RAIRO-Oper. Res. 48 (2014) 211–233
DOI: 10.1051/ro/2014008

RAIRO Operations Research
www.rairo-ro.org

INTEGER PROGRAMMING APPROACHES FOR MINIMUM STABBING PROBLEMS

Breno Piva[1,2], Cid C. de Souza[2], Yuri Frota[3] and Luidi Simonetti[3]

Abstract. The problem of finding structures with minimum stabbing number has received considerable attention from researchers. Particularly, [10] study the minimum stabbing number of perfect matchings (MSPM), spanning trees (MSST) and triangulations (MSTR) associated to set of points in the plane. The complexity of the MSTR remains open whilst the other two are known to be $\mathcal{NP}$-hard. This paper presents integer programming (IP) formulations for these three problems, that allowed us to solve them to optimality through IP branch-and-bound (B&B) or branch-and-cut (B&C) algorithms. Moreover, these models are the basis for the development of Lagrangian heuristics. Computational tests were conducted with instances taken from the literature where the performance of the Lagrangian heuristics were compared with that of the exact B&B and B&C algorithms. The results reveal that the Lagrangian heuristics yield solutions with minute, and often null, duality gaps for instances with several hundreds of points in small computation times. To our knowledge, this is the first computational study ever reported in which these three stabbing problems are considered and where provably optimal solutions are given.

Keywords. Integer Programming, Lagrangian Relaxation, Stabbing Problems, Branch-and-Bound, Branch-and-Cut.

Mathematics Subject Classification. 90C10, 90C27, 90C47, 90C57, 90C59.

Received September 11, 2013. Accepted November 28, 2013.

[1] Universidade Federal de Sergipe, Departamento de Computação, Cidade Universitária Prof. Aloísio Campos, 49100-000 São Cristóvão (SE), Brazil. bpiva@ufs.br

[2] Universidade Estadual de Campinas, Instituto de Computação, Av. Albert Einstein 1251, 13083-852 Campinas (SP), Brazil. cid@ic.unicamp.br

[3] Universidade Federal Fluminense, Instituto de Computação, R. Passo da Pátria 156, Bloco E, 24210-240 Niterói (RJ), Brazil. {yuri,luidi}@ic.uff.br

Article published by EDP Sciences

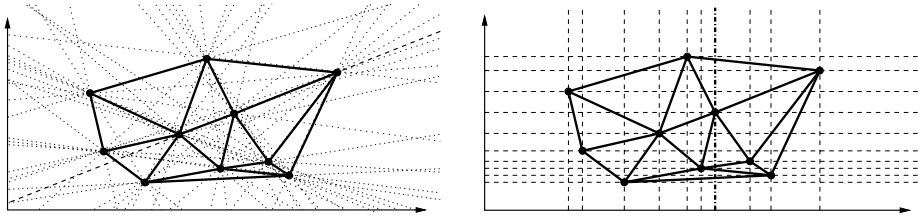

FIGURE 1. A triangulation with general (axis parallel) stabbing number 14 (9).

1. INTRODUCTION

Given a set of points P in the plane, the *geometric graph* associated to P is the graph $G(P) = (V, E)$ whose vertices are the points in P and whose edges are the straight line segments with both extremities in P. The *stabbing number of a line* ℓ passing through a geometric (sub)graph $G(P) = (V, E)$ is defined as the number of edges in E having a non-empty intersection with ℓ. Given a set L of straight lines, the *stabbing number of a (sub)graph* $G(P) = (V, E)$ is the maximum number of intersections between any line in L and the edges in E. The problem of finding a structure with minimum stabbing number can be defined for any kind of structure, *e.g.* Perfect Matchings, Spanning Trees, Triangulations *etc.* So, for example, the problem of finding the Minimum Stabbing Perfect Matching (MSPM) can be described as follows: given a set of points P, and a set of straight lines L, find a perfect matching in the geometric graph $G(P)$, among every possible perfect matchings in $G(P)$, having a stabbing number with minimum value. Two versions of the problem are presented in [9,10] and are related to the choice of the set L. In the first version, here referred as the *general stabbing* one, L is defined as the infinite set formed by all straight lines that can be drawn in the plane. In the *axis parallel* version, L is the, also infinite, set composed solely by the vertical and horizontal lines in the plane. Figure 1 illustrates the two versions of the problem with a triangulation of stabbing numbers 14 and 9, respectively.

Motivation. Stabbing problems have received considerably attention in the Computational Geometry community. In 2001 Mitchell and O'Rourke published a list with thirty open problems in the field [16], given rise to *The Open Problems Project* [6], containing a list of geometric problems whose complexity, at that time, was unknown. The list, which is constantly updated, is an invaluable source of challenging problems in Computational Geometry. In [9,10] general and axis parallel versions of the Minimum Stabbing Perfect Matching (MSPM), Minimum Stabbing Spanning Tree (MSST) – problem #20 of the aforementioned list – and Minimum Stabbing Triangulation (MSTR) were discussed. For the first two problems approximation algorithms were presented and $\mathcal{NP}$-hardness proofs were given for both versions of the problems. Computational results are presented for the MSPM. The

complexity status of MSTR could not be established and no algorithms were developed or tested to solve it. Heuristics for the spanning tree, perfect matching and triangulation stabbing problems were investigated in [17]. These heuristics are mostly based on greedy and divide-and-conquer techniques. Contrarily to the Lagrangian heuristics proposed here, they are not able to provide the duality gap associated to the solution they yield. In [17] the limited amount of information about computational experiments refers exclusively to the spanning tree case. Other works related to finding geometric structures with minimum or low stabbing number include [1, 4, 24, 26].

Our contribution. This paper presents two IP formulations for the MSTR based on the ideas described in [9, 10, 20] and one formulation for the MSST which explores the results given in [9, 10, 15]. Later, these formulations and a variation of the one described in [9, 10] for the MSPM are used to implement exact branch-and-bound (B&B) and branch-and-cut (B&C) algorithms for the corresponding problems, which allowed, for the first time in the literature, to obtain solutions with proven optimality. Besides, Lagrangian relaxation (LR) heuristics based on the IP models for the three problems are presented and appropriate subgradient methods are implemented. Computational results obtained by the Lagrangian algorithms are reported with instances taken from the literature and reveal that optimality or minute duality gaps are achieved in small computation times.

In the triangulation case, it was of paramount importance the realization of the relation existing between the Minimum Weight Triangulation (MWT) and the MSTR. This led to the development of strong IP models for the latter and also to the usage of effective algorithms to solve the MWT. As we will see later, such algorithms play an important role in our Lagrangian heuristic for MSTR.

Before continuing, we must observe that an early version of this paper appeared in the Proceedings of ISCO 2012 [22]. Thus, this work is to be seen as an extended and more complete version of that previous work.

Organization of the text. The remaining of this document is organized as follows. Section 2 presents IP models for the problems studied. Section 3 describes how to derive a LR heuristic for the problems from the IP models, whilst in Section 4 we present our computational results. At last, in Section 5 we draw some conclusions and indicate future research directions to be pursued.

2. Integer programming models

In the current section we present IP models for the three problems under consideration in this paper, where the model for the MSPM is extracted from [9, 10] and the models for the MSST and MSTR are based on the ideas presented in those papers. The formulations described here will be used in the implementation of exact B&B and B&C algorithms. Also, in Section 3, we show how to obtain LRs

for each problem using the models introduced in this section, and use them to produce primal and dual bounds for the true optimum.

Stabbing Perfect Matchings. We first present the model for the MSPM. We are given the sets P and L of points and stabbing lines, respectively, and E denotes the set of edges of the geometric graph $G(P)$. Variable k denotes the stabbing number and, therefore, must be minimized. Variable x_{ij} is set to 1 when the edge ij is in the solution and 0 otherwise.:

$$(MSPM) \quad z = \min k \tag{2.1}$$

subject to

$$\sum_{ij \in E} x_{ij} = 1, \quad \forall\, i \in P, \tag{2.2}$$

$$\sum_{ij \in E : i,j \in S} x_{ij} \leq (|S|-1)/2, \quad \forall\, S \subset P, |S| \text{ odd}, \tag{2.3}$$

$$\sum_{ij \in E : ij \bigcap s \neq \emptyset} x_{ij} \leq k, \quad \forall\, s \in L, \tag{2.4}$$

$$k \in \mathbb{Z}, x_{ij} \in \mathbb{B} \quad \forall\, ij \in E. \tag{2.5}$$

In this formulation, constraints (2.2) and (2.3) guarantee that the solution is a perfect matching. The first enforces each vertex to have degree one and the second – although, satisfied by any integral solution and, therefore, not strictly necessary for the correctness of the model – strengthens the linear relaxation, as proved by Edmonds [8]. The third class of inequalities is formed by the stabbing inequalities and they state that the sum of the variables corresponding to the edges intersecting a given line $s \in L$ must always be smaller or equal to the stabbing number, k. Notice that, as observed in [9, 10], in principle, this formulation in not finite since there are infinitely many stabbing lines. However, considering the axis parallel version, when sweeping a stabbing line in a direction d, the stabbing number only changes at a point of P. For this reason, we only need to look at a linear number of stabbing lines, thus, making the model finite. Following a similar reasoning, when considering the general version, we only need to look at a quadratic number of lines, namely, those defined by each pair of points in P.

Stabbing Spanning Trees. There are a number of known IP formulations for the Minimum Spanning Tree Problem (MST), including some that define the convex hull of the points corresponding to integer solutions. So, in order to decide which one should be used to build a formulation for the MSST, we first implemented three of the strongest formulations described in [15] for the MST. After a few computational tests, we observed that the directed cut formulation had the best

practical performance compared to the other alternatives. Hence, we decide to use this model as the basis for our MSST formulation described below.

Consider a digraph $D = (P, A)$, where A is the set of arcs connecting each pair of vertices in P, *i.e.*, for each edge $ij \in E$ there is a pair of arcs (i, j) and (j, i). We arbitrarily set a vertex r as the root of the tree. The notation $\delta^+(C)$ refers to the cutset directed out of vertex set C and $\delta^-(C)$ to the cutset directed into the vertex set C. The variable $y_{ij} = 1$ if the tree contains arc (i, j) when rooted at r and $x_{ij} = 1$ if one of the arcs (i, j) or (j, i) is in the tree with r as root. The relationship between y and x variables is established by constraint (2.9).

$$(MSST) \quad z = \min k \tag{2.6}$$

$$\text{subject to} \quad \sum_{(i,j)\in\delta^+(C)} y_{ij} \geq 1, \quad \forall\, C \subset V \text{ with } r \in C \tag{2.7}$$

$$\sum_{ij\in A} y_{ij} = |P| - 1, \tag{2.8}$$

$$y_{ij} + y_{ji} = x_{ij}, \quad \forall ij \in E \tag{2.9}$$

$$\sum_{ij\in E: ij \bigcap s\neq\emptyset} x_{ij} \leq k, \quad \forall\, s \in L. \tag{2.10}$$

$$y_{ij} \in \mathbb{B} \quad \forall (i, j) \in A \tag{2.11}$$

$$k \in \mathbb{Z}, x_{ij} \in \mathbb{B} \quad \forall\, ij \in E. \tag{2.12}$$

As before, part of the formulation is composed by a set of constraints ((2.7)−(2.9)) ensuring that the resulting solution is a geometric subgraph of the required type, in this case a spanning tree. The remaining constraints are stabbing inequalities (2.10), which have the same meaning as before. Constraint (2.8) guarantees that the solution has $|P| - 1$ arcs, as required in a directed spanning tree. Finally, constraints (2.7) enforces that the solution is a directed connected graph.

Stabbing Triangulations. Next, the ideas used in the models above and the IP models for the MWT that can be found in [20] form the point of departure to build the Edge and Triangle Stabbing models for the MSTR. The first of these two models is simpler and, for this reason, easier to use in a Lagrangian Relaxation algorithm. The second, although more complicated, provides better bounds and, therefore, was used in a exact B&B algorithm.

In the Edge Stabbing model (MSTE), P_H is the set of vertices on the convex hull of P; a *crossing set* (Cr) is defined as a maximal set of edges which are pairwise intersecting (endpoints excluded); the set of all crossing sets in $G(P)$ is denoted by S_{Cr}; for an edge $pq \in E$, $Cr(pq)$ denotes the set of edges intersecting pq (again with endpoints excluded) plus pq itself; the rest of the notation stands

for the same as before. For every $ij \in E$, $x_{ij} = 1$ if and only if the edge ij is in the triangulation. The variable k, once again, denotes the stabbing number. Then, the Edge Stabbing Model reads:

$$\text{(MSTE)} \quad z = \min k \tag{2.13}$$

$$\text{subject to} \quad \sum_{ij \in E} x_{ij} = 3|P| - |P_H| - 3, \tag{2.14}$$

$$\sum_{ij \in Cr} x_{ij} \leq 1, \quad \forall\, Cr \in S_{Cr}, \tag{2.15}$$

$$\sum_{ij \in Cr(pq)} x_{ij} \geq 1, \quad \forall\, pq \in E, \tag{2.16}$$

$$\sum_{ij \in E : ij \bigcap s \neq \emptyset} x_{ij} \leq k, \quad \forall\, s \in L, \tag{2.17}$$

$$k \in \mathbb{Z}, x_{ij} \in \mathbb{B} \quad \forall\, ij \in E. \tag{2.18}$$

In this model, (2.14) guarantees that the solution has the right number of edges required for a triangulation of P. Constraint (2.15) states that only one edge in a crossing set can be in the solution, thus, ensuring planarity. Constraint (2.16) states that, either pq or at least one of the edges in $Cr(pq)$ must be in the solution, therefore, enforcing maximality (recall that a triangulation is a maximal planar subgraph of $G(P)$). It is noteworthy that constraint (2.16) is not strictly necessary for the formulation. However, as observed in [20], it greatly enhances the computational performance of the IP algorithms. Constraint (2.17) states that, for each stabbing line s in L, the number of edges from triangulation that intersect s is bounded from above by the stabbing number.

Another way to represent a triangulation using IP is to assign variables to the set of triangles with vertices in P. This idea was discussed in [5,20], where it was shown that the dual bounds generated by the relaxation of the resulting IP dominate those produced by the previous formulation on edge variables. In the description of the Triangle Stabbing Model below, $\Delta(P)$ is the set of empty triangles over P, *i.e.*, triangles that do not contain any point P in their interior; $L^+(ij)$ and $L^-(ij)$ are the two half-planes defined by the line containing ij; E_H is the set of edges on the convex hull of P. For every triangle $ijl \in \Delta(P)$, $x_{ijl} = 1$ if and only if the triangle ijl is in the triangulation. The variable k has the same meaning as in the previous models.

$$\text{(MSTT)} \quad z = \min k \tag{2.19}$$

$$\text{subject to} \quad \sum_{\substack{ijl \in \Delta(P) : \\ ijl \subset L^+(ij)}} x_{ijl} = \sum_{\substack{ijl \in \Delta(P) : \\ ijl \subset L^-(ij)}} x_{ijl}, \quad \forall ij \in E \setminus E_H, \tag{2.20}$$

$$\sum_{ijl \in \Delta(P)} x_{ijl} = 1, \quad \forall\, ij \in E_H, \tag{2.21}$$

$$\sum_{ijl \in \Delta(P) : ijl \bigcap s \neq \emptyset} c^s_{ijl} x_{ijl} \leq k, \quad \forall\, s \in L. \tag{2.22}$$

$$k \in \mathbb{Z}, x_{ijl} \in \mathbb{B} \quad \forall\, ijl \in \Delta(P). \tag{2.23}$$

In the model above, constraint (2.20) states that the number of triangles containing an edge ij (which is not in E_H) must be the same in both half-planes defined by the line containing ij. As the edges in E_H are present in every planar triangulation, constraint (2.21) ensures that a triangle containing one such edge is in the triangulation. Constraint (2.22) states that the sum of the coefficients c^s_{ijl} of the triangles ijl intersecting a line s of L can not be larger than the stabbing number. A triangle ijl intersecting a line s has coefficient $c^s_{ijl} = \beta^s_{ij} + \beta^s_{il} + \beta^s_{jl}$, where $\beta^s_{ij} = 1$ if ij intersects s and is on the convex hull, $\beta^s_{ij} = 0.5$ if ij intersects s but is not on the convex hull and $\beta^s_{ij} = 0$ if ij does not intersect s.

Later we will see that both models presented in this section for the MSTR are used in our implementations: (MSTT) in the B&B (exact) algorithm and (MSTE) in the Lagrangian heuristic.

3. Lagrangian relaxation

Using the IP formulations from the previous section, we now derive Lagrangian relaxation (LR) models for the three stabbing problems. We solve the dual of this relaxation *via* the subgradient method (SGM), which allows us to obtain a lower bound for the optimal value of the problems. Besides, at each iteration of the SGM, we compute the primal Lagrangian problem whose solution is a minimum perfect matching, spanning tree and triangulation, respectively for the MSPM, MSST and MSTR, and, thus, can be used to obtain upper bounds for these problems. For the basic theory of Lagrangian relaxation the reader is referred to [27].

The presentation of our LR is based on a model for a generic stabbing problem (STAB), presented below. This model is composed by the generic constraints (3.2) that define the form of the subgraph of $G(P)$ to be found (in our case either a perfect matching, a spanning tree or a triangulation) and the constraints (3.3) which define that the stabbing number of the subgraph is greater than or equal to the stabbing number of any line.

$$\text{(STAB)} \qquad z = \min k \tag{3.1}$$

subject to

$$Ax \leq B, \tag{3.2}$$

$$\sum_{ij \in E : ij \bigcap s \neq \emptyset} x_{ij} \leq k, \qquad \forall\, s \in L, \tag{3.3}$$

$$k \in \mathbb{Z}, x_{ij} \in \mathbb{B} \qquad \forall\, ij \in E. \tag{3.4}$$

To obtain the LR (STAB(u)) of problem (STAB) we simply dualize the constraints (3.3), penalizing them in the objective function. This operation results in the following model for the Lagrangian primal problem:

$$\text{(STAB}(u)\text{)} \quad z(u) \;=\; \min k - \sum_{s\in L} u_s \left(k - \sum_{ij\in E: ij \bigcap s\neq\emptyset} x_{ij} \right) \tag{3.5}$$

subject to

$$Ax \;\leq\; B, \tag{3.6}$$

$$k \in \mathbb{Z}_+, x_{ij} \in \mathbb{B} \qquad \forall\; ij \in E. \tag{3.7}$$

Notice that the constraints (3.2) that remain in the model are those that define the subgraphs of interest. Also, since the constraints being dualized are in the "$\leq$" form, u_s is non-negative for all $s \in L$. As a consequence, the Lagrangian primal problem is equivalent to the problem of finding one such subgraph having minimum weight (the weight of the subgraph being defined as the sum of its edge weights). In the Lagrangian case, the weight of edge ij is given by

$$c_{ij} = \sum_{s\in L: s \bigcap ij\neq\emptyset} u_s. \tag{3.8}$$

From the Lagrangian theory, we know that whenever the primal problem can be solved in polynomial time, as is the case for the MSPM and MSST, we are able to obtain a dual bound for the original problem in short computation times. However, when the primal problem is $\mathcal{NP}$-hard, one may wonder if the relaxation is useful after all. This is precisely the situation with the MSTR since the MWT was proven to be $\mathcal{NP}$-hard in [19]. However, as we shall see later in Section 4, there are highly effective algorithms to compute large subsets of optimal MWT solutions. As a result, one can expect to solve instances of the MWT with several hundreds of points very quickly. Our approach relies on this observation and the results reported in this paper confirmed our expectations.

Now, as (STAB(u)) is a relaxation of (STAB), we know that $z(u) \leq z$ and, since we want to find the best possible bound, we must find the value of u that maximizes $z(u)$, *i.e.*, we must solve the Lagrangian dual problem given by

$$(DL) \;\; v_{DL} = \max\{z(u) : u \geq 0\}. \tag{3.9}$$

Problem (DL) can be solved using the SGM as described in [2,27]. To this end, the multipliers u_s are initialized with null values and are updated at iteration t by the formula:

$$u_s^t = \max(0, u_s^{t-1} - \mu G_s^{t-1}). \tag{3.10}$$

with μ given by

$$\mu = \frac{\pi(dist \times ub - lb)}{\sum_{s\in L}(G_s^{t-1})^2}, \tag{3.11}$$

and G_s^{t-1}, the sth component of the subgradient of $z(u)$ in u^{t-1}, given by

$$G_s^{t-1} = k - \sum_{ij\in E: ij \bigcap s\neq\emptyset} x(u^{t-1})_{ij}. \tag{3.12}$$

In the formulas above, ub and lb are, respectively, an upper and a lower bound for the optimal value, $dist$ is a perturbation factor (arbitrarily set to 1.05 in our experiments) and π is the step size (in our experiments initialized at 2 and halved every 30 iterations without improvement in the lower bound). The solution of the Lagrangian primal problem is denoted by $x(u)$ and the superscripts indicate the iteration at which each variable is been considered (*e.g.*, u^t is the Lagrangian multipliers vector at iteration t).

Now, notice that, after dualizing constraints (3.3), the objective function of (STAB(u)) can be rewritten as:

$$z(u) = \min k \left(1 - \sum_{s \in L} u_s\right) + \sum_{ij \in E} x_{ij} \sum_{s \in L : s \bigcap ij \neq \emptyset} u_s. \tag{3.13}$$

Therefore, if $\sum_{s \in L} u_s > 1$, the first term of that equation would have a negative value and, hence, the larger the value of k, the smaller the value of $z(u)$. As a result, when optimizing the (primal) Lagrangian problem, if the cost of variable k is negative, the lower bound $z(u)$ is unlimited and hence useless. Analogously, if the cost of k is non negative, the obvious solution is to set k to zero. However, by doing so, we may waste the opportunity to produce a better dual bound for z. To overcome these situations, we proceed in the following way. In the solution of (STAB(u)), k is set, respectively, to the best upper (ub) or lower (lb) bound available for z depending on whether its cost is negative or not. In fact, in our implementation, when the cost is non negative, k is set to $\lceil lb \rceil / 2$ rather than to lb to avoid an early convergence of the SGM. This tends to increase the number of iterations of the method, augmenting the chances of the Lagrangian heuristic to obtain a better feasible solution.

Notice that the dual bound obtained by setting k to $\lceil lb \rceil / 2$ or ub, depending on whether $(1 - \sum_{s \in L} u_s)$ is negative or non-negative, is valid. This is so because the model for the primal Lagrangian problem remains correct if the constraint requiring that k belongs to $\mathbb{Z}_+$ is replaced by one that forces k to be in an interval between proper lower and upper bounds. It turns out that $\lceil lb \rceil / 2$ and ub are, respectively, valid lower and upper bounds for k, ensuring the correctness of the computation of the dual bounds for $z(u)$.

The termination criteria implemented in our SGM are achieved when one of the following situations occur: the difference between the upper and lower bounds is smaller than 1 (one), the value of π is smaller than 0.005, or yet, a predefined time limit is reached.

Lagrangian Heuristic. Each iteration of the SGM solves a minimum weight problem (a MWPM, a MST, or a MWT, whichever is the case). The solution of this problem is a subgraph of $G(P)$ satisfying the property of interest (*i.e.*, it is a perfect matching, a spanning tree, or a triangulation) and, therefore, is also feasible for the original stabbing problem. Thus, an upper (primal) bound for the optimal value

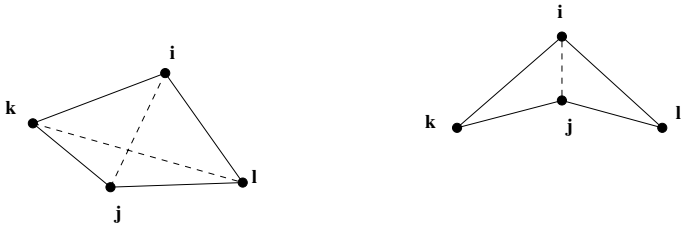

FIGURE 2. In both cases ij is locally minimum with respect to the quadrilateral $ijkl$.

of the stabbing problem can be immediately obtained by computing the stabbing number of this subgraph.

Solving the Lagrangian Primal. For the MSTR, (STAB(u)) corresponds to a MWT. As cited before, the MWT is known to be $\mathcal{NP}$-hard but there are algorithms to find subsets of optimal solutions. One of these algorithms is the one to find a Locally Minimum Triangulation Skeleton (LMT-skeleton) [3, 7]. This algorithm is based on the local minimality property of line segments (edges).

Given a planar triangulation T, let ij be an edge of T that is not in the convex hull. Then, ij must be the side of two empty triangles ijk and ijl in T. These two triangles together form a quadrilateral $ijkl$ having ij and kl as its diagonals. We say that ij is locally minimum with respect to $ijkl$ if this quadrilateral is not convex or, else, if the weight of ij is smaller than the weight of kl. Figure 2 illustrates this definition. If for any pair of points $\{k, l\}$ in $P - \{i, j\}$ the edge ij is locally minimum with respect to the quadrilateral $ijkl$, then ij is said to be locally minimum. When all the edges in a planar triangulation are locally minimum, we say that the triangulation itself is locally minimum. Clearly, any minimum weight triangulation is locally minimum. However, not all locally minimum triangulations have minimum weight. The LMT-skeleton is the subset of edges that are present in every locally minimum triangulation and, thus, is also a subset of any minimum weight triangulation.

In [7] the authors proposed a polynomial algorithm to find a LMT-skeleton and in [3] the algorithm was improved. The computational experiments performed with these algorithms showed that, together with a dynamic programming algorithm to find a MWT for convex polygons, it was capable to find the MWT of instances with thousands of points in quite small running times. The source code for this last algorithm written by Mulzer is available online at [18].

Therefore, we can make use of the LMT-skeleton algorithm to solve the Lagrangian Primal Problem through the following steps. First we determine three subsets T_m, T_p and T_f of edges which, respectively, are mandatory (the locally minimum ones), forbidden (those intersected by an edge in T_m) and uncertain (the remaining edges) in a optimal solution, using a LMT-skeleton algorithm [3, 7].

Then, we are left with a constrained MWT problem where all edges of T_m are forced to be in the solution, the ones in T_f are eliminated from the solution and those in T_p are the ones for which we have to make a decision. Typically, after fixing the appropriate variables to one or zero, the size of the MWT models reduces dramatically. This renders the usage of an IP solver to compute the model *via* a standard B&B algorithm a viable option, even for instances containing hundreds of points. Later we will see that this procedure is capable to solve the Lagrangian primal problems for MSTR in an extremely effective fashion in practice.

To conclude this section, we recall that the Lagrangian primal problems for the MSPM and MSST are, respectively, the MWPM and the MST. To solve the first one we use the Blossom V algorithm described in [14], whose source code is publicly available. The MST problem is solved by a simple implementation of Prim's algorithm, which can be found in several textbooks on Algorithms.

4. Computational results

We now describe the experiments we carried out to test the performance of the algorithms discussed in the previous sections. As mentioned earlier, we implemented exact B&C algorithms for the MSPM and MSST. An implementation of an exact B&B algorithm for the MSTR was also done. All these exact algorithms were based on the IP models discussed in Section 2. We also implemented LR algorithms for all the models using the ideas discussed in Section 3. All the experiments described in this section consider the axis parallel version of the problem.

Computational Environment. To perform the experiments, we used a computer with an Intel Core 2 Quad 1.60GHz, 4096 KB cache, 4GB of RAM memory and a Ubuntu 10.04.4 OS. The programming language used was `C/C++` with `gcc` 4.4.3 compiler and every program was compiled with `-O5` optimization flag. We also used the `XPRESS-Optimizer 64-bit v22.01.09` IP solver. The default cuts, heuristics and preprocessing were turned off. Also, the optimizer was set to use a single processor core.

4.1. MSPM Experiments

In order to evaluate the performance of our algorithms for the MSPM, we executed experiments with both, the exact B&C algorithm and the LR algorithm and then we tried to compare the results, although this kind of comparison is sometimes tricky, since the algorithms are different in nature.

For the exact B&C algorithm the model was initially loaded using only the degree inequalities (2.2) and stabbing inequalities (2.4). The heuristic proposed in [12] was implemented to separate violated inequalities (2.3). Only when the heuristic fails to find a cutting plane, we resort to the Padberg-Rao exact algorithm described in [21]. We also use a family of conditional cuts [11] that are not guaranteed to be valid for the problem, but can be used as a cutting plane as follows. Suppose

an upper bound U_b of the problem is available. One can note that during the search for the optimal solution of the MSPM, we are looking for solutions of value better (lesser) than U_b. In this sense, any inequality can be used as a cutting plane, provided that is satisfied by every feasible solution of value less than U_b. In this vein, we considered the following family of conditional cuts:

$$\sum_{ij \in E[V_s^+]} x_{ij} \geq \left\lceil \frac{|V_s^+| - U_b + 1}{2} \right\rceil, \quad \forall s \in L, \tag{4.1}$$

$$\sum_{ij \in E[V_s^-]} x_{ij} \geq \left\lceil \frac{|V_s^-| - U_b + 1}{2} \right\rceil, \quad \forall s \in L, \tag{4.2}$$

where V_s^+ and V_s^- are sets composed by vertices of V in the interior of one of the two half-planes defined by the line s. Besides, the sets $E[V_s^+]$ and $E[V_s^-]$ are formed by all the edges with both endpoints in V_s^+ and V_s^-, respectively. It can be seen in inequalities (4.1) that a solution of value U_b has at most U_b edges crossing s (each one connected with a vertex in V_s^+). Hence, there are $(|V_s^+| - U_b)$ disconnected vertices in V_s^+ that need $\lceil (|V_s^+| - U_b)/2 \rceil$ edges in $E[V_s^+]$ to complete a matching. Then, it follows that (4.1) can be used as a conditional cut because no solution of value U_b (or greater) is feasible in (4.1). Similar arguments lead to an analogous conclusion for inequalities (4.2).

The cutting plane strategy adds the inequalities with the highest percentage of violation, as long as this value is at least 1% (to control the tailing off effect). No more than 50 inequalities are added per iteration. As for the branching strategy, we select 5 variables whose values in the current linear relaxation are closest to 0.5 and use strong branching to select which variable to branch on.

The primal heuristic used in B&C is based on the linear relaxation of the problem. From a relaxed solution $\overline{x}$, the method attempts to find a matching $M \subseteq E$ maximizing $\sum_{ij \in M} \overline{x}_{ij}$. The method begins with an empty set M and builds a matching, one edge at a time. At each iteration, one edge $(i, j) \in E \backslash M$ is greedily chosen according to the value of $\overline{x}_{ij}$ (prioritizing the highest ones) and inserted into M. The procedure is repeated until a perfect matching is reached. In a second phase, the matching M may be improved by a local search procedure. The neighborhood of the current solution M is defined as the set of all feasible matchings obtained by exchanging pairs of edges (i, j) and (l, m) by edges (i, l) and (j, m). The procedure iteratively replaces the current solution by the one with minimum cost within its neighborhood, halting when no better solution is found in that way. This primal heuristic is applied at every node of the search tree.

For the LR algorithm, a Lagrangian relaxation of the model described for the MSPM in Section 2 is obtained (see Sect. 3). The standard subgradient method is then executed to compute the Lagrangian dual problem. As said before, the Lagrangian primal problem is solved by an implementation of the Blossom V algorithm whose code is available for download in the web. It is worth noting that this program only deals with instances having integer weights. However, in the

usual Lagrangian scheme, the edge weights are often not integer. To circumvent this difficulty, we multiplied all the edge weights in the Lagrangian primal problem by 10^6 before calling the routine. This is not expected to create major numerical problems and, in the end, is not more harmful to computation than the tolerance of 10^{-6} that we set for the IP solver.

As we will see in the results part of this subsection, the Lagrangian algorithm produces good bounds with small computation times. This suggests that it can be used together with the exact B&C algorithm to obtain better results. We used the primal bound from the LR algorithm to warm start the B&C algorithm. Our tests showed that, for the three problems studied, the use of primal bounds from LR algorithm to warm start the exact algorithms yielded better overall results. For this reason, we decided to use these results and compare them with the pure Lagrangian results.

Instances. For the MSPM, we experimented with the same instances tested in [9] (except for five TSPLIB instances [23] that are obviously infeasible since they have an odd number of vertices). These include 5 instances from TSPLIB, 16 from the clustered C1 and C2 classes of Solomon's Vehicle Routing Problem benchmark [25], 25 regular grid instances (5×5 to 20×20 grids with 20% of its points randomly removed) and 11 instances with up to 100 random points in the plane.

For the three problems under investigation, a time limit of $1,800$ seconds was set for the execution of any algorithm. Notice, however, that in the Tables 4 to 7, occasionally the time is bigger than this limit. This happens for two reasons, first, the times presented for warm started exact algorithm (WSEA) are the sum of the time spent by the Lagrangian and the B&C or B&B algorithms, therefore could go up to 3600. Second, the time limit is verified at certain points in the program codes and, it could be that the time elapsed between two check-points is not negligible. This situation arises, for example, when the model of a big instance is being uploaded by the IP solver. In our experiments an additional timeout script running on the operating system level was used that forces the process to halt after 2000 sec. In case the process ends naturally, a bound is always produced. On the other hand, if the process is killed by the timeout script, no output is produced. The latter situation is signalized in the tables by the symbol ‡. Also, duality gaps were computed through the formula $100 \times (ub - lb)/ub$, where ub and lb denote, respectively, the upper and lower bounds yielded by the algorithm.

Results. As we previously stated, all the WSEA outperformed the cold started exact algorithms and, for this reason, we compare the WSEA against the LR algorithms. Obviously, it does not make sense to just compare the times of these two kinds of algorithms because, first, as said before, the time of the WSEA is the sum of the LR algorithm and the B&C or B&B algorithm, thus, is always greater than the LR alone. Second, the algorithms are different in nature. So, the purpose of our comparison is to determine whether the WSEA can improve the bounds obtained by the LR algorithm, how much and how fast.

TABLE 1. Results for MSPM TSP and clustered instances.

Instance	LB		UB		Time		GAP%	
	LR	B&C	LR	B&C	LR	B&C	LR	B&C
a280	11	11	11	11	0.83	13.34	0.00	0.00
berlin52	3	4	4	4	0.86	1.23	25.00	0.00
lin318	9	9	9	9	29.17	52.43	0.00	0.00
pcb442	17	17	17	17	27.71	78.79	0.00	0.00
ulysses22	2	2	2	2	0.00	0.03	0.00	0.00
c101	7	7	7	7	0.05	0.41	0.00	0.00
c102	7	7	7	7	0.05	0.43	0.00	0.00
c103	7	7	7	7	0.05	0.41	0.00	0.00
c104	7	7	7	7	0.05	0.43	0.00	0.00
c105	7	7	7	7	0.06	0.43	0.00	0.00
c106	7	7	7	7	0.05	0.42	0.00	0.00
c107	7	7	7	7	0.05	0.43	0.00	0.00
c108	7	7	7	7	0.06	0.42	0.00	0.00
c201	6	6	6	6	0.08	0.55	0.00	0.00
c202	6	6	6	6	0.09	0.54	0.00	0.00
c203	6	6	6	6	0.09	0.55	0.00	0.00
c204	6	6	6	6	0.08	0.55	0.00	0.00
c205	6	6	6	6	0.09	0.55	0.00	0.00
c206	6	6	6	6	0.08	0.53	0.00	0.00
c207	6	6	6	6	0.08	0.53	0.00	0.00
c208	4	4	4	4	1.15	1.89	0.00	0.00

Our analysis of the results will be done in three parts: the first for the TSP and clustered instances, the second for the random instances and the third for the grid instances.

The results for the first set of instances are summarized in Table 1. We observe that the B&C algorithm proved optimality in all the cases within the fixed time limit. The Lagrangian SGM always converged, proving optimality in all but one case (berlin52), where there is an absolute gap of one unit (25.0%). For this set of instances the WSEA provided an average improvement of 1.19% in the relative gap with an average increasing of 4.48 sec in time when compared to the LR algorithm.

Results for the random instances can be seen in Table 2. Once again the LR algorithm always converged. However, whilst the exact algorithm proves optimality for all instances, the Lagrangian failed to prove optimality in four cases, where gaps of one unit remain. The average improvement in the relative gap obtained from the WSEA was 8.64% and the average time increasing was 1.74 sec.

The results for the grid instances are displayed in Table 3. This benchmark was the one for which the LR heuristic had the worst performance. The Lagrangian heuristic was unable to prove optimality in 11 out of 25 cases, leaving gaps of one unit in 10 cases and two units in 1 case. The exact algorithm, on the other hand, was able to prove optimality for all of the grid instances. The improvement

TABLE 2. Results for MSPM random instances.

Instance	LB		UB		Time		GAP%	
	LR	B&C	LR	B&C	LR	B&C	LR	B&C
rand10a	2	2	2	2	0.00	0.00	0.00	0.00
rand10b	2	2	2	2	0.00	0.00	0.00	0.00
rand10c	2	2	2	2	0.00	0.00	0.00	0.00
rand10d	2	2	2	2	0.00	0.01	0.00	0.00
rand10e	2	2	2	2	0.00	0.01	0.00	0.00
rand50a	3	3	3	3	0.15	0.67	0.00	0.00
rand50b	3	3	3	3	0.64	1.18	0.00	0.00
rand50c	3	4	4	4	0.62	1.20	25.00	0.00
rand50d	3	4	4	4	0.64	1.15	25.00	0.00
rand50e	3	4	4	4	0.77	1.32	25.00	0.00
rand100a	4	5	5	5	6.40	22.85	20.00	0.00

in the relative gap achieved using the exact algorithm was 4.85% and the average increasing of time was 8.95 sec.

Therefore it is possible to say that the LR algorithm have a very nice performance for these sets of instances. Also, the price in time necessary to prove optimality using the warm started B&C algorithm seems rather small. We recall that B&C is an exact algorithm while LR is an heuristic. So, when comparing their performances, one has to bear in mind that they are rather different in nature.

In order to compare our results against those presented in [9] we implemented the model presented in that paper and executed a B&C algorithm in the same computational environment used to test ours. This experiment showed that the algorithm using the model from [9] was unable to prove optimality in six, cases among all the instances tested for the MSPM, within a time limit of 1800 sec. Considering all the test cases for the MSPM, the average time of our WSEA was 5.91 sec while the implementation of the algorithm from [9] had an average time of 213.10 sec.

4.2. MSST EXPERIMENTS

To analyze the performance of our algorithms for the MSST, again we implemented an exact B&C algorithm. Once more, we found that warm starting the B&C algorithm with the primal bound obtained from the Lagrangian SGM gives us better results than simply executing the B&C. Therefore, all comparisons in this subsection are made between the WSEA and the LR algorithm.

For the exact algorithm we used the model described in Section 2. Initially the model was loaded without constraints (2.7). In the branch-and-cut method, at each node of the search tree, the linear relaxation of MSST is solved. If in the optimal solution all variables are integral, the node is pruned by optimality. Otherwise, the solution is fractional and violated valid inequalities are sought by solving a separation problem. The polynomial-time algorithm presented in [13], based on

TABLE 3. Results for MSPM grid instances.

Instance	LB		UB		Time		GAP%	
	LR	B&C	LR	B&C	LR	B&C	LR	B&C
grid5a	4	4	4	4	0.00	0.01	0.00	0.00
grid5b	4	4	4	4	0.00	0.01	0.00	0.00
grid5c	4	4	4	4	0.01	0.02	0.00	0.00
grid5d	4	4	4	4	0.00	0.01	0.00	0.00
grid5e	4	4	4	4	0.00	0.02	0.00	0.00
grid8a	6	6	6	6	0.10	0.15	0.00	0.00
grid8b	6	6	6	6	0.06	0.12	0.00	0.00
grid8c	5	5	6	5	0.19	0.28	16.67	0.00
grid8d	6	6	6	6	0.00	0.06	0.00	0.00
grid8e	6	6	7	6	0.30	0.35	14.29	0.00
grid10a	7	7	7	7	0.22	0.43	0.00	0.00
grid10b	6	6	7	6	0.64	0.83	14.29	0.00
grid10c	7	7	8	7	0.69	2.04	12.50	0.00
grid10d	7	7	7	7	0.19	0.41	0.00	0.00
grid10e	7	7	8	7	0.59	1.73	12.50	0.00
grid15a	10	10	10	10	1.59	3.61	0.00	0.00
grid15b	10	10	11	10	5.45	50.42	9.09	0.00
grid15c	10	10	10	10	1.32	3.28	0.00	0.00
grid15d	10	10	10	10	2.94	4.96	0.00	0.00
grid15e	10	10	10	10	1.77	4.04	0.00	0.00
grid20a	13	13	15	13	25.65	111.31	13.33	0.00
grid20b	13	13	14	13	26.28	40.70	7.14	0.00
grid20c	13	13	14	13	28.16	47.46	7.14	0.00
grid20d	13	13	14	13	24.06	39.43	7.14	0.00
grid20e	13	13	14	13	31.02	63.31	7.14	0.00

the minimum edge cut problem in graphs, is used to separate the Steiner cut inequalities (2.7).

As for the LR algorithm, the implementation was done as described in Section 3, with the primal Lagrangian problem been solved by a simple implementation of Prim's algorithm for the MST.

Instances. As a test suite we used 25 instances from TSPLIB [23] and the 25 regular grid instances used in [9] for the Minimum Stabbing Perfect Matching Problem. The choice of these instances is based on the fact that the TSPLIB is a well known test library for geometric problems and, besides, some TSPLIB and all grid instances were also used in [9] for the MSPM. The choice of the instance sizes was made seeking tests that were hard enough to provide meaningful computation times, allowing a more precise comparison of the algorithms.

Results. We divide our analysis into two parts, one for the TSP instances and another for the grid instances.

TABLE 4. Results for MSST TSP instances.

Instance	LB		UB		Time		GAP%	
	LR	B&C	LR	B&C	LR	B&C	LR	B&C
berlin52	6	6	6	6	0.15	3.77	0.00	0.00
ch130	7	7	8	8	12.21	1813.38	12.50	12.50
ch150	8	8	9	8	19.35	161.09	11.11	0.00
eil76	8	8	8	8	1.08	1.48	0.00	0.00
gil262	11	11	12	12	83.45	1907.68	8.33	8.33
gr202	9	9	10	9	58.70	1456.22	10.00	0.00
kroA100	7	7	8	7	4.85	1177.36	12.50	0.00
kroA150	8	8	9	9	14.69	1819.08	11.11	11.11
kroA200	9	9	9	9	29.95	1154.45	0.00	0.00
kroB100	7	7	7	7	3.98	5.20	0.00	0.00
kroB150	8	8	9	9	19.81	1823.96	11.11	11.11
kroB200	9	9	10	10	45.91	1858.87	10.00	10.00
kroC100	7	7	7	7	4.21	46.09	0.00	0.00
kroD100	7	7	7	7	3.27	4.40	0.00	0.00
kroE100	7	7	7	7	2.67	3.91	0.00	0.00
lin318	16	16	18	18	36.84	1860.34	11.11	11.11
pcb442	34	33	34	34	56.02	1915.33	0.00	2.94
pr124	24	24	24	24	22.47	26.06	0.00	0.00
pr136	17	17	18	17	2.75	87.52	5.56	0.00
pr144	21	21	21	21	0.50	1292.64	0.00	0.00
pr152	11	11	12	11	6.88	536.45	8.33	0.00
pr226	72	72	72	72	4.43	16.54	0.00	0.00
pr264	23	23	29	29	13.93	1821.02	20.69	20.69
rd100	7	7	8	7	4.98	247.18	12.50	0.00
rd400	11	‡	13	13	661.39	‡	15.38	‡

The results for the TSP part are displayed in Table 4. One can see that the LR algorithm converged in all the cases within the time limit, proving optimality in 11 of the 25 of them. The WSEA was unable to yield any output within the time limit for just one of the test instances. Among the 24 remaining instances, the B&C algorithm proved optimality in 16 cases. It is interesting to notice that the SGM was able to prove optimality in one case where the B&C was unable to do so (despite the warm start), while the opposite occurred 6 times. For this set of instances, when compared with the LR algorithm, the improvement in the relative gap provided by the WSEA was 2.38% and the necessary extra time to achieve this improvement was 857.79 sec.

Analyzing the results for the second group of instances given in Table 5, we observe that the performance of the LR algorithm is not as good as for the TSP instances, since optimality was achieved in fewer cases. The B&C failed to declare optimality in only 3 out of the 25 grid instances while the SGM failed in 14 other cases. In the grid instances, the execution of the WSEA improved the relative gap by 4.59% at the cost of 391.88 more seconds, both in average.

TABLE 5. Results for MSST grid instances.

Instance	LB		UB		Time		GAP%	
	LR	B&C	LR	B&C	LR	B&C	LR	B&C
grid5a	7	7	7	7	0.01	0.10	0.00	0.00
grid5b	7	7	7	7	0.01	0.10	0.00	0.00
grid5c	7	7	7	7	0.01	0.09	0.00	0.00
grid5d	7	7	7	7	0.01	0.09	0.00	0.00
grid5e	7	7	7	7	0.01	0.09	0.00	0.00
grid8a	10	10	10	10	0.04	1.57	0.00	0.00
grid8b	10	10	10	10	0.03	0.19	0.00	0.00
grid8c	10	10	10	10	0.07	0.22	0.00	0.00
grid8d	11	11	13	11	0.15	1.10	15.38	0.00
grid8e	11	11	11	11	0.08	0.24	0.00	0.00
grid10a	13	13	14	13	0.44	4.31	7.14	0.00
grid10b	12	12	12	12	0.17	0.44	0.00	0.00
grid10c	13	13	14	13	0.45	3.78	7.14	0.00
grid10d	13	13	13	13	0.18	0.48	0.00	0.00
grid10e	13	13	14	13	0.47	9.17	7.14	0.00
grid15a	18	18	20	18	2.97	117.97	10.00	0.00
grid15b	20	20	23	20	3.17	368.78	13.04	0.00
grid15c	18	18	19	18	2.87	84.31	5.26	0.00
grid15d	19	19	21	19	2.35	125.61	9.52	0.00
grid15e	18	18	20	18	2.44	828.30	10.00	0.00
grid20a	24	24	27	27	15.48	1828.94	11.11	11.11
grid20b	24	24	27	27	11.16	1824.14	11.11	11.11
grid20c	25	25	28	25	11.06	1415.05	10.71	0.00
grid20d	25	25	29	29	9.98	1827.44	13.79	13.79
grid20e	25	25	31	25	11.95	1430.14	19.35	0.00

The analysis of the improvement relative to the Lagrangian SGM algorithm and of the additional time spent to obtain such gain when using WSEA points to a remarkable performance of the LR algorithm.

4.3. MSTR EXPERIMENTS

The first stage of our testing comprised a comparison of the two alternative B&B algorithms that arise from the Edge and Triangle stabbing models discussed in Section 2. For the MWT, it was observed in [20] that the B&B algorithm performs better when it uses an IP model with variables defined on triangles than with variables associated to edges. Hence, a similar behavior was expected from the corresponding models when applied to the solution of the MSTR. Indeed, this was what happened and, thus, all the B&B results reported below were obtained using the Triangle Stabbing Model. More precisely, the results refer to a warm started exact algorithm (WSEA) using the mentioned formulation.

Regarding the LR algorithm, we implemented the subgradient method using both the Edge Stabbing Model and the Triangle Stabbing Model. Recall that,

irrespective to which of the two models we consider, when the stabbing constraints are relaxed we are left with an IP formulation for the MWT problem (we use the term "relaxed" to refer to these models). However, in the subgradient procedure several such problems have to be solved at each iteration. This is done in two steps. The first step consists in the calculation of the LMT-skeleton while the second step actually solves the MWT problem in case the first step fails to do so.

Observe that the edge weights are the only differences between the instances of the MWT problems solved in two iterations of the subgradient method. The computation of the LMT-skeleton only depends on the edge costs. Therefore, for the first step, it is convenient from a computational point of view to have the problem defined in terms of the Edge Stabbing Model, as it allows for a quick recalculation of these costs. On the other hand, in the second step, when it comes to actually solve the MWT instance, we rely on the results reported in [20] where it was observed that the B&B algorithm for the MWT performs much better with the relaxed Triangle Stabbing Model than with the relaxed Edge Stabbing Model. Now, given two iterations of the subgradient method, the triangle costs are the only differences between the associated MWT instances. These costs can be easily computed after the LMT-skeleton has been found in the first step. Some additional details are given below.

As said in Section 3, to solve the Lagrangian primal problem, we used the LMT-skeleton code written by Beirouti and Snoeyink and downloadable at [18]. A few modifications were introduced in this program to make possible the usage of arbitrary edge weights instead of Euclidean ones. This included, for instance, the removal of the *diamond test*, a simple and effective way to determine whether an edge could be part of a triangulation of minimum (Euclidean) length. Such changes do not have significantly damaged the algorithm's performance, relative to Euclidean weights, confirming it as a viable option for general MWTs.

After running the LMT-skeleton, quite often we still do not have a triangulation. Hence, a B&B algorithm is used to solve the constrained MWT that remains, *i.e.*, a MWT with sets of mandatory and forbidden edges. Since we use the (relaxed) Triangle Stabbing Model as the input for the B&B algorithm, these sets of edges have to be processed to identify the corresponding sets of triangles. Thus, if an empty triangle contains a forbidden edge, the associated variable is set to zero while, if all the edges forming its sides are mandatory, this variable is set to one.

Instances. The test suite used to analyze the performance of the MSTR algorithms was the same as in the MSST case. The reasons that support this choice are the same as before. Also, the time limit parameters inside the programs and in the timeout script remain unchanged, *i.e.*, 1800 and 2000, respectively. Once again, the symbol ‡ in the tables with results signalizes that the process was killed by the timeout script and, thus, did not produced any output.

Results. As in the MSST case, we divide our analysis into two parts, one for the TSP instances and the other for the grid instances. Concerning the TSP instances,

TABLE 6. Results for MSTR TSP instances.

Instance	LB		UB		Time		GAP%	
	LR	B&B	LR	B&B	LR	B&B	LR	B&B
berlin52	24	24	24	24	7.70	9.11	0.00	0.00
ch130	32	‡	33	33	165.09	‡	3.03	‡
ch150	34	‡	35	35	268.69	‡	2.86	‡
eil76	32	32	33	32	112.64	178.18	3.03	0.00
gil262	49	‡	50	50	1779.50	‡	2.00	‡
gr202	42	‡	42	42	615.63	‡	0.00	‡
kroA100	29	29	30	30	107.21	1967.38	3.33	3.33
kroA150	35	‡	36	36	330.66	‡	2.78	‡
kroA200	40	‡	41	41	736.80	‡	2.44	‡
kroB100	29	29	30	30	119.87	1976.12	3.33	3.33
kroB150	34	‡	35	35	408.44	‡	2.86	‡
kroB200	39	‡	40	40	705.75	‡	2.50	‡
kroC100	29	29	29	29	96.18	161.44	0.00	0.00
kroD100	29	29	29	29	30.45	86.90	0.00	0.00
kroE100	29	29	30	30	98.93	1962.76	3.33	3.33
lin318	69	‡	71	71	1803.40	‡	2.82	‡
pcb442	157	‡	180	180	1827.53	‡	12.78	‡
pr124	48	49	49	49	405.61	463.30	2.04	0.00
pr136	66	66	67	66	589.67	658.60	1.49	0.00
pr144	74	74	74	74	675.39	848.44	0.00	0.00
pr152	45	45	45	45	420.93	1015.55	0.00	0.00
pr226	141	150	150	150	1884.99	2855.06	6.00	0.00
pr264	90	‡	92	92	1811.44	‡	2.17	‡
rd100	29	29	29	29	17.45	82.05	0.00	0.00
rd400	52	‡	55	55	1803.73	‡	5.45	‡

the B&B algorithm had its process killed in 12 out of the 25 instances and, when this was not the case, it proved optimality in all but three instances, where there is a 3.33% gap (the gap exists because of the 1800 sec time limit). On the other hand, the Lagrangian SGM converged in all cases within the imposed time limit, with an average gap of 2.57%. The performance of the heuristic is remarkable. Optimality was proven for 7 instances, one of which could not be reached by the exact algorithm within the time limit (the inverse situation occurred four times). In 13 instances the difference between the upper and lower bounds was of just one unit. Using the WSEA we were able to improve the bounds provided by the LR algorithm in average by 0.97% while the time spent for this was 592.14 sec in average. These results are summarized in Table 6.

The results for the grid instances can be seen in Table 7. For those instances, the Lagrangian subgradient method was able to solve to optimality every instance. The B&B algorithm was unable to solve 4 out of 25 grid instances. In fact, only one of the 20×20 grid instances was solved within the time limit (the processes were killed by the timeout script) and every other grid instance was solved to optimality.

TABLE 7. Results for MSTR grid instances.

Instance	LB		UB		Time		GAP%	
	LR	B&B	LR	B&B	LR	B&B	LR	B&B
grid5a	22	22	22	22	0.17	0.17	0.00	0.00
grid5b	21	21	21	21	0.27	0.36	0.00	0.00
grid5c	21	21	21	21	0.17	0.17	0.00	0.00
grid5d	21	21	21	21	23.14	23.21	0.00	0.00
grid5e	20	20	20	20	0.18	0.18	0.00	0.00
grid8a	34	34	34	34	2.20	2.36	0.00	0.00
grid8b	34	34	34	34	3.48	3.71	0.00	0.00
grid8c	34	34	34	34	1.61	1.81	0.00	0.00
grid8d	35	35	35	35	1.07	1.26	0.00	0.00
grid8e	35	35	35	35	1.11	1.35	0.00	0.00
grid10a	44	44	44	44	8.01	9.03	0.00	0.00
grid10b	42	42	42	42	3.31	3.93	0.00	0.00
grid10c	47	47	47	47	9.52	10.48	0.00	0.00
grid10d	46	46	46	46	2.61	3.43	0.00	0.00
grid10e	46	46	46	46	7.05	8.10	0.00	0.00
grid15a	66	66	66	66	75.13	127.64	0.00	0.00
grid15b	68	68	68	68	13.65	70.36	0.00	0.00
grid15c	64	64	64	64	20.70	67.39	0.00	0.00
grid15d	66	66	66	66	39.24	86.21	0.00	0.00
grid15e	67	67	67	67	79.53	141.38	0.00	0.00
grid20a	89	89	89	89	500.78	2491.35	0.00	0.00
grid20b	86	‡	86	86	73.09	‡	0.00	‡
grid20c	90	‡	90	90	1781.70	‡	0.00	‡
grid20d	87	‡	87	87	204.77	‡	0.00	‡
grid20e	90	‡	90	90	1213.83	‡	0.00	‡

Regarding this set of instances, it is simply not worth executing a WSEA, since the LR is able to solve them relatively easy.

5. Conclusions and future directions

To our knowledge, this paper proposes the first exact approach to tackle the MSTR. Concerning the MSPM, our B&C algorithm is able to solve exactly all instance and runs in smaller computational times when compared to the results reported in [9]. As for the MSST, we developed an exact B&C algorithm based on a stronger formulation than the one introduced in [9, 10]. This algorithm obtained optimal solutions for several instances as well as high quality primal and dual bounds for many others in short computation times.

Moreover, we also devised Lagrangian heuristics for the three problems and conducted several computational experiments with them. These tests showed that they rapidly yield solutions with small costs, often proven optimal ones. It should

be noticed that, we are not aware of another work in the literature which reports on computational results for the MSTR.

Future directions in this research are currently being considered. This includes improving the performance of our heuristics by adding new features to it, such as, a procedure for variable fixing in the traditional Lagrangian fashion and a fast local search to reduce primal bounds.

Acknowledgements. This research was partially supported by *Conselho Nacional de Desenvolvimento Científico e Tecnológico* – Grants #301732/2007-8, #473867/2010-9, #147619/2010-6; *Fundação de Amparo à Pesquisa do Estado de São Paulo* – Grant #07/52015-0, and a grant from FAEPEX/UNICAMP. The authors are also grateful to Prof. M. Lübbecke for making available some of the instances used in the experiments.

References

[1] P. Agarwal, B. Aronov and S. Suri, Stabbing triangulations by lines in 3D, in *Proceedings of the eleventh annual symposium on Computational geometry*, SCG '95, *New York, NY, USA. ACM* (1995) 267–276.

[2] J. Beasley, Lagrangean relaxation, in *Modern Heuristic Techniques for Combinatorial Problems*. McGraw-Hill (1993) 243–303.

[3] R. Beirouti and J. Snoeyink, Implementations of the LMT heuristic for minimum weight triangulation, in *Proceedings of the Fourteenth Annual Symposium on Computational Geometry*, SCG '98, *New York, NY, USA* ACM (1998) 96–105.

[4] M. Berg and M. Kreveld, Rectilinear decompositions with low stabbing number. *Infor. Proc. Lett.* **52** (1994) 215–221.

[5] J.A. de Loera, S. Hosten, F. Santos and B. Sturmfels, The polytope of all triangulations of a point configuration. *Documenta Math.* **1** (1996) 103–119.

[6] E. Demaine, J. Mitchell and J. O'Rourke, The open problems project. Available online (acessed in January 2010). http://maven.smith.edu/~orourke/TOPP/.

[7] M.T. Dickerson and M.H. Montague, A (usually) connected subgraph of the minimum weight triangulation, in *Proceedings of the 12th Annual ACM Symposyum on Computational Geometry* (1996) 204–213.

[8] J. Edmonds, Maximum matching and a polyhedron with 0,1-vertices. *J. Res. Nat. Bur. Stand.* B **69** (1965) 125–130.

[9] S. Fekete, M. Lübbecke and H. Meijer, Minimizing the stabbing number of matchings, trees, and triangulations, in *SODA* edited by J. Munro. *SIAM* (2004) 437–446.

[10] S. Fekete, M. Lübbecke and H. Meijer, Minimizing the stabbing number of matchings, trees, and triangulations. *Discrete Comput. Geometry* **40** (2008) 595–621.

[11] M. Fischetti, J.J.S. Gonzalez and P. Toth, Solving the orienteering problem through branch-and-cut. *INFORMS J. Comput.* **10** 133–148, 1998.

[12] M. Grötschel and O. Holland, Solving matching problems with linear programming. *Math. Program.* **33** (1985) 243–259.

[13] T. Koch and A. Martin, Solving Steiner tree problems in graphs to optimality. *Networks* **33** (1998) 207–232.

[14] V. Kolmogorov, Blossom V: a new implementation of a minimum cost perfect matching algorithm. *Math. Program. Comput.* **1** (2009) 43–67.

[15] T.L. Magnanti and L.A. Wolsey, Optimal trees. *Handbooks in Operations Research and Management Science* **7** (1995) 503–615.

[16] J. Mitchell and J. O'Rourke, Computational geometry. *SIGACT News* **32** (2001) 63–72.

[17] J. Mitchell and E. Packer, Computing geometric structures of low stabbing number in the plane, in *Proc. 17th Annual Fall Workshop on Comput. Geometry and Visualization*. IBM Watson (2007).
[18] W. Mulzer, Index of /~mulzer/pubs/mwt_software/old/ipelets. Available online (accessed in March 2011). http://page.mi.fu-berlin.de/mulzer/pubs/mwt_software/old/ipelets/LMTSkeleton.tar.gz.
[19] W. Mulzer and G. Rote, Minimum-weight triangulation is NP-hard. *J. ACM* **55** (2008) 1–11.
[20] A.P. Nunes, *Uma abordagem de programação inteira para o problema da triangulação de custo mínimo*. Master's thesis, Institute of Computing, University of Campinas, Campinas, Brazil (1997). In Portuguese.
[21] M.W. Padberg and M.R. Rao, Odd minimum cut-sets and b-matchings. *Math. Oper. Res.* **7** (1982) 67–80.
[22] B. Piva and C.C. de Souza, The minimum stabbing triangulation problem: IP models and computational evaluation. ISCO (2012) 36–47.
[23] G. Reinelt, TSPLIB. Available online (acessed in March 2011). http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.
[24] J.R. Shewchuk, Stabbing Delaunay tetrahedralizations. *Discrete and Comput. Geometry* **32** (2002) 343.
[25] M. Solomon, *VRPTW benchmark problems*. Available online (acessed in August 2011). http://w.cba.neu.edu/~msolomon/problems.htm.
[26] C.D. Tóth, Orthogonal subdivisions with low stabbing numbers, Vol. 3608. *Lect. Notes in Comput. Sci.* Springer, Berlin/Heidelberg (2005) 256–268.
[27] L.A. Wolsey, *Integer Programming*. John Wiley & Sons (1998).