. The

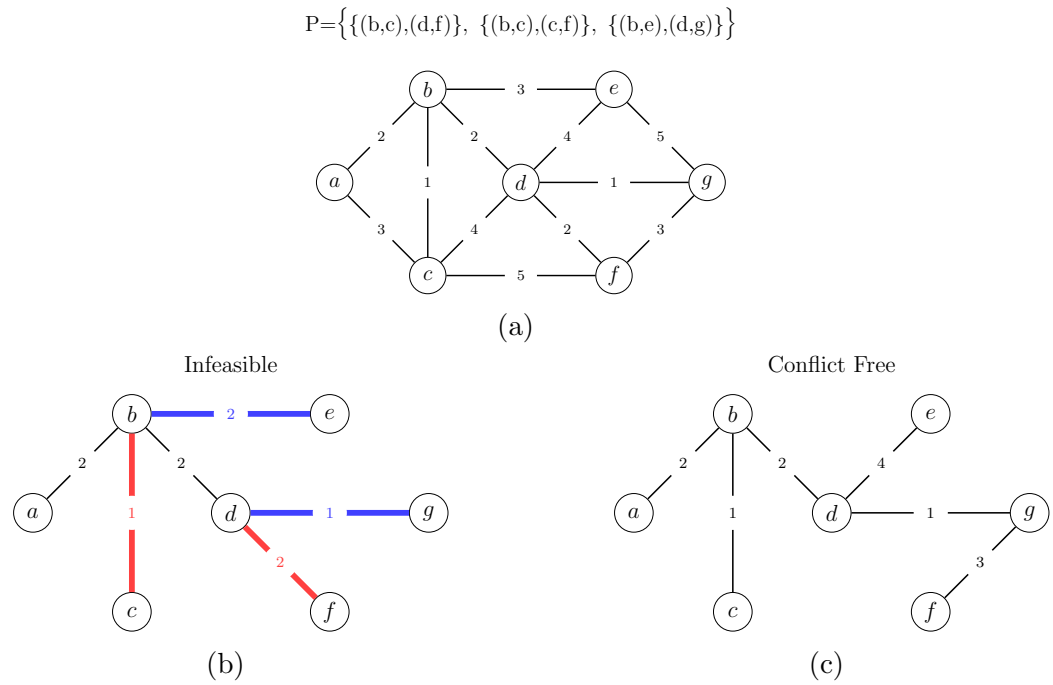

Figure 1: (a) A graph G having three conflict edge pairs. (b) The minimum spanning tree of G. (c) A feasible solution for the MSTC.

formulation of MSTC only uses the binary variables x_e, associated with the edges of G, with the following meaning:

$$x_e = \begin{cases} 1 & \text{if edge } e \text{ is selected,} \\ 0 & \text{otherwise.} \end{cases}$$

The mathematical programming formulation of the MSTC is the following one.

$$\textbf{IP} \quad \min \sum_{e \in E} w_e x_e \tag{1}$$

subject to:

$$\sum_{e \in E} x_e = |V| - 1, \tag{2}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1, \qquad \forall S \subseteq V, |S| \geq 3, \tag{3}$$

$$x_{e_i} + x_{e_j} \leq 1, \qquad \forall \{e_i, e_j\} \in P, \tag{4}$$

$$x_e \in \{0, 1\}, \qquad \forall e \in E. \tag{5}$$

The objective function (1) minimizes the weight of the spanning tree. Constraint (2) imposes the selection of $n-1$ edges (recall that $|V| = n$) while Constraints (3) are the classical subtour elimination constraints. Finally, Constraints (4) are the *conflict constraints*; these constraints ensure that two edges in conflict cannot be simultaneously selected in the solution. Finally, Constraints (5) are variable restrictions.

4. A Lagrangian approach dual

In tackling MSTC we adopt a Lagrangian-inspired approach which consists in defining a Lagrangian relaxation of the problem and in solving the related Lagrangian dual. In addition, we equip our method with a heuristic to be run at each iteration of the algorithm for Lagrangian dual, with the aim of calculating possibly improving upper bounds for the problem at hand.

4.1 The Lagrangian relaxation of MSTC

We focus on the (IP) formulation of MSTC presented in previous section and address its Lagrangian relaxation (see the seminal paper [15] and the survey [13]).

In particular, by relaxing the conflict constraints (4) by means of the Lagrangian multipliers $\lambda_{ij} \geq 0$, $\{e_i, e_j\} \in P$ (grouped into the vector λ of

appropriate dimension), we come out with the relaxed problem $LR(\lambda)$:

$$LR(\lambda) \qquad z(\lambda) = \min \sum_{e \in E} w_e x_e + \sum_{\{e_i, e_j\} \in P} \lambda_{ij}(x_{e_i} + x_{e_j} - 1) \tag{6}$$

subject to:

$$\sum_{e \in E} x_e = |V| - 1, \tag{7}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1, \qquad \forall S \subseteq V, |S| \geq 3, \tag{8}$$

$$x_e \in \{0, 1\}, \qquad \forall e \in E. \tag{9}$$

It can be easily recognized that, apart the constant term $(-\sum_{\{e_i, e_j\} \in P} \lambda_{ij})$, problem (6)–(9) is a standard minimum spanning tree one where the weights of the edges are assigned as follows:

$$\tilde{w}_{e_i}(\lambda) = \begin{cases} w_{e_i} & \text{if } \chi(e_i) = \emptyset, \\ w_{e_i} + \sum_{e_j \in \chi(e_i)} \lambda_{ij} & \text{otherwise.} \end{cases}$$

Function $z(\lambda)$ provides a lower bound on the optimal value of problem MSTC and in the following it will be referred to as the Lagrangian function. Note that, in case the minimum spanning tree $T(\lambda)$ computed for any choice of the multiplier vector λ is conflict–free, the solution of the relaxed problem is feasible for the original **IP** and it is ϵ-optimal, for ϵ defined as follows:

$$\epsilon = \sum_{\{e_i, e_j\} \in P} \lambda_{ij}\big(1 - (x_{e_i}(\lambda) + x_{e_j}(\lambda))\big),$$

where we have grouped into vector $x(\lambda)$ the optimal solution to $LR(\lambda)$. We remind that a solution is ϵ-optimal when the gap between its value and the optimal value is lower than or equal to ϵ.

In order to get the *best* lower bound, the Lagrangian dual (LD) problem is defined:

$$z_{LD} = \max_{\lambda \geq 0} z(\lambda). \tag{10}$$

Tackling the Lagrangian dual amounts to solve a maximization problem where the objective function is concave and piecewise affine, thus nonsmooth (see [16] for both the theoretical and computational aspects of nonsmooth optimization). Our approach to deal with problem (10) proceeds along with the following two guidelines:

- Construct an ad hoc dual ascent procedure, that is generate a sequence of multipliers vectors $\{\lambda^{(k)}\}$, $k = 1, 2, ...$, such that the corresponding values $z(\lambda^{(k)})$ are monotonically increasing;

- Whenever necessary, switch to a classic subgradient algorithm, where the multiplier setting $\{\lambda^{(k+1)}\}$ is obtained from $\{\lambda^{(k)}\}$ on the basis of any subgradient $g^{(k)} \in \partial z(\lambda^{(k)})$

The effectiveness of Lagrangian relaxation method to get a tight lower bound lies on possible easy calculation of the Lagrangian function $z(\lambda^{(k)})$, which amounts to have at disposal a fast method to solve $LR(\lambda^{(k)})$. This is, in fact, our case since it can be solved in polynomial time by, e.g., Kruskal algorithm.

We describe now our approach to multiplier updating in view of tackling the Lagrangian dual.

4.2 Solving the Lagrangian dual

Infeasibility of $T(\lambda^{(k)})$ is the key point to design a strategy for updating the multiplier vector in view of possible increase of the bound $z(\lambda^{(k)})$. It is based on the idea of updating just *one component* of the multiplier vector at a time (see the seminal paper [12] and [14], [4], [2], [6], [11] for applications in diverse areas).

Assuming, in fact, $T(\lambda^{(k)})$ infeasible, then there exists at least a pair of edges $\{e_i, e_j\} \in P$, such that

$$x_{e_i}(\lambda^{(k)}) + x_{e_j}(\lambda^{(k)}) - 1 > 0,$$

thus with $e_i \in E_{T(\lambda^{(k)})}$ and $e_j \in E_{T(\lambda^{(k)})}$. The idea at basis of the dual ascent procedure is that an increase of the corresponding multiplier leads to increase the dual function. In fact, once one of such pair of edges is selected, the corresponding multiplier is updated as follows

$$\lambda_{ij}^{(k+1)} = \lambda_{ij}^{(k)} + \Delta_{ij}, \tag{11}$$

for some $\Delta_{ij} > 0$. To calculate Δ_{ij} we consider the list $L^{(k)}$ of edges, sorted in nondecreasing order in Kruskal's algorithm. By defining $s(e_i)$ and $s(e_j)$, the successors of the edges e_i and e_j in $L^{(k)}$, respectively, we set

$$\Delta_{ij} = \min\{\tilde{w}_{s(e_i)} - \tilde{w}_{e_i},\ \tilde{w}_{s(e_j)} - \tilde{w}_{e_j}\} \geq 0 \tag{12}$$

Assuming $\Delta_{ij} > 0$ and implementing the multiplier update (11), it is easy to verify that

$$z(\lambda^{(k+1)}) = z(\lambda^{(k)}) + \Delta_{ij}.$$

We note in passing that $\Delta_{ij} = 0$ corresponds, typically, to nonsmoothness of function z at point $\lambda^{(k)}$. Note, in fact, that the Lagrangian dual is a max min problem, being z a min-type function. Assuming, w.l.o.g., $\tilde{w}_{s(e_i)} = \tilde{w}_{e_i}$, we have that the current MST problem, which provides the value of z, has possibly more than one optimal solution and this is a typical case when nonsmoothness occurs in a max min framework.

In case more than just one pair of conflicting edges are in $E_{T(\lambda^{(k)})}$ (that is $|\zeta(E_{T(\lambda^{(k)})})| > 1$), we update the multiplier of the pair characterized by the maximum Δ_{ij}.

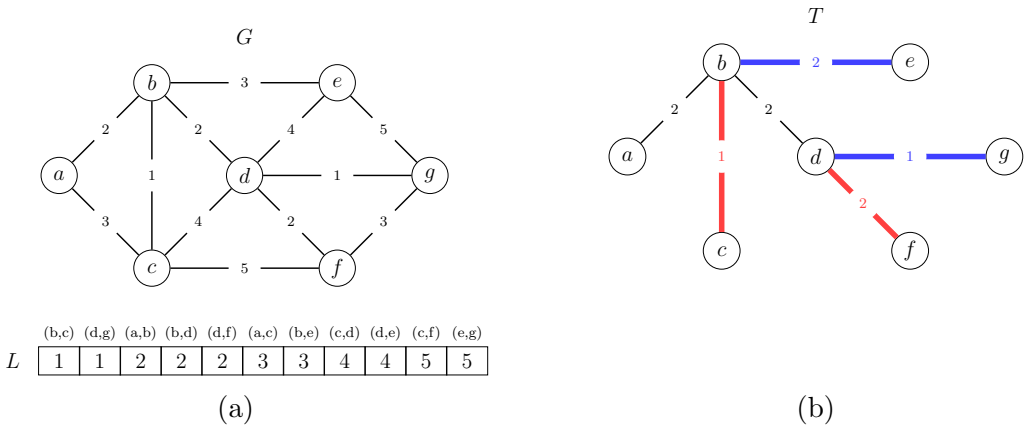

Figure 2: (a) The sorted edge list L generated on G by Kruskal's algorithm. (b) The minimum spanning tree of G found by Kruskal's algorithm.

Let us consider the graph G depicted in Figure 2(a) (see also Figure 1) and assume $k = 0$, $\lambda^{(0)} = 0$. Then application of Kruskal algorithm to the relaxed problem (see the sorted edge list $L^{(0)}$) provides the minimum spanning tree $T(0)$ shown in Figure 2(b) (note that, since current multipliers are zero, it coincides with the minimum spanning tree in Figure 1). Note that out of the three conflicting pair of edges

$$\{(b,c),(d,f)\},\ \{(b,c),(c,f)\},\ \{(b,e),(d,g)\},$$

the two pairs $\{(b,c),(d,f)\}$ and $\{(b,e),(d,g)\}$ are in $T(0)$. Applying (12) we obtain that the corresponding multipliers should increase by 0 and 1, respectively. As consequence, we select the conflicting pair $\{(b,e),(d,g)\}$ and increase by 1 the correspondent multiplier. It is easy to verify that, in the new multiplier setting $\lambda^{(1)}$, it is $z(\lambda^{(1)}) = z(\lambda^{(0)}) + 1$ and dual ascent has been achieved.

In practical application of our multiplier update rule, formula (11) is replaced by

$$\lambda_{ij}^{(k+1)} = \lambda_{ij}^{(k)} + \Delta_{ij} + \delta, \tag{13}$$

for some typical small $\delta > 0$. Motivation is to tie break, thus skipping the kinks of the Lagrangian function, while trying to expel a conflicting arc from the tree.

It is worth noting that, apart from the edge pair selection rule previously described, different heuristics can be devised. In particular, taking into account that a small value of a multiplier results in weak penalization of the violation of the corresponding constraint, we have adopted a rule which is based on a combination of the values of Δ_{ij} and of the multiplier, trying to give precedence to the edge pairs characterized by *large* Δ_{ij} and *small* multiplier. In particular, we select the edge pair where $max(\Delta_{ij} - \gamma\lambda_{ij}^{(k)})$ is achieved, where $\gamma > 0$ is a tradeoff parameter.

A major drawback of the dual ascent strategy based on updating one variable at a time is possible premature stop at a point which is far from being a maximizer of the Lagrangian function, while no coordinate axis is an ascent direction. Such a phenomenon is a consequence of nonsmoothness of function z (see [22], pages 138–139 for an illustrative example).

To overcome this difficulty, we have embedded into our algorithm a switching step to classic subgradient algorithm whenever the dual ascent procedure is unable to guarantee an increase in the objective function. We describe next the subgradient procedure we have embedded into our algorithm.

4.3 Subgradient procedure

A subgradient $g(\lambda)$ of $z(\lambda)$ is an element of the subdifferential $\partial z(\lambda) \subset R^{|P|}$ and can be easily calculated once an optimal solution $x(\lambda)$ to the relaxed problem is available. The generic component of $g(\lambda)$ is:

$$g_{ij}(\lambda) = x_{e_i}(\lambda) + x_{e_j}(\lambda) - 1, \qquad (e_i, e_j) \in P, \tag{14}$$

and iteration k of the subgradient method consists in updating the Lagrangian multipliers as follows:

$$\lambda_{ij}^{(k+1)} = \max\left(0, \lambda_{ij}^{(k)} + \alpha_k g_{ij}(\lambda^{(k)})\right), \qquad (e_i, e_j) \in P \tag{15}$$

where the stepsize α_k is chosen according to the Polyak's rule [22]:

$$\alpha_k = 2\frac{z_{UB} - z(\lambda^k)}{||g(\lambda^k)||^2}, \tag{16}$$

where z_{UB} is an upper bound on the optimal solution value. A trivial way to obtain a valid value for z_{UB} is to sum the weight of the n-1 heaviest edges of G. A more accurate bound is the maximum spanning tree of G, which can be calculated by multiplying the weights of all edges by -1 and then applying any minimum spanning tree algorithm.

5. The algorithm

In this section, we describe our algorithm, named *HDA* (Hybrid Dual Ascent), based on the dual ascent procedure previously described, with possible switching to the subgradient method. The method is primarily devoted to solve the Lagrangian dual, thus to obtain an optimal lower bound. In addition, we introduce a variant of *HDA* equipped with a local search procedure, which is called each time the relaxed problem is solved and it is aimed at finding a feasible solution to **IP** by removing the conflicts. In other words, we propose also a Lagrangian heuristic which is able to provide an upper bound too for MSTC. Figure 3 displays the flowchart of *HDA*.

The algorithm takes as input the graph G and applies to it the preprocessing phase described in [25]. This is a three-step iterative process, where each step is executed as long as the problem instance is updated. The aim

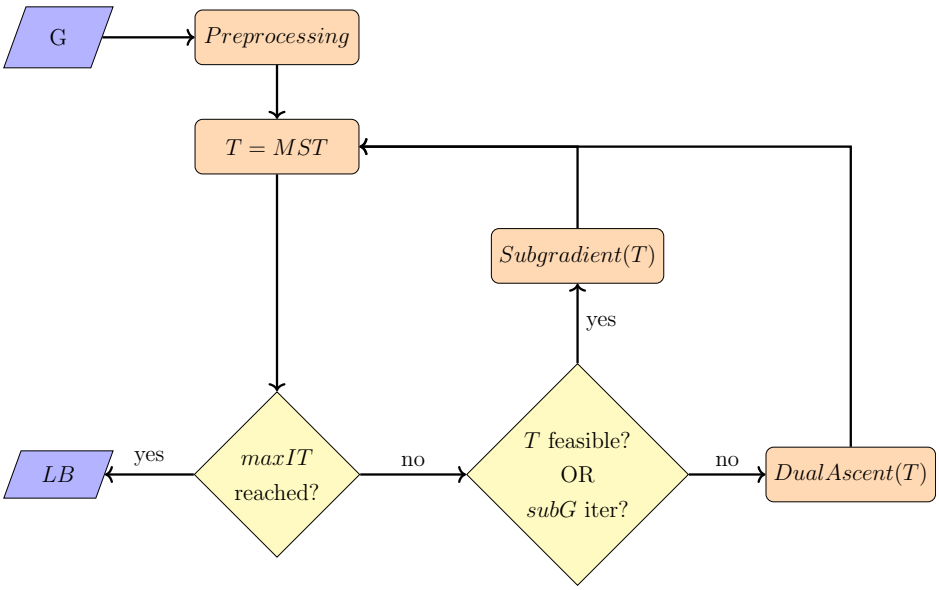

Figure 3: *HDA* flowchart

is to reduce, whenever possible, the size of the instance by removing edges that cannot belong to any feasible solution of MSTC. The first step looks for cut-edges (bridges) in G. Since a cut-edge e_i is in any spanning tree of G, any feasible solution of MSTC cannot contain edges in $\chi(e_i)$ and then these edges are removed from G. In the second step, it is verified the connectivity of the graph when an edge e_i is forced to be selected. If, due to the edges in $\chi(e_i)$, the resulting graph is disconnected, then e_i is removed from G. Finally, in the third step pairs of edges $\{e_i, e_j\}$ are selected and, after the removal of the edges in $\chi(e_i) \cup \chi(e_j)$, the connectivity of G is verified. If G is disconnected then the pair $\{e_i, e_j\}$ is added to P. In the following, we assume that G is the input graph on which the preprocessing phase is already applied.

At each iteration, a new minimum spanning tree T of G is computed by Kruskal's algorithm according to the weight $\tilde{w}_{e_i}$. The algorithm stops when a fixed number of iterations $maxIT$ is reached and the best lower bound

LB is returned. In our implementation, $maxIT$ is set to 500. The dual ascent procedure is invoked on T until one of the following conditions is satisfied. The former condition is that T is a feasible solution, that is, T is conflict–free. In this case, the dual ascent cannot work because there are not edge pairs in conflict to select. The second condition is the achievement of a fixed number of iterations. In our implementation, such a number is equal to 10% of $maxIT$. When one of the previous two conditions holds, *HDA* invokes ten times the *Subgradient* procedure. The flag *subG* states which one between the *Subgradient* and the *DualAscent* procedure is to be invoked at the current iteration. Once the ten iterations of the *Subgradient* procedure have been completed, the multipliers of the best lower bound are saved and used to update the weights $\tilde{w}_{e_i}$. We have introduced the second condition because the ascent of the *DualAscent* procedure can be slow due to the low value of Δ. By invoking the *Subgradient* procedure, after a fixed number of iterations, we obtain "jumps" on the lower bound values that often speed up the ascent of *DualAscent* procedure.

5.1 Upper Bound Computation

During the computation of *HDA* several spanning trees are generated and the conflict–free ones are feasible solutions for **IP**. Therefore, *HDA* could also provide upper bounds of **IP** optimal solution without additional costs. However, in order to increase the number of upper bounds found and their quality, we introduce a variant of *HDA* named *HDA*$^+$, that invokes a local search procedure on each spanning tree generated by Kruskal's algorithm. The aim of this local search procedure, named *ConflictRemoval*, is to make conflict–free the spanning tree given in input.

ConflictRemoval was introduced in [3] and used without modification in *HDA*$^+$. The procedure works as follow. Given an infeasible spanning tree

T, *ConflictRemoval* finds the edge $e_k \in E_T$ having the maximum number of conflicts with the other edges of E_T, i.e. $e_k = \underset{e_i \in E_T}{\operatorname{argmax}} |\mathbb{P}(e_i, E_T)|$. The procedure removes e_k from E_T generating a forest composed of two subtrees T_1 and T_2. To obtain a new spanning tree T', *ConflictRemoval* connects T_1 and T_2 by using the edge $e_r \in E \setminus E_T$ where $e_r = \underset{e_i \in E \setminus E_T}{\operatorname{argmin}} |\mathbb{P}(e_i, E_T \setminus \{e_k\})|$. If $|\zeta(E_{T'})| < |\zeta(E_T)|$ the procedure restart from T' otherwise it stops.

We will see in the next section that *HDA*$^+$ is slower than *HDA*, due to the use of *ConflictRemoval*, but it can find upper bounds whose quality is comparable with the ones obtained by the best metaheuristics for MSTC, proposed in the literature.

6. Computational Tests

In this section we describe the results of algorithms *HDA* and *HDA*$^+$. They have been coded in C++ using the LEMON graph library [10]. All tests have been performed on an OSX platform (iMac mid 2011), running on an Intel Core i7-2600 3.4 GHz processor (family 6, model 42, stepping 7) with 8 GB of RAM.

The computational tests are carried out on the instances proposed in [26] and in [5]. The dataset is available here: `http://www.dipmat2.unisa.it/people/carrabs/www/` or, alternatively, upon request to the authors. The first set of benchmark instances has been generated by using a different value for nodes, edges and number of conflicts and they are classified into two types: *type 1* and *type 2*. By construction, there exists at least one conflict–free solution for all type 2 instances while type 1 instances may not have conflict–free solutions. In the computational tests, we will consider only the type 1 instances where the presence of a feasible solution is guaranteed. The instances 10 and 11 are classified in literature as type 1 (unknown feasibility), even though the presence of a feasible solution for these instances was proved

	Instance				Opt	LB-MST			LB-MI			*HDA*		
	ID	**n**	**m**	**p**		**LB**	**Time**	**Gap**	**LB**	**Time**	**Gap**	**LB**	**Time**	**Gap**
Type 1 Feasible	1	50	200	199	708	701.09	0.12	0.98%	702.79	8.04	0.74%	705.50	0.26	0.35%
	2	50	200	398	770	739.84	0.16	3.92%	757.82	8.33	1.58%	760.98	0.31	1.17%
	3	50	200	597	917	782.67	0.35	14.65%	807.75	39.39	11.91%	867.66	0.35	5.38%
	4	50	200	995	1324	835.68	0.38	36.88%	877.50	39.72	33.72%	961.89	0.44	27.35%
	5	100	300	448	4041	3893.48	1.42	3.65%	3991.18	232.00	1.23%	4036.59	0.89	0.11%
	6	100	300	897	5658	4508.16	1.38	20.32%	4624.24	322.01	18.27%	4982.01	1.37	11.95%
	7	100	500	1247	4275	4124.53	2.54	3.52%	4165.68	610.83	2.56%	4268.57	2.25	0.15%
	8	100	500	2495	5997	4701.87	2.40	21.60%	4805.40	579.58	19.87%	5238.15	2.51	12.65%
	9	100	500	3741	7665*	4743.83	2.87	38.11%	4871.27	600.95	36.45%	5418.78	2.63	29.30%
Type 1 F.Unknown	10	200	600	1797	15029*	1111.60	10.37	92.60%	11425.80	3422.03	23.97%	12451.60	4.86	17.15%
	11	200	800	3196	22110*	17428.80	13.32	21.17%	17922.60	5107.72	18.94%	19685.10	8.59	10.97%
Type 2	12	50	200	3903	1636	775.12	0.81	52.62%	877.47	62.52	46.37%	1042.77	1.21	36.26%
	13	50	200	4877	2043	698.68	1.32	65.80%	887.48	60.01	56.56%	1116.34	1.63	45.36%
	14	50	200	5864	2338	626.92	1.02	73.19%	1030.25	62.14	55.93%	2338.00	0.81	0.00%
	15	100	300	8609	7434	4043.38	3.39	45.61%	5754.85	489.34	22.59%	7434.00	0.59	0.00%
	16	100	300	10686	7968	3970.52	2.98	50.17%	6192.29	436.52	22.29%	7968.00	0.55	0.00%
	17	100	300	12761	8166	3936.27	3.59	51.80%	6758.57	450.65	17.24%	8166.00	0.47	0.00%
	18	100	500	24740	12652	4289.85	11.30	66.09%	5104.90	794.09	59.65%	5565.20	9.04	56.01%
	19	100	500	30886	11232	3972.68	19.18	64.63%	5078.82	732.86	54.78%	5672.69	15.52	49.50%
	20	100	500	36827	11481	3918.06	26.44	65.87%	5710.77	662.14	50.26%	11481.00	22.90	0.00%
	21	200	400	13660	17728	14085.80	3.23	20.54%	17245.90	328.60	2.72%	17728.00	1.34	0.00%
	22	200	400	17089	18617	14067.30	4.33	24.44%	18048.20	341.14	3.06%	18617.00	1.46	0.00%
	23	200	400	20470	19140	13998.70	7.54	26.86%	18646.20	423.51	2.58%	19140.00	1.41	0.00%
	24	200	600	34504	20716	9466.06	36.91	54.31%	15393.10	3754.52	25.69%	20716.00	2.98	0.00%
	25	200	600	42860	18025	9100.64	51.67	49.51%	13971.50	3952.81	22.49%	18025.00	2.12	0.00%
	26	200	600	50984	20864	8734.53	58.49	58.14%	16708.10	4087.03	19.92%	20864.00	1.88	0.00%
	27	200	800	62625	39895	16806.50	88.95	57.87%	23792.30	5927.01	40.36%	39895.00	43.31	0.00%
	28	200	800	78387	37671	15803.10	107.83	58.05%	22174.20	5742.03	41.14%	37671.00	7.69	0.00%
	29	200	800	93978	38798	15470.10	131.01	60.13%	24907.00	5708.92	35.80%	38798.00	4.51	0.00%
	30	300	600	31000	43721	34154.20	29.44	21.88%	42720.60	2020.12	2.29%	43721.00	3.04	0.00%
	31	300	600	38216	44267	33320.10	16.59	24.73%	43486.70	1315.62	1.76%	44267.00	3.13	0.00%
	32	300	600	45310	43071	32072.30	8.67	25.54%	42149.00	1073.15	2.14%	43071.00	2.98	0.00%
	33	300	800	59600	43125	24384.30	90.67	43.46%	36629.60	15342.54	15.06%	43125.00	3.27	0.00%
	34	300	800	74500	42292	22913.20	107.97	45.82%	38069.30	14037.46	9.98%	42292.00	3.18	0.00%
	35	300	800	89300	44114	21624.60	31.68	50.98%	38843.00	13822.46	11.95%	44114.00	3.43	0.00%
	36	300	1000	96590	71562	36544.70	167.21	48.93%	56048.30	23406.23	21.68%	71562.00	9.20	0.00%
	37	300	1000	120500	76345	34380.80	195.12	54.97%	58780.10	21368.69	23.01%	76345.00	6.61	0.00%
	38	300	1000	144090	78880	33481.20	244.48	57.55%	60810.80	21375.46	22.91%	78880.00	5.15	0.00%
AVG							**39.14**	**41.50%**		**4177.58**	**22.62%**		**4.84**	**7.99%**

Table 1: Comparison among the Lagrangian approaches for the MSTC.

in [3]. We verify the effectiveness and the performance of *HDA* by comparing it with the two Lagrangian approaches, *LB-MST* and *LB-MI*, proposed for the MSTC in [26]. The results of this comparison are reported in Table 1. In order to have a fair comparative study, the CPU time reported in [26] has been scaled according to the Whetstone benchmark [1].

The first five columns of this table show the characteristics of the instances while the column Opt reports the optimal solution value or the best known solution value when this value is reported with the "*" symbol. The next nine columns report the lower bound (LB), the computational time (Time), in seconds, and the percentage gap (Gap) from the best known value of LB-MST, LB-MI and *HDA* algorithms, respectively. Notice that, in reporting the computational time of our algorithm, we have included the time required

by the preprocessing phase described in Section 5.

At the bottom of the table, the line AVG shows the average computational time and Gap values for the three algorithms. The results reported on AVG line show that *HDA* is much more effective of the other two algorithms. Indeed, its gap from the Opt value is around 8% while this gap increases to 22.62% for LB-MI and 41.50% for LB-MST. In any case, the Gap value of *HDA* is always better than the Gap value of the other two algorithms. Moreover, *HDA* reaches to find the optimal solution on 23 out of 35 instances, where an optimal solution is known, while LB-MI and LB-MST never find an optimal solution.

In comparing the running times of *HDA*, LB-MI and LB-MST, we observe that *HDA* is always faster than the other two algorithms. On average, the computational time of *HDA* is lower than 5 seconds and only in three cases this time is greater than 10 seconds. On the contrary, the average computational time of LB-MST and LB-MI is equal to 39 and to 4177 seconds, respectively. Actually, there are cases in which LB-MI takes so much time (more than 21000 seconds) that its use in practice is not convenient. According to the results of Table 1, we observe that *HDA* is suitable for being embedded into a branch and bound approach because it can quickly find tight lower bounds.

In order to further investigate the effectiveness and performance of *HDA*, we run our algorithm even on the new instances proposed in [5]. These instances were generated with a number of nodes that ranges from 25 to 100 with a step of 25. The number of edges m is assigned according to a density equal to 0.2, 0.3, 0.4. A random integer weight chosen in the interval [10, 30] is assigned to each edge. This means that a graph with a density equal to d has $m = dn(n-1)/2$ edges. Finally, the number of conflicts pairs p associated with each instance is equal to 1%, 4%, 7% of $m(m-1)/2$. For each

Instance						*HDA*			Instance						*HDA*		
ID	**n**	**m**	**p**	**s**	**Opt**	**LB**	**Time**	**Gap**	**ID**	**n**	**m**	**p**	**s**	**Opt**	**LB**	**Time**	**Gap**
51	25	60	18	1	347	347.00	0.04	0.00%	96	50	245	299	271	619	619.00	0.34	0.00%
52	25	60	18	7	389	389.00	0.04	0.00%	97	50	245	299	277	604	604.00	0.36	0.00%
53	25	60	18	13	353	353.00	0.04	0.00%	98	50	245	299	283	634	634.00	0.34	0.00%
54	25	60	18	19	346	346.00	0.04	0.00%	99	50	245	299	289	616	615.50	0.34	0.08%
55	25	60	18	25	336	336.00	0.04	0.00%	100	50	245	299	295	595	595.00	0.34	0.00%
56	25	60	71	31	381	379.63	0.06	0.36%	101	50	245	1196	301	678	668.50	0.57	1.40%
57	25	60	71	37	390	381.50	0.05	2.18%	102	50	245	1196	307	681	655.49	0.56	3.75%
58	25	60	71	43	372	372.00	0.05	0.00%	103	50	245	1196	313	709	678.22	0.59	4.34%
59	25	60	71	49	357	357.00	0.05	0.00%	104	50	245	1196	319	639	634.04	0.56	0.78%
60	25	60	71	55	406	406.00	0.05	0.00%	105	50	245	1196	325	681	658.82	0.56	3.26%
61	25	60	124	61	385	385.00	0.06	0.00%	106	50	245	2093	331	833*	661.27	0.76	20.62%
62	25	60	124	67	432	432.00	0.06	0.00%	107	50	245	2093	337	835	705.98	0.75	15.45%
63	25	60	124	73	458	424.48	0.07	7.32%	108	50	245	2093	343	840*	666.45	0.74	20.66%
64	25	60	124	79	400	398.00	0.07	0.50%	109	50	245	2093	349	836*	680.81	0.76	18.56%
65	25	60	124	85	420	406.77	0.08	3.15%	110	50	245	2093	355	769	693.86	0.74	9.77%
66	25	90	41	91	311	310.10	0.05	0.29%	111	50	367	672	361	570	570.00	0.82	0.00%
67	25	90	41	97	306	306.00	0.05	0.00%	112	50	367	672	367	561	561.00	0.86	0.00%
68	25	90	41	103	299	299.00	0.05	0.00%	113	50	367	672	373	573	573.00	0.86	0.00%
69	25	90	41	109	297	297.00	0.05	0.00%	114	50	367	672	379	560	560.00	0.86	0.00%
70	25	90	41	115	318	318.00	0.05	0.00%	115	50	367	672	385	549	549.00	0.83	0.00%
71	25	90	161	121	305	305.00	0.08	0.00%	116	50	367	2687	391	612	596.44	1.26	2.54%
72	25	90	161	127	339	339.00	0.09	0.00%	117	50	367	2687	397	615	596.67	1.27	2.98%
73	25	90	161	133	344	344.00	0.08	0.00%	118	50	367	2687	403	587	577.09	1.26	1.69%
74	25	90	161	139	329	328.00	0.09	0.31%	119	50	367	2687	409	634	608.47	1.26	4.03%
75	25	90	161	145	326	325.00	0.08	0.31%	120	50	367	2687	415	643	636.06	1.26	1.08%
76	25	90	281	151	349	347.93	0.10	0.31%	121	50	367	4702	421	726*	624.43	1.66	13.99%
77	25	90	281	157	385	370.63	0.10	3.73%	122	50	367	4702	427	770*	635.39	1.65	17.48%
78	25	90	281	163	335	330.78	0.11	1.26%	123	50	367	4702	433	786*	646.89	1.64	17.70%
79	25	90	281	169	348	334.67	0.10	3.83%	124	50	367	4702	439	711*	597.47	1.72	15.97%
80	25	90	281	175	357	350.05	0.10	1.95%	125	50	367	4702	445	764*	665.14	1.71	12.94%
81	25	120	72	181	282	282.00	0.07	0.00%	126	50	490	1199	451	548	548.00	1.68	0.00%
82	25	120	72	187	294	294.00	0.06	0.00%	127	50	490	1199	457	530	530.00	1.66	0.00%
83	25	120	72	193	284	284.00	0.06	0.00%	128	50	490	1199	463	549	549.00	1.65	0.00%
84	25	120	72	199	281	281.00	0.06	0.00%	129	50	490	1199	469	540	540.00	1.67	0.00%
85	25	120	72	205	292	292.00	0.06	0.00%	130	50	490	1199	475	540	540.00	1.65	0.00%
86	25	120	286	211	321	320.25	0.12	0.24%	131	50	490	4793	481	594	584.26	2.38	1.64%
87	25	120	286	217	317	317.00	0.12	0.00%	132	50	490	4793	487	579	559.85	2.40	3.31%
88	25	120	286	223	284	284.00	0.12	0.00%	133	50	490	4793	493	589	578.34	2.43	1.81%
89	25	120	286	229	311	311.00	0.12	0.00%	134	50	490	4793	499	577	565.41	2.39	2.01%
90	25	120	286	235	290	290.00	0.13	0.00%	135	50	490	4793	505	592	577.44	2.40	2.46%
91	25	120	500	241	329	319.24	0.16	2.97%	136	50	490	8387	511	678*	574.99	3.08	15.19%
92	25	120	500	247	339	326.05	0.16	3.82%	137	50	490	8387	517	651*	569.00	3.07	12.60%
93	25	120	500	253	368	354.49	0.16	3.67%	138	50	490	8387	523	689*	592.20	3.08	14.05%
94	25	120	500	259	311	306.54	0.16	1.44%	139	50	490	8387	529	682*	592.50	3.07	13.12%
95	25	120	500	265	321	318.01	0.16	0.93%	140	50	490	8387	535	674*	587.45	3.09	12.84%
AVG							**0.08**	**0.86%**	**AVG**							**1.40**	**5.96%**

Table 2: Comparison between the optimal solutions and the lower bounds found by *HDA* on the small instances proposed in [5].

combination of parameters n, m and p, 5 different instances were generated for a total of 180 instances. The parameter s is the seed used to initialize the random number generator. The results of *HDA* on these instances are shown in Table 2 and Table 3.

The first five columns of these tables show the characteristics of the in-

stances. The column Opt reports the optimal solution value computed by the branch-and-cut (BC) proposed in [5]. When a certified optimal solution is not found within the time limit of 5000 seconds, the best solution found is shown, if available, and a "*" symbol is added near the related value. The next three columns show the lower bound (LB), the CPU time (Time), the gap percentage from the Opt value (Gap), respectively. Finally, the last line of the table reports the average values of Time and Gap columns.

In Table 2 the results on the small instances with 25 and 50 nodes are reported. For the instances with 25 nodes, *HDA* finds the optimal solution on 23 out of 38 instances and the gap is greater than 3% only six times. The average gap value of *HDA* is equal to 0.86%. The CPU time requires to obtain these results is around 0.1 seconds. By increasing the size of the instances to 50 nodes, we observe that the average gap grows to 6% and *HDA* finds the optimal solution only 13 times. However, we have to consider here that BC provides the optimal solution for 32 out of 45 instances while on the remaining instances only an upper bound is provided. Of course, comparison of the lower bound with an upper bound, rather than with the optimal solution, provides an overestimate of the gap. Indeed, the average gap value computed only on the 32 instances, where the optimal solution is known, is equal to 2%. The computational time of *HDA* remains negligible with an average of 1.4 seconds.

In Table 3 the results of *HDA* on the large instances are reported. Unfortunately, due to the size of these instances and, in particular, to the number of conflicts present in them, the BC algorithm finds the optimal solution only for the instances with the minimum number of conflicts. Instead, on the instances with the greatest number of conflicts, BC fails to provide even an upper bound. As a consequence, the number of comparisons performed

Instance						*HDA*		
ID	n	m	p	s	Opt	LB	Time	Gap
141	75	555	1538	541	868	868.00	2.51	0.00%
142	75	555	1538	547	871	870.93	2.49	0.01%
143	75	555	1538	553	838	838.00	2.45	0.00%
144	75	555	1538	559	855	855.00	2.47	0.00%
145	75	555	1538	565	857	857.00	2.47	0.00%
146	75	555	6150	571	1047*	942.34	3.49	10.00%
147	75	555	6150	577	1069*	937.61	3.54	12.29%
148	75	555	6150	583	1040*	910.90	3.48	12.41%
149	75	555	6150	589	998*	913.03	3.53	8.51%
150	75	555	6150	595	994*	897.88	3.54	9.67%
151	75	555	10762	601		922.49	4.44	
152	75	555	10762	607		947.99	4.40	
153	75	555	10762	613		901.94	4.44	
154	75	555	10762	619		888.35	4.44	
155	75	555	10762	625		914.72	4.42	
156	75	832	3457	631	798	797.77	6.86	0.03%
157	75	832	3457	637	821	820.00	6.83	0.12%
158	75	832	3457	643	816	815.32	6.87	0.08%
159	75	832	3457	649	820	819.97	7.16	0.00%
160	75	832	3457	655	815	814.99	7.23	0.00%
161	75	832	13828	661	903*	830.85	9.08	7.99%
162	75	832	13828	667	953*	859.49	9.05	9.81%
163	75	832	13828	673	892*	829.00	9.04	7.06%
164	75	832	13828	679	915*	842.24	8.98	7.95%
165	75	832	13828	685	896*	846.70	9.05	5.50%
166	75	832	24199	691		869.99	12.24	
167	75	832	24199	697		837.71	12.33	
168	75	832	24199	703		842.00	12.01	
169	75	832	24199	709		862.99	11.98	
170	75	832	24199	715		878.66	12.14	
171	75	1110	6155	721	787	787.00	14.88	0.00%
172	75	1110	6155	727	785	785.00	14.82	0.00%
173	75	1110	6155	733	783	783.00	14.78	0.00%
174	75	1110	6155	739	784	783.88	14.94	0.02%
175	75	1110	6155	745	797	796.50	14.86	0.06%
176	75	1110	24620	751	867*	816.88	19.16	5.78%
177	75	1110	24620	757	851*	797.98	19.28	6.23%
178	75	1110	24620	763	892*	808.43	19.44	9.37%
179	75	1110	24620	769	864*	807.97	19.31	6.48%
180	75	1110	24620	775	882*	803.99	19.32	8.84%
181	75	1110	43085	781		816.97	27.33	
182	75	1110	43085	787		804.00	26.74	
183	75	1110	43085	793	1194*	827.29	26.90	30.71%
184	75	1110	43085	799		799.00	27.17	
185	75	1110	43085	805		806.00	27.14	
AVG							**11.09**	**5.13%**

Instance						*HDA*		
ID	n	m	p	s	Opt	LB	Time	Gap
186	100	990	4896	811	1119	1118.48	11.59	0.05%
187	100	990	4896	817	1137	1134.74	11.43	0.20%
188	100	990	4896	823	1113	1112.45	11.71	0.05%
189	100	990	4896	829	1110	1109.63	11.48	0.03%
190	100	990	4896	835	1090	1088.67	11.99	0.12%
191	100	990	19583	841		1175.62	15.37	
192	100	990	19583	847	1491*	1141.98	14.98	23.41%
193	100	990	19583	853	1510*	1140.74	14.80	24.45%
194	100	990	19583	859	1441*	1178.06	14.82	18.25%
195	100	990	19583	865	1560*	1176.36	15.36	24.59%
196	100	990	34269	871		1126.42	22.20	
197	100	990	34269	877		1148.91	24.59	
198	100	990	34269	883		1179.91	21.17	
199	100	990	34269	889		1146.74	20.30	
200	100	990	34269	895		1154.99	20.42	
201	100	1485	11019	901	1079	1077.94	34.81	0.10%
202	100	1485	11019	907	1056	1054.88	34.46	0.11%
203	100	1485	11019	913	1059	1059.00	34.29	0.00%
204	100	1485	11019	919	1046	1045.99	34.09	0.00%
205	100	1485	11019	925	1072	1071.22	34.17	0.07%
206	100	1485	44075	931	1374*	1096.86	46.07	20.17%
207	100	1485	44075	937	1291*	1090.97	45.60	15.49%
208	100	1485	44075	943	1344*	1084.44	46.54	19.31%
209	100	1485	44075	949	1286*	1089.38	46.01	15.29%
210	100	1485	44075	955	1370*	1088.39	45.84	20.56%
211	100	1485	77131	961		1083.49	60.68	
212	100	1485	77131	967		1085.99	60.29	
213	100	1485	77131	973		1093.98	60.69	
214	100	1485	77131	979		1105.91	60.83	
215	100	1485	77131	985		1090.90	60.63	
216	100	1980	19593	991	1031	1030.91	76.15	0.01%
217	100	1980	19593	997	1036	1034.88	76.02	0.11%
218	100	1980	19593	1003	1024	1024.00	76.43	0.00%
219	100	1980	19593	1009	1025	1024.98	76.67	0.00%
220	100	1980	19593	1015	1028	1027.50	77.49	0.05%
221	100	1980	78369	1021	1234*	1060.50	106.68	14.06%
222	100	1980	78369	1027	1187*	1033.99	106.82	12.89%
223	100	1980	78369	1033	1213*	1050.54	104.96	13.39%
224	100	1980	78369	1039	1221*	1048.10	103.83	14.16%
225	100	1980	78369	1045	1245*	1048.88	103.99	15.75%
226	100	1980	137145	1051		1041.79	129.47	
227	100	1980	137145	1057		1066.49	129.26	
228	100	1980	137145	1063		1052.36	130.38	
229	100	1980	137145	1069		1062.85	130.01	
230	100	1980	137145	1075		1055.48	132.96	
AVG							**55.74**	**8.71%**

Table 3: Comparison between the optimal solutions and the lower bounds found by *HDA* on the large instances proposed in [5].

on these instances is limited. The AVG values show that on the instances with 75 nodes, the Gap value is around 5% even if this value is computed by comparing our lower bound with an upper bound, instead of with the optimal solution. Indeed, the worst Gap values are often observed on the instances where BC provides an upper bound. On average, the CPU time is around 11 seconds and it never exceeds 30 seconds. It is interesting to observe that as the number of conflicts increases as the computational time

of *HDA* increases. This trend will be confirmed even on the largest instances with 100 nodes and it proves that the number of conflicts is the main parameter that affects the performance of *HDA* much more than the number of nodes or edges on the instance.

Finally, on the instances with 100 nodes, the average gap value is equal to 8.71%. As for the instances with 75 nodes, the highest Gap occurs on those ones where BC provides an upper bound. By computing the average gap only on the instance with an optimal solution, this value is equal to 0.06%. On average, the computational time is around 55 seconds and it never exceeds 133 seconds. These results show that *HDA* remains very fast. However, also for these instances, the trend observed for the 75 nodes ones is confirmed, that is the computational time increases with the number of conflicts. To highlight the impact of the conflicts on the performance of *HDA* let us compare the time required to solve the instances number 216 and 230. Both these instances have the same number of nodes and edges but the instance 230 contains much more conflicts. In fact, the time required to solve the instance 230 is 75% greater than the time required to solve the instance 216.

6.1 Upper Bounds Results

In Section 5.1, we described the *ConflictRemoval* procedure that tries to remove the conflicts from a spanning tree to produce a feasible solution for the MSTC. We introduced *ConflictRemoval* in the computations carried out by *HDA* to obtain the new algorithm HDA^+ that provides a larger number of upper bounds. In order to evaluate the quality of these upper bounds, we compare them with those produced by metaheuristic *MEGA* proposed in [3]. Although we do not expect that our simple local search procedure overcomes the effectiveness of such ad hoc metaheuristic, nevertheless the

	Instance					*MEGA*			*HDA*$^+$		
	ID	**n**	**m**	**p**	**Opt**	**UB**	**Time**	**Gap**	**UB**	**Time**	**Gap**
Type 1 Feasible	1	50	200	199	708	708	0.71	0.00%	726	0.35	2.54%
	2	50	200	398	770	770	0.68	0.00%	780	0.45	1.30%
	3	50	200	597	917	917	0.63	0.00%	1176	0.56	28.24%
	4	50	200	995	1324	1336	0.66	0.91%	1587	0.79	19.86%
	5	100	300	448	4041	4088	2.39	1.16%	4099	1.19	1.44%
	6	100	300	897	5658	6095	1.81	7.72%		2.14	
	7	100	500	1247	4275	4275	5.18	0.00%	4301	2.96	0.61%
	8	100	500	2495	5997	6199	5.12	3.37%	8214	5.49	36.97%
	9	100	500	3741	7665*	7665	3.72	0.00%		6.81	
Type 1 F.Unknown	10	200	600	1797	15029*	15029	12.23	0.00%		8.57	
	11	200	800	3196	22110*	22110	23.42	0.00%		18.93	
Type 2	12	50	200	3903	1636	1636	0.46	0.00%	1653	2.17	1.04%
	13	50	200	4877	2043	2043	0.47	0.00%	2043	2.51	0.00%
	14	50	200	5864	2338	2338	0.51	0.00%	2338	0.82	0.00%
	15	100	300	8609	7434	7434	1.88	0.00%	7434	0.64	0.00%
	16	100	300	10686	7968	7968	1.68	0.00%	7968	0.54	0.00%
	17	100	300	12761	8166	8166	1.72	0.00%	8166	0.46	0.00%
	18	100	500	24740	12652	12652	3.30	0.00%	12681	20.82	0.23%
	19	100	500	30886	11232	11232	3.48	0.00%	11287	26.37	0.49%
	20	100	500	36827	11481	11481	3.51	0.00%	11481	24.36	0.00%
	21	200	400	13660	17728	17728	7.25	0.00%	17728	1.40	0.00%
	22	200	400	17089	18617	18617	7.49	0.00%	18617	1.44	0.00%
	23	200	400	20470	19140	19140	7.40	0.00%	19140	1.44	0.00%
	24	200	600	34504	20716	20716	11.17	0.00%	20716	3.01	0.00%
	25	200	600	42860	18025	18025	11.35	0.00%	18025	2.24	0.00%
	26	200	600	50984	20864	20864	12.38	0.00%	20864	1.99	0.00%
	27	200	800	62625	39895	39895	16.35	0.00%	39895	45.78	0.00%
	28	200	800	78387	37671	37671	15.70	0.00%	37671	8.19	0.00%
	29	200	800	93978	38798	38798	16.02	0.00%	38798	4.71	0.00%
	30	300	600	31000	43721	43721	18.61	0.00%	43721	3.44	0.00%
	31	300	600	38216	44267	44267	21.28	0.00%	44267	3.36	0.00%
	32	300	600	45310	43071	43071	24.32	0.00%	43071	3.06	0.00%
	33	300	800	59600	43125	43125	31.86	0.00%	43125	3.10	0.00%
	34	300	800	74500	42292	42292	34.31	0.00%	42292	3.21	0.00%
	35	300	800	89300	44114	44114	34.25	0.00%	44114	3.15	0.00%
	36	300	1000	96590	71562	71562	39.81	0.00%	71562	9.52	0.00%
	37	300	1000	120500	76345	76345	31.60	0.00%	76345	6.70	0.00%
	38	300	1000	144090	78880	78880	36.11	0.00%	78880	5.43	0.00%
AVG							**11.86**	**0.35%**		**6.27**	**2.73%**

Table 4: Comparison between the feasible solutions found by Mega and *HDA*$^+$ for the MSTC.

results reported in Table 4 show that the upper bounds of *HDA*$^+$ and *MEGA* are at least comparable with *HDA*$^+$ being significantly faster than *MEGA*. The headings of Table 4 are the same as in Table 1, except for the LB columns that are replaced by the upper bound (UB) columns.

It is apparent (see AVG line) that *MEGA* is more effective than *HDA*$^+$ because its average gap from the Opt value is equal to 0.35% while this gap is

equal to 2.73% for *HDA*$^+$. *MEGA* finds the optimal solution on 34 out of 38 instances and, in the remaining instances, its Gap value is smaller than 8%. Instead, *HDA*$^+$ finds the optimal solution for 24 out of 38 instances and, on 31 out of 38 instances, its Gap value is lower than 2.6%. In four cases *HDA*$^+$ did not even find a feasible solution and on the remaining three instances the Gap value is poor. The counterpart is *HDA*$^+$ ability to obtain rather tight upper bounds by just invoking a relatively cheap local search procedure. Regarding the performance, since *MEGA* and *HDA*$^+$ are executed on the same machine, their CPU times are directly comparable. From the average CPU time reported in the last row of Table 4, we can see that *HDA*$^+$ is around two times faster than *MEGA*. In particular, the CPU time of *HDA*$^+$ results 27 times lower than 5 seconds while for *MEGA* this occurs only 16 times. There are, however, some instances where *MEGA* is faster than *HDA*$^+$.

It is worth noting that the computational time of *HDA*$^+$ is not significantly greater than the computational time of *HDA*, at least on these instances because it requires two seconds more than *HDA*, on average.

To further investigate the effectiveness of *HDA*$^+$, we compare the upper bounds with the optimal solutions provided by B&C on the set of small instances proposed in [5]. The results of this comparison are reported in Table 5. Table headings have the same meaning they have for Table 3, the only difference being that column LB is replaced by column UB. On the instances, with 25 nodes the upper bounds provided by *HDA*$^+$ coincide with the optimal solution on 33 out of 45 instances. On the remaining 12 instances, the gap is greater than 3% in only three cases and, in the worst case, it is equal to 4.08%. The average gap in all these instances is 0.46%. These results show that effectiveness of *HDA*$^+$, on the small instances. The computational time is negligible because, on average, it is equal to 0.11 seconds and in the worst case it is equal to 0.3 seconds. On the instances with 50 nodes the quality of the upper bounds decreases. Indeed, these upper bounds coincide with

Instance						HDA^+			Instance						HDA^+		
ID	**n**	**m**	**p**	**s**	**Opt**	**UB**	**Time**	**Gap**	**ID**	**n**	**m**	**p**	**s**	**Opt**	**UB**	**Time**	**Gap**
51	25	60	18	1	347	347	0.05	0.00%	96	50	245	299	271	619	619	0.37	0.00%
52	25	60	18	7	389	389	0.04	0.00%	97	50	245	299	277	604	604	0.38	0.00%
53	25	60	18	13	353	353	0.04	0.00%	98	50	245	299	283	634	634	0.35	0.00%
54	25	60	18	19	346	346	0.05	0.00%	99	50	245	299	289	616	616	0.40	0.00%
55	25	60	18	25	336	336	0.04	0.00%	100	50	245	299	295	595	595	0.44	0.00%
56	25	60	71	31	381	381	0.07	0.00%	101	50	245	1196	301	678	698	0.96	2.95%
57	25	60	71	37	390	390	0.07	0.00%	102	50	245	1196	307	681	721	0.99	5.87%
58	25	60	71	43	372	372	0.06	0.00%	103	50	245	1196	313	709	725	1.00	2.26%
59	25	60	71	49	357	357	0.06	0.00%	104	50	245	1196	319	639	656	0.86	2.66%
60	25	60	71	55	406	406	0.06	0.00%	105	50	245	1196	325	681	748	1.01	9.84%
61	25	60	124	61	385	385	0.07	0.00%	106	50	245	2093	331	833*			
62	25	60	124	67	432	432	0.07	0.00%	107	50	245	2093	337	835			
63	25	60	124	73	458	474	0.10	3.49%	108	50	245	2093	343	840*			
64	25	60	124	79	400	400	0.08	0.00%	109	50	245	2093	349	836*			
65	25	60	124	85	420	421	0.12	0.24%	110	50	245	2093	355	769			
66	25	90	41	91	311	311	0.05	0.00%	111	50	367	672	361	570	570	0.88	0.00%
67	25	90	41	97	306	306	0.06	0.00%	112	50	367	672	367	561	561	0.98	0.00%
68	25	90	41	103	299	299	0.05	0.00%	113	50	367	672	373	573	573	1.06	0.00%
69	25	90	41	109	297	297	0.05	0.00%	114	50	367	672	379	560	560	0.87	0.00%
70	25	90	41	115	318	318	0.05	0.00%	115	50	367	672	385	549	551	0.99	0.36%
71	25	90	161	121	305	305	0.09	0.00%	116	50	367	2687	391	612	657	2.37	7.35%
72	25	90	161	127	339	339	0.11	0.00%	117	50	367	2687	397	615	663	2.24	7.80%
73	25	90	161	133	344	344	0.09	0.00%	118	50	367	2687	403	587	635	2.33	8.18%
74	25	90	161	139	329	331	0.11	0.61%	119	50	367	2687	409	634	721	2.40	13.72%
75	25	90	161	145	326	327	0.11	0.31%	120	50	367	2687	415	643	688	2.45	7.00%
76	25	90	281	151	349	349	0.13	0.00%	121	50	367	4702	421	726*			
77	25	90	281	157	385	385	0.16	0.00%	122	50	367	4702	427	770*			
78	25	90	281	163	335	335	0.14	0.00%	123	50	367	4702	433	786*			
79	25	90	281	169	348	358	0.18	2.87%	124	50	367	4702	439	711*			
80	25	90	281	175	357	359	0.17	0.56%	125	50	367	4702	445	764*	868	3.36	13.61%
81	25	120	72	181	282	282	0.06	0.00%	126	50	490	1199	451	548	552	2.03	0.73%
82	25	120	72	187	294	294	0.06	0.00%	127	50	490	1199	457	530	531	1.88	0.19%
83	25	120	72	193	284	284	0.08	0.00%	128	50	490	1199	463	549	549	1.76	0.00%
84	25	120	72	199	281	281	0.07	0.00%	129	50	490	1199	469	540	541	2.01	0.19%
85	25	120	72	205	292	292	0.06	0.00%	130	50	490	1199	475	540	540	1.71	0.00%
86	25	120	286	211	321	321	0.16	0.00%	131	50	490	4793	481	594	629	4.63	5.89%
87	25	120	286	217	317	317	0.17	0.00%	132	50	490	4793	487	579	650	4.90	12.26%
88	25	120	286	223	284	284	0.12	0.00%	133	50	490	4793	493	589	657	4.75	11.54%
89	25	120	286	229	311	312	0.17	0.32%	134	50	490	4793	499	577	643	4.74	11.44%
90	25	120	286	235	290	290	0.13	0.00%	135	50	490	4793	505	592	670	4.71	13.18%
91	25	120	500	241	329	341	0.26	3.65%	136	50	490	8387	511	678*	812	6.54	19.76%
92	25	120	500	247	339	347	0.25	2.36%	137	50	490	8387	517	651*			
93	25	120	500	253	368	383	0.30	4.08%	138	50	490	8387	523	689*			
94	25	120	500	259	311	314	0.22	0.96%	139	50	490	8387	529	682*			
95	25	120	500	265	321	325	0.25	1.25%	140	50	490	8387	535	674*	828	6.95	22.85%
AVG							**0.11**	**0.46%**								**2.56**	**5.44%**

Table 5: Comparison between the optimal solutions and the upper bounds found by HDA^+ on the small instances proposed in [5].

the optimal solution on 11 out of 45 instances while there are 12 instances where HDA^+ does not find an upper bound. Often these instances coincide with the cases where the B&C does not provide the optimal solution. The

average gap is equal to 5.44% but in seven cases it is greater than 10%. The computational time is always lower than 7 seconds and, on average, it is equal to 2.56 seconds.

7. Conclusion

In this paper, we developed a Lagrangian approach, *HDA*, to solve the Minimum Spanning Tree Problem with Conflicting Edge Pairs; the method computes both upper and lower bounds for the MSTC. The computation of the lower bound is obtained by alternating a Dual Ascent and a Subgradient procedure, while the upper bound is obtained by using a local search procedure acting on the infeasible solutions generated during the lower bound computation. We tested the performance of *HDA* in terms of accuracy and running time. The computational results on type 1 and type 2 instances show that the lower bounds of *HDA* coincide with the optimal solution in 60% of the cases. Moreover, *HDA* is faster and more effective than the other two Lagrangian approaches proposed in the literature. Regarding the upper bounds, HDA^+ does not reach the performance of *MEGA* but in 60% of instances the bounds provided by the two algorithms coincide with the optimal solution. Moreover, the upper bounds provided by HDA^+ often coincide or are very close to the optimal solution on the instances with 25 nodes proposed in [5] while, on the instances with 50, nodes we observed a quality reduction of these bounds. Finally, the results on the type 3 instances confirm the good solution quality of the lower bounds provided by our approach.

Acknowledgements. The authors have been partially supported by Ministero dell'Istruzione, dell'Universitá e della Ricerca (MIUR) with the program PRIN 2015, project title SPORT - Smart PORt Terminals, code 2015XAPRKF 005.

References

[1] Whetstone benchmark. `https://setiathome.berkeley.edu/cpu_list.php`.

[2] A. Astorino, M. Gaudioso, and G. Miglionico. Lagrangian relaxation for the directional sensor coverage problem with continuous orientation. *Omega*, 75:77–86, 2018.

[3] F. Carrabs, C. Cerrone, and R. Pentangelo. A multi-ethnic genetic approach for the minimum conflict weighted spanning tree problem. *Networks*, 74(2):134–147, 2019.

[4] F. Carrabs, R. Cerulli, M. Gaudioso, and M. Gentili. Lower and upper bounds for the spanning tree with minimum branch vertices. *Computational Optimization and Applications*, 56(2):405–438, 2013.

[5] F. Carrabs, R. Cerulli, R. Pentangelo, and A. Raiconi. Minimum spanning tree with conflicting edge pairs: a branch-and-cut approach. *Annals of Operations Research. To appear*, 2019.

[6] A. Chiarello, M. Gaudioso, and M. Sammarra. Truck synchronization at single door cross-docking terminals. *OR Spectrum*, 40:395–447, 2018.

[7] Ante Ćustić, Ruonan Zhang, and Abraham P. Punnen. The quadratic minimum spanning tree problem and its variations. *Discrete Optimization*, 27:73 – 87, 2018.

[8] A. Darmann, U. Pferschy, and J. Schauer. Determining a minimum spanning tree with disjunctive constraints. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5783 LNAI:414–423, 2009.

[9] A. Darmann, U. Pferschy, J. Schauer, and G.J. Woeginger. Paths, trees and matchings under disjunctive constraints. *Discrete Applied Mathematics*, 159(16):1726–1735, 2011.

[10] B. Dezső, A. Jüttner, and P. Kovács. LEMON – an Open Source C++ Graph Template Library. *Electronic Notes in Theoretical Computer Science*, 264:23–45, 2011. `http://lemon.cs.elte.hu/trac/lemon`.

[11] L. Di Puglia Pugliese, F. Guerriero, M. Gaudioso, and G. Miglionico. Decomposition approach for the link constrained steiner tree problem. *Optimization Methods and Software*, 33:650–670, 2018.

[12] M.L. Fisher, R. Jaikumar, and L.N. Van Wassenhove. A multiplier adjustment method for the generalized assignment problem. *Management Science*, 32(9):1095–1103, september 1986.

[13] A. Frangioni. About lagrangian methods in integer optimization. *Annals of Operations Research*, 139:163–193, 2005.

[14] M. Gaudioso, G. Giallombardo, and G. Miglionico. On solving the Lagrangian dual of integer programs via an incremental approach. *Computational Optimization and Applications*, 44:117–138, 2009.

[15] A. Geoffrion. Lagrangean relaxation and its uses in integer programming. *Math. Prog. Study*, 2:82–114, 1974.

[16] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex analysis and minimization algorithms Vol. I-II.* Springer-Verlag, 1993.

[17] M. M. Kanté, C. Laforest, and B. Momège. Trees in graphs with conflict edges or forbidden transitions. *Theory and Applications of Models of Computation: 10th International Conference, TAMC 2013, Hong Kong, China, May 20-22, 2013. Proceedings*, pages 343–354, 2013.

[18] A. Klein, D. Haugland, J. Bauer, and M. Mommer. An integer programming model for branching cable layouts in offshore wind farms. *Advances in Intelligent Systems and Computing*, 359:27–36, 2015.

[19] T. Öncan, R. Zhang, and A.P. Punnen. The minimum cost perfect matching problem with conflict pair constraints. *Computers and Operations Research*, 40(4):920–930, 2013.

[20] U. Pferschy and J. Schauer. The knapsack problem with conflict graphs. *Journal of Graph Algorithms and Applications*, 13:233–249, 2009.

[21] U. Pferschy and J. Schauer. The maximum flow problem with disjunctive constraints. *Journal of Combinatorial Optimization*, 26(1):109–119, 2013.

[22] B. T. Polyak. *Introduction to optimization*. Optimization Software Inc., 1987.

[23] B. Rostami and F. Malucelli. Lower bounds for the quadratic minimum spanning tree problem based on reduced cost computation. *Computers & Operations Research*, 64:178 – 188, 2015.

[24] R. Sadykov and F. Vanderbeck. Bin packing with conflicts: A generic branch-and-price algorithm. *INFORMS Journal on Computing*, 25(2):244–255, 2013.

[25] P. Samer and S. Urrutia. A branch and cut algorithm for minimum spanning trees under conflict constraints. *Optimization Letters*, 9(1):41–55, 2014.

[26] R. Zhang, S.N. Kabadi, and A.P. Punnen. The minimum spanning tree problem with conflict constraints and its variations. *Discrete Optimization*, 8(2):191–205, 2011.