**ISTITUTO DI ANALISI DEI SISTEMI ED INFORMATICA**
**CONSIGLIO NAZIONALE DELLE RICERCHE**

**A. Frangioni, A. Lodi, G. Rinaldi**

# NEW APPROACHES FOR OPTIMIZING OVER THE SEMIMETRIC POLYTOPE

**R. 561 Maggio 2004**

**Antonio Frangioni** – Dipartimento di Informatica, Università di Pisa, largo Bruno Pontecorvo 1, 56127 Pisa, Italy (`frangio@di.unipi.it`)..

**Andrea Lodi** – Dipartimento di Elettronica, Informatica e Sistemistica, Università di Bologna, viale Risorgimento 2, 40136 Bologna, Italy (`alodi@deis.unibo.it`)..

**Giovanni Rinaldi** – Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti" del CNR, viale Manzoni 30, 00185 Roma, Italy (`rinaldi@iasi.cnr.it`)..

ISSN: 1128–3378

Collana dei Rapporti dell'Istituto di Analisi dei Sistemi ed Informatica, CNR

viale Manzoni 30, 00185 ROMA, Italy

tel. ++39-06-77161
fax ++39-06-7716461
email: `iasi@iasi.rm.cnr.it`
URL: `http://www.iasi.rm.cnr.it`

## Abstract

The semimetric polytope is an important polyhedral structure lying at the heart of hard combinatorial problems. Therefore, linear optimization over the semimetric polytope is crucial for a number of relevant applications. Building on some recent polyhedral and algorithmic results about a related polyhedron, the rooted semimetric polytope, we develop and test several approaches, mainly based over Lagrangian relaxation and application of Non Differentiable Optimization algorithms, for linear optimization over the semimetric polytope. We show that some of these approaches can obtain very accurate primal and dual solutions in a small fraction of the time required for the same task by state-of-the-art general purpose linear programming technology. In some cases, good estimates of the dual optimal solution (but not of the primal solution) can be obtained even quicker.

*Key words:* semimetric polytope, Lagrangian methods, max-cut, network design

## 1. Introduction

Let $G = (V, E)$ be a simple loopless undirected graph, $\mathcal{C}$ be the set of all chordless cycles of $G$, and $\bar{E}$ be the subset of the edges of $G$ that do not belong to a 3-edge cycle (a *triangle*) of $G$. For an edge function $x \in \mathbb{R}^E$ and an edge subset $F \subseteq E$, by $x(F)$ we denote the sum $\sum_{e \in F} x_e$. The *semimetric polytope* $\mathcal{M}(G)$ associated with $G$ is defined by the following system of inequalities:

$$x(C \setminus F) - x(F) \le |F| - 1 \qquad F \subseteq C \text{ with } |F| \text{ odd and } C \in \mathcal{C} \tag{1}$$

$$0 \le x_e \le 1 \qquad e \in \bar{E}. \tag{2}$$

The inequalities (1) are called the *cycle inequalities*. In the special case in which $G$ is complete, the only chordless cycles of $G$ are its triangles, therefore $\mathcal{M}(G)$ is defined by the set of inequalities

$$\left.\begin{array}{r} x_{ij} + x_{ik} + x_{jk} \le 2 \\ x_{ij} - x_{ik} - x_{jk} \le 0 \\ -x_{ij} + x_{ik} - x_{jk} \le 0 \\ -x_{ij} - x_{ik} + x_{jk} \le 0 \end{array}\right\} \quad \text{for all distinct } i, j, k \in V, \tag{3}$$

called the *triangle inequalities*. Observe that in this case the set $\bar{E}$ is empty, and so all the bounds on the variables are redundant.

The paper deals with the problem of finding efficient algorithms to optimize a linear function with coefficients $c \in \mathbb{R}^E$ over $\mathcal{M}(G)$.

We briefly mention two reasons to seek for an efficient way to optimize a linear function over $\mathcal{M}(G)$ that are related to the optimization of two difficult combinatorial optimization problems.

### The maximum cut of a graph

For a node set $W \subseteq V$, the set of all the edges in $E$ having exactly one endpoint in $W$ is denoted by $\delta(W)$ and is called a *cut* of the graph $G$. The node sets $W$ and $V \setminus W$ are called the *shores* of the cut. The *maximum cut problem* (*max-cut* for short) is to find a cut $\delta(W^*)$ of $G$ having maximum weight $c(\delta(W^*))$. With every cut $\delta(W)$ of $G$ we associate an *incidence vector* $\chi^W \in \mathbb{R}^E$ where, for each $e \in E$, the component $\chi^W_e$ is equal to 1 if $e \in \delta(W)$ and is equal to 0 otherwise. Thus the weight of the cut $\delta(W)$ is also given by the inner product $c \cdot \chi^W$. The convex hull of the incidence vectors of all cuts of $G$ is the *cut polytope* $CUT(G)$ associated with $G$ (see, e.g., [6] for the details). Then max-cut can be formulated as the linear program

$$\max\{c \cdot x \mid x \in CUT(G)\}.$$

It is not difficult to see that the semimetric and the cut polytopes are related by the inclusion

$$CUT(G) \subseteq \mathcal{M}(G),$$

which is strict for $|V| > 4$.

Thus, maximizing $c$ over $\mathcal{M}(G)$ produces an upper bound on the maximum $c$-value cut of $G$ that can be exploited in branch and bound or in a branch and cut scheme for solving max-cut to optimality. Actually, in all the computational studies concerning instances of max-cut for very large sparse graphs based on a branch and cut scheme, the only relaxation exploited is, to a large extent, $\mathcal{M}(G)$. Consequently, the computation of a maximum cut merely amounts to a (possibly long) series of linear optimizations over $\mathcal{M}(G)$.

### Network design

If an edge capacity function $q \in \mathbb{R}^E$ and an edge demand $d \in \mathbb{R}^E$ functions are given, the existence of a feasible *multiflow* in the network defined by $G$, $q$, and $d$ is established by the Japanese Theorem (see, e.g., [23]). According to this theorem a feasible multiflow exists if and only if $\mu \cdot (q - d) \ge 0$ holds for every *metric* $\mu$ on $V$, i.e., for every point of the cone defined by all the homogeneous equations of (3). It is not hard to see that this is equivalent to the condition $\min\{(q-d)\cdot x \mid x \in \mathcal{M}(G)\} \geq 0$. In network design such a feasibility problem has to be solved several times. This again calls for an effective solution algorithm.

Although the problem of optimizing the linear function $c \cdot x$ over the polytope defined, e.g., by (3) in the case of a complete graph, is just a standard linear program with a polynomial number $(4\binom{|V|}{3})$ of constraints, it happens to be surprisingly difficult to solve with standard LP tools such as the simplex method or interior-point methods, as it will be shown in the following, even if state-of-the-art software is used.

In light of the previous observations, it is therefore of considerable interest to develop alternative algorithmic techniques that are able to compute (possibly with some degree of approximation) optimal primal and dual solutions of these LP's, faster than it is currently doable with standard methods.

A natural alternative technique is the Lagrangian approach where we "dualize" all the triangle (or the cycle) inequalities and leave, as explicit constraints, only the upper and lower bounds on the variables (although redundant). Such an approach has been successfully applied, e.g., in [3] where the "Volume Algorithm" is used to solve the Lagrangian dual problem.

We propose a slight modification of this approach. First, we dualize only a subset of the cycle inequalities, leaving, as explicit constraints, the inequalities that define the so called *rooted semimetric polytope.* In [12, 13] it is shown how a linear program defined on this polytope can be solved efficiently. Then, to solve the Lagrangian dual problem, besides the Volume Algorithm, we make use also of a bundle-type algorithm.

In this paper we report on our experience along this route. We tested several different variants of Lagrangian relaxation and two of the main current algorithmic approaches available for solving the Lagrangian dual: a subgradient-type algorithm and a bundle-type algorithm; at termination, we also used a projection-type heuristic for quickly obtaining feasible primal solutions out of the (slightly) infeasible primal solutions that are typically generated by the Lagrangian approaches. We discovered that the proper combination of these tools results in algorithms that can obtain very accurate primal and dual solutions in a small fraction of the time required for the same task by state-of-the-art general purpose LP technology. In some cases, good estimates of the dual optimal solution (but not of the primal solution) can be obtained even quicker. These results show that the proper combination of Lagrangian techniques and efficient algorithms for optimization over the rooted semimetric polytope hold some promise as building blocks for approaches to the related hard combinatorial optimization problems mentioned above.

The structure of the paper is as follows. In Section 2 we recall the relevant polyhedral and algorithmic properties of the semimetric and rooted semimetric polytopes. In Section 3 we propose a number of different ways for exploiting the efficient algorithm for the rooted semimetric polytope in order to solve the linear optimization problem over the semimetric polytope. Then, in Section 4 the relevant details of the implementation of the proposed approaches are discussed; in Section 5 the obtained computational results are presented and discussed, and, finally, in Section 6 some conclusions are drawn.

## 2. Semimetric polytopes

As it was mentioned in the Section 1, the inequalities defining the semimetric polytope for a complete graph are $4n(n-1)(n-2)/6$. Therefore, already for graphs of a few hundred nodes, the inequalities are too many to be handled explicitly and the use of a cutting-plane approach is mandatory. In this case, the separation procedure, which provides triangle inequalities violated by a given point, runs in polynomial time as it trivially amounts to making a check in a set of inequalities whose size is polynomial in $n$.

In general, for a graph $G$ the cycle inequalities (1) are exponentially many. Nevertheless, as it was proved in [4], the separation for these inequalities runs in polynomial time, as it amounts for finding at most $n$ shortest paths in a graph that has twice the size of $G$.

By the polynomial equivalence between separation and optimization (see, e.g., [16]), it follows that optimizing a linear function over the semimetric polynomial can be done in polynomial time. As it was noticed before, this task may take a substantial amount of time if a standard LP optimizer is used in combination with a cutting-plane algorithm. Therefore it would be extremely helpful if a purely combinatorial algorithm were available to optimize over the semimetric polytope in polynomial time. Unfortunately, at present finding such an algorithm appears to be quite a difficult task.

To the contrary, a purely combinatorial algorithm has been found for a relaxation of the semimetric polytope that we describe next.

Let $r$ be a selected node of $G$ that will be called the *root node*. Without loss of generality, we assume that $r$ is adjacent to every other node of $G$ (if this is not the case we can add new edges to the graph with zero weight). Let $E^r$ be the edgeset of the subgraph of $G$ induced by $V^r = V \setminus \{r\}$. The subset of triangle inequalities (3) corresponding to all triples $(r,i,j)$ for all $(i,j) \in E^r$ defines the *rooted semimetric polytope* $\mathcal{M}^r(G)$.

Despite the fact that $\mathcal{M}^r(G)$ has much less defining inequalities than $\mathcal{M}(G)$, it is still true that the incidence vector of any cut of $G$ belongs to $\mathcal{M}^r(G)$ and, conversely, that every integral point in $\mathcal{M}^r(G)$ is the incidence vector of a cut of $G$. In other words, $\mathcal{M}^r(G)$ is a relaxation of $\mathcal{M}(G)$ that provides an integer linear programming formulation of max-cut. This formulation is minimal, i.e., the removal of any its inequalities results in a formulation that has integral feasible solutions that are not incidence vectors of cuts of $G$.

Therefore, optimizing $c \cdot x$ over $\mathcal{M}^r(G)$ yields an upper bound on the value of an optimal cut, as optimizing over $\mathcal{M}(G)$ does. However, in most practical cases the former bound is far weaker than the latter, and thus is of little use for any procedure aiming, e.g., at solving max-cut. On the other hand, as shown in [12], optimizing over $\mathcal{M}^r(G)$ can be accomplished through very efficient network flow algorithms, much faster than using standard linear programming technology.

More in details, let us denote by $A^r x \leq b^r$ the set of constraints that define $\mathcal{M}^r(G)$ and let us consider the linear program

$$\max\{c \cdot x \mid A^r x \leq b^r\}. \tag{4}$$

Consider a capacitated directed graph with node set $\{r\} \cup U \cup U'$ defined as follows. For each node $i \in V^r$ we associate the two nodes $i$ and $i'$ belonging to $U$ and $U'$, respectively. Two opposite uncapacitated arcs connect $r$ with every node in $U \cup U'$. With each edge $ij$ $(i < j)$ of $E^r$ we associate two arcs $(i,j')$ and $(j,i')$ if $ij \in E^r_+ = \{ij \in E^r \mid c_{ij} > 0\}$, and the two arcs $(i,j)$ and $(i',j')$ if $ij \in E^r_- = \{ij \in E^r \mid c_{ij} \leq 0\}$. The capacity of these arcs is $2|c_{ij}|$. Finally, consider the following minimum cost flow problem (MCFP) associated with this directed graph:

$$\begin{aligned}
&\min \sum_{i \in V^r} (u_{ri} + u_{ri'}) + \sum_{ij \in E^r_+} (v_{ij'} + v_{ji'}) \\
&\text{subject to} \\
&\left.\begin{aligned}
u_{ri} + \sum_{ij \in E^r_+} v_{ij'} + \sum_{\substack{ki \in E^r_- \\ k < i}} w_{ki} - \sum_{\substack{ki \in E^r_- \\ k > i}} w_{ik} - q_{ir} &= d_i \\
- u_{ri'} - \sum_{ij \in E^r_+} v_{ji'} - \sum_{\substack{ki \in E^r_- \\ k < i}} w_{k'i'} + \sum_{\substack{ki \in E^r_- \\ k > i}} w_{i'k'} + q_{i'r} &= -d_i
\end{aligned}\right\} \quad i \in V^r \\
&\left.\begin{aligned}
v_{ij'} &\leq 2|c_{ij}| \\
v_{ji'} &\leq 2|c_{ij}|
\end{aligned}\right\} \quad ij \in E^r_+ \\
&\left.\begin{aligned}
w_{ij} &\leq 2|c_{ij}| \\
w_{i'j'} &\leq 2|c_{ij}|
\end{aligned}\right\} \quad ij \in E^r_- \\
&u, v, w, q \geq 0 \,,
\end{aligned} \tag{5}$$

where

$$d_i = c_{ri} + \sum_{ij \in E^r} c_{ij} - 2 \sum_{\substack{hi \in E^r_- \\ h < i}} c_{hi}.$$

Then we have the following

**Theorem 2.1 ([12])** *The optimal objective function value of (5) equals twice the optimal value of (4). An optimal solution of (4) is obtained from an optimal solution of (5) with a simple linear time algorithm.*

Since strongly polynomial algorithms for MCFP exist (see, e.g., [1]), it follows that (4) is also solvable in strongly polynomial time. Furthermore, since the largest absolute value of the components of the cost vector, $C$, is equal to one in our case, the Cost Scaling algorithm, that runs in $O(n^2 m \log nC)$ time, also solves (4) in strongly polynomial time. Indeed, in [13] numerical results using a Cost Scaling implementation [15] are reported which suggest that this class of approaches is indeed very effective for the type of instances produced by (5).

Thus, (4) is a "very easy" problem. However, we are rather interested in solving

$$\max\{c \cdot x \mid A^r x \leq b^r,\ r = 1, \ldots, n-1\} \tag{6}$$

that is, in the linear optimization over the whole $\mathcal{M}(G)$. Note that any two blocks of constraints $A^r x \leq b^r$ and $A^q x \leq b^q$ are in general not disjoint; in fact, it is easy to verify that only $n-1$ blocks (without loss of generality, the first $n-1$) suffice to "cover" all the constraints in $\mathcal{M}(G)$. In order not to overburden the notation, we will assume that, each time that multiple copies of a constraint (and the corresponding dual multipliers) are present, only one copy is actually considered; accordingly, we will write

$$\min\left\{\sum_{r=1}^{n-1} y^r b^r \mid \sum_{r=1}^{n-1} y^r A^r \geq c\right\} \tag{7}$$

for the dual of (6), although in principle any two subvectors $y^r$ and $y^q$ of the dual vector of variables $y$, corresponding to blocks $A^r x \leq b^r$ and $A^q x \leq b^q$, share some variables.

Our goal is to solve the primal problem (6) which has $n-1$ blocks of constraints, exploiting the fact that if the problem had only one of these blocks it would be "very easy" to solve. Equivalently, we want to solve the dual problem (7), which has $n-1$ blocks of variables, exploiting the fact that if we knew the optimal value of the dual variables for all but one of these blocks, we could "very easily" compute the optimal value for the remaining variables.

The presence of an "easy" structure embedded into a "more complex" problem is one rather common occurrence in optimization, and it is often dealt with by means of *Lagrangian* approaches. However, the present case is (qualitatively) different by most of the standard ones in that the blocks are *many.* For example, for a graph with 100 nodes, we have in fact 99 blocks of constraints. Since all blocks are of roughly equal size, each block contains approximately 1/100 of the constraints, i.e., only a *few* of them. Put it in dual terms, we have an efficient way for optimizing over a subset of the dual variables, but the subset is rather "small".

Yet, there are quite a number of different ways for exploiting this structure for algorithmic purposes; the next section will be devoted to discussing some of them.

## 3. Solving the semimetric problem

The main tool used to exploit the algorithmic results on (4) in order to solve (6) is Lagrangian relaxation; the reader is therefore assumed to be familiar with the basic principles of Lagrangian duality (see, e.g., [10, 21]).

### 3.1. Keep one, relax many

The easiest route for solving (6) is perhaps to arbitrarily select one $r \in V$, form the Lagrangian relaxation of (6) with respect to all other blocks of constraints

$$z(y) = \max_x \left\{c \cdot x - \sum_{h \neq r} y^h (b^h - A^h x) \mid A^r x \leq b^r\right\} \tag{8}$$

where the $y^h$, $h \neq r$ are fixed Lagrangian multipliers, and then solve the corresponding Lagrangian dual

$$\min_y \{z(y) \mid y \geq 0\} \tag{9}$$

This is easily seen that solving (9) is equivalent to solving problem (7); indeed, the Lagrangian multipliers in (9) are precisely the dual variables of (7), except for the $O(n^2)$ dual variables $y^r$ that are *implicitly* handled by the Lagrangian relaxation.

The program (9) is a large-scale Non Differentiable Optimization problem, with $O(n^3)$ constrained (in sign) variables. As such, it can be expected to be, in practice, a relatively "difficult" task; NDO problems with hundreds of thousands of variables are not generally considered as routinely solvable by standard approaches. However, as shown by the results in Section 5, the problems corresponding to this application turn out to be solvable quite efficiently. For this to be true, the choice of a proper NDO algorithm and its correct implementation, as discussed in Section 4, are crucial.

Yet, while planning the implementation of this approach, we had to confront with the following two issues.

The first is about the choice of $r$. How should $r$ be chosen? In other words, can we tell a priori which constraints (dual variables) are most "relevant" to determine the optimal solution? Furthermore, is a "good" choice of the root node at the initial iterations of a NDO approach still "good" in a later phase of the algorithm? In other words, is there any "good" fixed choice of the root, or should we resort to some "root hopping" approach? In the latter case, care should be paid with the algorithmic aspects. In principle, a "root hopping" approach is simple when we have to solve (7): we seek for a full optimal dual vector $[y^1, \ldots, y^{n-1}]$ for (7), dynamically changing the root only influences which of the blocks of dual variables is "controlled" by the solution of the Lagrangian problem (8) rather than by the NDO algorithm. However, for each choice $r$ of the root, a different Lagrangian function $z(y)$ is defined; therefore, it is not at all straightforward to adapt existing NDO algorithms to the optimization of such a family of related functions.

The second issue stems from the fact, already discussed in Section 2, that each of the constraints blocks $A^r x \leq b^r$ involves only a "few" ($O(n^2)$) of the "many" ($O(n^3)$) constraints of (6). In dual terms, the optimization over the single block in (8) can only set a "few" of the "many" variables of (7). Therefore, it is not obvious a priori that the Lagrangian approach using the rooted semimetric polytope is computationally superior to solving the equivalent Lagrangian dual

$$\min_{y \geq 0} \left\{ \max_x \left\{ c \cdot x - \sum_{r=1}^{n-1} y^h (b^h - A^h x) \mid x \in \{0,1\}^E \right\} \right\} \tag{10}$$

of (6) with respect to *all* blocks of constraints. In fact, the advantage of having "slightly fewer" dual variables $y$ is paid by the need of solving one flow problem at each iteration, that is clearly more costly than the trivial optimization over the unitary hypercube required by (10).

### 3.2. Lagrangian decomposition

A different possibility for exploiting multiple "easy" structures in a problem is to use *Lagrangian decomposition.* In our case, this amounts to rewriting (6) in the following equivalent form

$$\begin{aligned} \max \quad & \bar{c} \sum_{r=1}^{n-1} x^r \\ \text{subject to} \quad & \\ & A^r x^r \leq b^r \quad r = 1, \ldots, n-1 \\ & x^r = x^{r+1} \quad r = 1, \ldots, n-2 \end{aligned} \tag{11}$$

where $\bar{c} = \frac{c}{n-1}$. The program (11) has $n-1$ blocks of *duplicated variables*, so that each block of constraints has its own independent block of variables. Thus, the Lagrangian dual of (11) with respect to all the

$x^r = x^{r+1}$ constraints

$$\begin{aligned}\min_{\pi} \Big\{ &\max_{x^1} \big\{ (\bar{c} - \pi^1)x^1 \mid A^1 x^1 \leq b^1 \big\} \\ &+ \sum_{r=2}^{n-3} \max_{x^r} \big\{ (\bar{c} - \pi^r + \pi^{r-1})x^r \mid A^r x^r \leq b^r \big\} \\ &+ \max_{x^{n-1}} \big\{ (\bar{c} + \pi^{n-2})x^{n-1} \mid A^{n-1}x^{n-1} \leq b^{n-1} \big\} \Big\}\end{aligned} \tag{12}$$

can be solved by means of $n-1$ optimizations over $\mathcal{M}^r(G)$ (with $n-1$ distinct roots). Elementary theory of Lagrangian duality shows that this yields the same bound of (7), and therefore of the previous approaches.

An advantage of the Lagrangian decomposition approach is that it is independent on the choice of a particular root node; a disadvantage, however, is that at each iteration the solution of $n-2$ problems over $\mathcal{M}^r(G)$ is required, which may be significantly more expensive than the iteration cost of (9) and, *a fortiori*, that of (10).

Moreover, for the solution (12) we need to solve a large-scale Non Differentiable Optimization problem, with $O(n^3)$ unconstrained variables $\pi$, and therefore similar considerations as in the previous case apply: such a problem cannot be expected to be, in practice, "easy" to solve. The results in Section 5 actually confirm that this is the case: the rate of convergence (number of iterations required to reach a specified precision) is, on the test instances, far slower than that of (9), making this approach impractical for obtaining high quality solutions.

### 3.3. A hybrid approach

It is possible to combine the two previous approaches in one unified method of which the two are special cases. The idea is to select a *set* $R = \{r_1, r_2, \ldots, r_k\} \subset V$ of root nodes, and to consider the following equivalent form of (6)

$$\begin{aligned} \max \quad & \bar{c} \sum_{r \in R} x^r \\ \text{subject to} \quad & \\ & A^r x^r \leq b^r & r \in R \\ & x^{r_i} = x^{r_{i+1}} & i = 1, \ldots, k-1 \\ & A^q \left( \frac{1}{k} \sum_{r \in R} x^r \right) \leq b^q & q \notin R \end{aligned} \tag{13}$$

where $k = |R|$ and $\bar{c} = \frac{c}{k}$. The problem has $k$ copies of the original variables $x$ linked by $k-1$ blocks of equality constraints, plus all the blocks of constraints (3) not "covered" by the blocks in $R$. In these constraints it is convenient to express the identical value of all variables blocks $x^r$ in terms of the average of the duplicated variables.

Thus, the Lagrangian dual of (11) with respect to all the $x^{r_i} = x^{r_{i+1}}$ constraints *and* the blocks out of $R$

$$\begin{aligned}\min_{\pi, y} \Big\{ &\max_{x^{r_1}} \big\{ (\bar{c} - \pi^1)x^{r_1} - \frac{1}{k} \sum_{q \notin R} y^q (b^q - A^q x^{r_1}) \mid A^{r_1} x^{r_1} \leq b^{r_1} \big\} \\ &+ \sum_{i=2}^{k-1} \max_{x^{r_i}} \big\{ (\bar{c} - \pi^r + \pi^{r-1})x^{r_i} - \frac{1}{k} \sum_{q \notin R} y^q (b^q - A^q x^{r_i}) \mid A^{r_i} x^{r_i} \leq b^{r_i} \big\} \\ &+ \max_{x^{r_k}} \big\{ (\bar{c} + \pi^k)x^{r_k} - \frac{1}{k} \sum_{q \notin R} y^q (b^q - A^q x^{r_k}) \mid A^{r_k} x^{r_k} \leq b^{r_k} \big\} \Big\}\end{aligned} \tag{14}$$

can be solved by means by $k$ optimizations over $\mathcal{M}^r(G)$, with $k$ distinct roots. The program (14) is still a large-scale Non Differentiable Optimization problem, with $O(kn^2)$ unconstrained variables $\pi^i$, corresponding to the $x^{r_i} = x^{r_{i+1}}$ constraints, plus $O((n-k)n^2)$ constrained variables $y^q$, corresponding to all the blocks $q \notin R$. Thus, in terms of number of variables (14) is essentially equivalent to the three previous approaches, and the same considerations apply. Indeed, we can consider all the approaches proposed so far as special cases of (14): problem (9) corresponds to $|R| = 1$, (12) corresponds to $|R| = n-1$ and even (10) can be thought as corresponding to (14) with $R = \emptyset$.

Thus, the hybrid approach offers an "handle", namely $|R|$, that allows one to compromise between the number of blocks which have their own vector of variables and those that do not. The immediate consequence is that a possibly critical decision has to be made: what is a good value for $|R|$? Then, similar questions as for (9) arise: how to select the elements of $R$? Should $R$ be a fixed set, or "roots hopping" should be used? We remark once again that modifying NDO approaches in order to construct provably convergent algorithms to solve (14), if the set $R$ is allowed to change, is again nontrivial.

## 4. Implementation details

All the approaches of the previous section require to solve some Lagrangian dual, i.e., large-scale Non Differentiable Optimization problems. Several algorithms have been proposed that can be used to perform this task; the choice of the proper algorithm, and possibly even some details of the implementation, is therefore potentially crucial for making the proposed approaches computationally effective.

### 4.1. Non Differentiable Optimization algorithms

The form of the dual problem (7) and the availability of efficient algorithms for solving its restrictions, in particular the Lagrangian relaxation (8), immediately suggest to apply a *block dual-ascent* approach. The approach can be shortly described as follows: at each iteration a current dual point $[y^1, \ldots, y^{n-1}]$ is kept. One root node $r$ is selected, in some cyclical fashion, all the $y^h, h \neq r$ are fixed, and (8) is solved. This gives a new value $\bar{y}^r$ for the block of dual variables corresponding to the selected root, that substitutes $y^r$ in the current point; then, a new iteration begins. The algorithm stops when all the $n-1$ roots are tested without producing any change in the dual multipliers (alternatively, no improvement in the value of the objective function). As previously noted, the blocks of variables are not disjoint, but this does not affect the applicability of the approach.

This is a heuristic approach for (7), in that there is no guarantee that such an algorithm will eventually converge to an optimal solution. Furthermore, even if a dual optimal solution is attained, no optimal primal solution—nor even a close approximation—is in general available. Finally, the approach does not extend to different models such as (14).

Therefore, solving all the models with any arbitrary degree of precision requires using some standard Non Differentiable Optimization algorithm. It is clearly out of the scope of this paper to discuss the current status of computational NDO in details; we will simply provide a synthetic description of the characteristics of the algorithms that are relevant in our case.

Implementable NDO algorithms can, for the sake of taxonomy, be grouped in two categories:

1. cutting-plane-type approaches;

2. subgradient-type approaches.

The first class of algorithms stems from the classical cutting-plane algorithm, perhaps better known for its specialized version for structured linear programs: Dantzig-Wolfe's decomposition method. A number of other approaches have, however, been proposed in an attempt to improve on the relatively poor performances in practice of the cutting-plane method. These algorithms can be further subdivided into two different groups: bundle algorithms (e.g., [9, 18, 21]) and algorithms based on centers (e.g., [7, 21]).

All algorithms of this category need to solve a "complex" *master problem* at each iteration, in order to compute the next vector of Lagrangian multipliers. The corresponding Lagrangian relaxation is then solved, producing a set of (approximately) optimal solutions that are used to compute (approximate) *subgradients* of the Lagrangian function. These subgradients are in turn used to modify the master problem, usually increasing its size (number of constraints in the primal, variables in the dual). Rules can be defined for some, but not all, algorithms that allow one to keep the size of the master problem, and therefore its computational cost, bounded.

The actual form of the master problem is the most prominent aspect that differentiates cutting-plane-type approaches among themselves. The master problem can be: a) a linear program, as in the cutting-plane algorithm and in some form of bundle algorithms, b) a convex quadratic program, as in most bundle algorithms, or c) a general nonlinear program (e.g., with logarithmic objective function), as in other bundle algorithms and in the algorithms based on "centers". Therefore, even for an identical set of subgradients, the cost of solving the master problem can be markedly different according to the NDO algorithm employed and also to the algorithm used to solve the master problem itself. For our implementation we have chosen a "classical" *proximal bundle* method, similar to those successfully used, e.g., in [5, 11]. The master problem for this algorithm is a structured convex quadratic program, which can be efficiently solved e.g. with the specialized code of [8].

An important characteristic of cutting-plane-type approaches is that, at each iteration, the solution of the master problem produces a set of convex multipliers that can be used to construct a solution to problem (6) out of the solutions of the Lagrangian relaxations obtained at the previous iterations. This solution may or may not be feasible with respect to the relaxed constraints, depending on the particular approach; for the proximal bundle method, for instance, it is in general not so. However, for all methods this solution is guaranteed to converge to a primal optimal solution of (6); in practice, the solution tends to quickly become "almost feasible". This is crucial for our application because both a primal and a dual optimal solution of (6) are in principle required. At termination, the (possibly infeasible) primal solution constructed with the information provided by the master problem is used to obtain an (almost) optimal solution with a procedure described later. Furthermore, the (normally infeasible) primal solution is also computed at each iteration where we check if new Lagrangian variables have to be created, to serve as the primal point where the separator is called, as discussed in more details below.

Subgradient methods are simpler as they do not, on the surface, require the solution of a master problem; in fact, they choose the next point where the Lagrangian function has to be evaluated along a direction that is a linear combination of the latest obtained subgradient and the direction used at the previous iteration. The rules for computing the parameters of the linear combination and the step size differentiate subgradient-type approaches among themselves.

However, the distinction between cutting-plane and subgradient approaches is more blurred than it appears from the above description. In fact, some subgradient methods [20, 3] also produce information that can be used to construct a primal solution, out of the previous solutions of the Lagrangian relaxations, which converges to an optimal solution of (6). On the other hand, the algorithmic parameters of some bundle methods (such as the one we used) can be chosen in such a way that the computational effort required for the master problem is potentially comparable to that required for computing the direction in a subgradient approach. In other words, subgradient approaches which produce primal solutions are very similar to bundle approaches where the size of the master problem is always limited to the smallest possible value [2]. Thus, bundle approaches can be considered to be more general, as they provide a "knob"—the maximum size of the master problem—that can be set according to the characteristics of the problem to balance between the cost of the master problem solution and the overall convergence speed (see, e.g., [5]). Yet, a number of possibly important algorithmic details, such as the step-size selection rule and the stopping condition, differentiate practical implementations of bundle approaches from that of subgradient approaches, as further discussed in Section 5. Thus, we decided to also test the algorithm of [3] for the solution of the proposed Lagrangian duals.

An important characteristic of the NDO problems to be solved in our application is their extremely large size; for a complete graph with 150 nodes, for instance, there are more than two million inequalities (3). However, the optimal Lagrangian multipliers corresponding to most of these inequalities are expected to be zero. Therefore, the solution of (14) for "small" values of $|R|$ may greatly benefit from a dynamic generation of Lagrangian variables (already used with success, e.g., in [14, 11]), that is, an ordinary row generation approach for the solution of (6).

Indeed, our preliminary results have shown that such a strategy has a considerable impact on the overall efficiency of the algorithms. This is of course far more relevant for the bundle approach, where the solution of the master problem is costly, but also the subgradient approach greatly benefits from it. It has to be remarked that a large fraction of the improvement in the running time obtained through the use of the dynamic Lagrangian variables generation in the bundle approach is due to the use of the specialized code of [8]. Indeed, the two-level active-set approach employed in the algorithm of [8] allows one to deal very efficiently with the changes in the master problem corresponding to Lagrangian variables creation/destruction. Moreover, the initial tests actually suggested a number of improvements in the code that allowed to better exploit the creation/deletion of large "chunks" of variables. Implementations of bundle approaches using non-specialized codes might have benefited less from the dynamic generation of Lagrangian variables.

Since generating new Lagrangian variables corresponds to separating violated inequalities (3), a primal solution is needed. Traditionally, the solution of the latest Lagrangian problem has been used [14]; however, this solution is not always a "good" input for separation routines [17]. Fortunately, recent NDO algorithms provide, as noted before, a very convenient alternative under the form of the primal solution obtained by convex combination of the previous ones, and we have found this point to provide a completely satisfactory input for the separation routines.

## 4.2. Finding primal feasible solutions

As previously mentioned, the NDO algorithms employed for solving the Lagrangian duals provide (at least asymptotically) optimal solutions to the primal problem as well. However, when the algorithms are finitely stopped the available primal solutions are most often infeasible. This is especially true for the subgradient approach which, due to its stopping rules (discussed in more details in the next Section), most often terminates with a primal solution that is very far from being feasible. The situation for the bundle approach is different in that, normally, the obtained primal solution is very near to being feasible; however, it is most often not feasible within the typical precision provided by a standard linear programming solver, i.e., at least `1e-8` (relative) for interior-point solvers and `1e-12` for simplex solvers. It is often possible to obtain feasible primal solutions with the bundle approach by properly setting the algorithmic parameters which control the stopping conditions of the code. However, normally the bundle algorithm reaches a very accurate dual solution much earlier than producing such a primal solution. Therefore, setting the parameters in such a way, as to require a very high precision in feasibility, may result in a considerable increase in the number of iterations.

For all these reasons, we decided to experiment with a very simple projection-type approach for producing feasible primal solution out of the (slightly) infeasible ones "naturally" generated by the NDO algorithms. For each inequality in (3) the feasibility (to within `1e-12`) of the current primal solution is tested: if a violated inequality is found the primal solution is projected over the inequality, using simple closed formulae, and the process is repeated until no more violated inequalities are found. Note that this process does not depend on which particular NDO approach has been used, nor from which Lagrangian dual has been solved.

Because each inequality has only three nonzero coefficients (when $G$ is complete), both checking the violation of an inequality and projecting one point over a violated inequality requires $O(1)$. Thus, an entire scan of all the constraints requires linear time in the total number of inequalities. We found this approach to be almost surprisingly efficient, in that it has always, in all our experiments, produced a feasible solution in very few scans. Indeed, the running time required for finding the primal feasible solution has invariably been a very small fraction of the time required by the overall algorithm. Also, the objective function value of the obtained feasible primal solution is typically very close to that of the starting infeasible solution, even though the objective function is not taken into account at all in the process. As the next section shows, this simple approach allowed us to produce very high quality primal solutions, together with the dual solutions provided by the NDO algorithms.

## 5. Computational results

We empirically evaluated the proposed approaches with a large-scale comparison. We used 5 different types of graph: a) *clique* graphs, b) *planar* graphs randomly generated with density between 50% to 100% of the maximum possible density of a planar graph, c) *simplex* graphs (see, e.g., [19]). The edge costs were drawn from uniform random distribution with several ranges (namely [-10,10], [-50,50], [-100,100], [-10,80], and [0,100]). The last two types were d) *toroidal 2D* and e) *toroidal 3D-grid* graphs, i.e., 2- and 3-dimensional grid graphs where the first and the last node of each "line" of the grid are adjacent. These instances are relevant, e.g., in Statistical Physics for analyzing the properties of spin glass (see, e.g., [22]). As it is customary in the spin glass literature, for the these two classes of graphs we experimented both with $\pm 1$ costs, drawn from a uniform random binary distribution and with costs drawn from a Gaussian distribution. Thus, we had a total of seven groups of instances. Within each group we produced instances with different number of nodes (comprised between 25 and 150) according to the characteristics of the class. For each group and size we generated either 5 or 15 different instances, for a grand total of 175 instances.

In the instances, sparse graphs are "completed" by adding zero-cost edges, so that the separation procedure was done by trivial enumeration of all triangle inequalities and the results were not affected by the degree of sophistication of the algorithm used to separate the cycle inequalities. Note that turning a sparse graph into a complete one by adding edges of zero cost, produces a new instance in which the shores of a maximum cut define a maximum cut of the original (sparse) instance. Although this transformation makes it possible to deal with polynomially sized semimetric relaxations, it typically produces instances that are much more difficult to solve than the original ones (they have much more variables and most of them have nonzero value at the optimal solution of the semimetric relaxation). Therefore, our choice of completing all graphs of our test-bed has also the effect of testing the algorithms with supposedly difficult instances.

All the instances have been produced by the *rudy* random generator (available at [24]), whose main feature is to produce machine-independent instances; the parameters for creating the instances are available upon request from the authors.

Apart from comparing the approaches among themselves, we also tested their efficiency against a sotware based on the state-of-the-art general-purpose linear programming code CPLEX 9.0 for solving (6). Although writing a code for solving (6) with CPLEX may appear to be a trivial exercise, we found ourselves confronted with some computational choices. The first was whether to provide the solver with a full formulation of (6), or to use a row generation approach completely analogous to that used by the Lagrangian approaches. The second was which of the three main LP algorithms (primal simplex, dual simplex and primal-dual interior-point) should be used. The third is, if row generation is performed, the maximum number $h$ of violated inequalities insert at most at each row generation.

We tested all possible combinations of the above choices, and we discovered that solving the full formulation of the problem with the primal-dual algorithm is always consistently faster than all other alternatives. However, the maximum size of the solvable instances for this approach, on our machines with 1Gb RAM, is roughly 150 nodes, while the row generation methods can solve much larger instances. So we decided to report the results of both the "monolihitc" approach and of the best of the row generation ones, that is, the one using – again – the interior-point algorithm and $h = n(n-1)$ (corresponding to twice the size of a base). It may be worth remarking that in row generation the primal simplex was always consistently slower than the other two approaches, while the dual simplex was often, but not always, almost competitive with the interior-point algorithm. Also, the version of the code that was tested did not include any procedure for removing "useless" inequalities from the current model: all the removal strategies that we tested considerably increased the number of row generation iterations and the running time. This may be partly justified by the fact that, with "reasonable" values of $h$, the LP-based code never performs more than a few row generations, and therefore it is difficult to decide which inequalities are actually useless.

The Lagrangian approaches have been implemented using a general-purpose bundle solver developed by the first author and a version of the Volume algorithm directly derived by the publicly available version of [3]. These codes have numerous algorithmic parameters that can be tuned to maximize their performances. In particular, note that at least one of these algorithmic parameters—the maximum number $h$ of violated inequalities to be inserted at each Lagrangian variables generation—is common with the LP-based approach. In order to obtain a fair comparison with the LP-based approach, where all instances are run with the same default parameters for the LP solver, we refrained from instance- or even class-specific tuning of the algorithmic parameters of both the NDO algorithms (the only, unavoidable exception being a stopping parameter which depends on the scaling of the Lagrangian function). After a few preliminary experiments (described below) on a subset of the instances, all the instances have been run with the same set of parameters, mostly set to the "default" values advised for a generic problem.

## 5.1. Preliminary results

Before running the large-scale comparison on all the instances, we performed numerous tests on a selected set of instances in order to gather a first understanding of the behavior of the approaches. We briefly report the findings of this first phase of the experiments, without showing any detailed table of results in order not to clutter the presentation.

**1)** The block dual-ascent approach is not computationally efficient: each iteration is indeed very fast, but only the first very few iterations actually result in a sizable decrease of the upper bound. All the subsequent iterations obtain only a negligible—even if in general non-null—improvement, resulting in an extremely poor overall rate of convergence to an upper bound of very low quality. This holds true for all instances and for all methods for selecting the next root that we tested. This behavior can probably be explained by the fact that each iteration takes into account only a "small" block of the variables, resulting in an approach akin to the coordinate ascent one, which is well-known to be inefficient.

**2)** Concerning the way of exploiting the rooted semimetric relaxation, the experiments have shown that the rate of convergence of both Lagrangian approaches applied to (12) is much slower than the rate of convergence of the same approaches applied to (9) or (10). More in general, the rate of convergence of the Lagrangian approaches applied to (14) suffers a very sharp decrease passing from $|R| = 1$ to $|R| > 1$, and seems to slowly deteriorate as $|R|$ increases. Since the cost of solving the Lagrangian subproblem also increases with $|R|$, the only computationally viable choices for $|R|$ are 0 and 1, corresponding to (10) and (9). In other words, the rate of convergence of the hybrid approach with $|R| \leq 1$ is surprisingly high, while for $|R| > 1$ (and, in particular, for $|R| = n - 1$) is the "normal" low rate that can be expected for a large-scale NDO problem. This is probably at least partly explained by the fact that the optimal Lagrangian multipliers for most of the inequalities (3) are zero, and therefore they are set to their optimal value at the beginning of the algorithm (the all-0 vector is the starting point for both Lagrangian approaches), and possibly, the resulting inequalities are never even explicitly generated. By contrast, the optimal Lagrangian multipliers for the equality constraints in (13) are most likely to be nonzero, and therefore they have to be (painfully) found by the algorithm. Yet, for "small" values of $|R|$ only relatively "few" of the "difficult" multipliers are in the problem, so the sharp decrease in the rate of convergence remains somewhat surprising.

**3)** The choice of the root node in (9) is scarcely relevant: a simple heuristic (choosing the node with largest sum of the costs of the incident arcs) appear to consistently provide a good root node. Similarly, the low rate of convergence of the hybrid approach with $|R| > 1$ does not seem to be materially influenced by the choice of the nodes in $R$. Therefore we decided not to test "root hopping" techniques, also in view of the already good results that we have obtained.

**4)** A suitable value for the critical parameter $h$ (the maximum number of violated inequalities to be inserted at each Lagrangian variables generation) appears to be the same used for the LP-based approach. For the Lagrangian approaches, however, it is also necessary to decide when attempting to perform the separation, since waiting for an "optimal" solution of the Lagrangian dual restricted to the current set of variables is rather inefficient. Indeed, we found that the simple rule of attempting separation every a few iterations of the NDO algorithm (typically 10) worked nicely. Furthermore, for the Lagrangian approaches removal strategies do appear to improve performances; once again, a simple rule such as eliminating all variables that have been set to 0 for five consecutive iterations of the NDO algorithm appears to be adequate.

**5)** The subgradient algorithm is, on most instances, incapable of obtaining solutions with an arbitrary precision. While the "default" parameters normally produce an upper bound of medium-to-good quality reasonably fast, no of the parameter settings we tried was capable of substantially improving on it; only very minor gains could be obtained, but at a very high computational costs. Therefore, we decided to accept the solution produced by the default parameters. The bundle algorithm, instead, was always able to produce solutions whose precision was in excess of the `1-e8` required by its stopping parameters. We remark that such a precision is very high for the solution of a NDO problem, and generally comparable with that obtained by interior-point algorithms for linear programming.

After the preliminary experiments, we could therefore discard the block dual-ascent approach and the hybrid approaches with $|R| > 1$. We also had a reasonable set of parameters for both the Lagrangian and the LP-based approaches to run all the instances with.

## 5.2. Results of the large-scale experiments

The results of the experiments are shown next. All the codes have been written in C/C++ and compiled with `gcc 3.3` using `-O2` optimization. CPLEX 9.0, the LP solver that we used, was as usual only available as a library. The experiments have been performed on a PC sporting an Athlon MP 2400+ processor and 1Gb of RAM, running Linux.

In the table, each row is labeled by $\langle\text{type}\rangle n$, where $n$ indicates the number of nodes and $\langle\text{type}\rangle$ denotes the instance class: "`c`" for clique graphs, "`p`" for planar graphs, "`s`" for simplex graphs, "`g2-pm`" for toroidal 2D-grid graphs with $\pm 1$ costs, "`g2-g`" for toroidal 2D-grid graphs with Gaussian costs, "`g3-pm`" for toroidal 3D-grid graphs with $\pm 1$ costs and "`g3-g`" for toroidal 3D-grid graphs with Gaussian costs. For all columns, the entries of each row correspond to the average among all the instances of the corresponding class (15 for planar graphs, 5 for all the others).

For each algorithm we report, in the column labelled "`Time`", the total time in seconds required to solve the problem, excluding the loading time. For the Lagrangian approaches this includes the time required for running the projection heuristic for finding the feasible primal solution; we deemed it unnecessary to report this time separately because it was always a very small fraction of the total time. For the LP-based algorithms, the reported time does not include any "crossover" procedure for generating a more "accurate" basic solution out of the interior one produced by the algorithm; the solutions obtained by this approach are of completely comparable quality with those obtained by the Lagrangian ones. For the latter approaches, the columns labelled "`DGap`" and "`PGap`" report the obtained (relative) dual and primal gaps, respectively, i.e.,

$$\begin{aligned} \text{DGap} &= \frac{\text{obtained lower bound} - v(6)}{v(6)} \\ \text{PGap} &= \frac{v(6) - \text{cost of obtained primal solution}}{v(6)} \end{aligned}$$

where $v(6)$, the optimal value of the problem (6), has been computed "exactly" by the LP-based approach (with the dual simplex method). An empty entry corresponds to a gap not larger than `1e-10`.

In Table 5.2 we report the main comparison between solving (6) with CPLEX, both providing it the full formulation (column "`C0`") or by row generation (column "`C2`"), and the most promising of our proposed Lagrangian approaches. In particular, the columns labeled "`V0`" report the results obtained by using the subgradient approach for solving (10), the columns labeled "`V1`" report the results obtained by using the subgradient approach for solving (9), the columns labeled "`B0`" report the results obtained by using the bundle approach for solving (10) and, finally, the columns labeled "`B1`" report the results obtained by using the bundle approach for solving (9).

The following facts clearly emerge from the table:

**1)** The Lagrangian approaches are competitive with the LP-based ones. In particular, for the largest instances of each class the "`B1`" variant finds primal and dual solutions of very good quality at least four times faster, and up to two orders of magnitude faster, than the "`C0`" code. The results are even more impressive when compared to the row generation "`C2`" code, that is usually three to four times slower than the "`C0`" one, and that rapidly becomes the only available choice due to the memory requirements of the interior point methods.

**2)** Exploiting the efficient approaches for optimization over $\mathcal{M}^r(G)$, i.e., solving (9), is in general convenient with respect to not doing it, i.e., solving (10). This is especially true when using the bundle method, since the "`B1`" variant almost always finds primal and dual solutions of better quality than those found by the "`B0`" variant with a running time that can be more than two orders of magnitude smaller on the largest instances. This is due to the fact that the "`B1`" variant usually shows a much faster rate of convergence; since most of the computing time, using a bundle approach, is spent solving the master problem (whose cost is identical, for the same size, for the two variants), the extra cost of solving (4) at each iteration is largely compensated by the reduction in the number of iterations. The only exception to this rule are the clique graphs, where the two approaches obtain comparable precisions in comparable time, but the "`B0`" variant is usually faster, up to a factor of two on the largest instances. It seems that for these "completely unstructured" problems the ability of automatically setting a "small" block of dual variables to their optimal value is indeed not so important as for "more structured" ones, and therefore the extra cost incurred for solving (4) is not justified. A somewhat analogous picture can be drawn comparing the "`V0`" and "`V1`" variants: the latter most often obtains a much better (dual) precision than the former. Once again, the only exception are the clique graphs, where the obtained precisions are always very similar. However, because in the subgradient algorithm the computing time for solving the Lagrangian problem is a significant fraction of the total time, this is obtained at the cost of an increase in the running time, up to more than a factor of two. Hence, on the clique graphs the "`V0`" variant obtains comparable (albeit slightly worse) precisions than the "`V1`" variant in significantly less time.

**3)** Comparing the best bundle variant (i.e., "`B1`" except for clique graphs) with the best subgradient variant (ditto) we see that most often the bundle method obtains much better dual precision in comparable or even less time. This is partly due to the fact that the bundle method converges more rapidly, but most importantly due to the fact that the bundle method has an effective stopping criterion – producing a(n almost) feasible (almost) optimal primal solution – that allows it to determine when optimization may be stopped. The bundle code may reach convergence in as little as 10 iterations, and on average requires less than 250 iterations to find a proper primal solution. By contrast, the volume algorithm never terminates before 500 iterations; typically, the last 500 ones are non descent ones, where the algorithm is only waiting to have reached the minimum number of consecutive non descent iterations required to declare the best point "optimal". So, while the bundle terminates by having constructed a proof of optimality, the volume algorithm terminates because it is no longer capable of improving the dual solution; note that in theory it also would eventually produce a primal solution of comparable quality, but this would take a very long time. This is clearly reflected by the precision of the obtained primal solutions: the subgradient approach never produces solutions of even moderate quality, whereas the bundle approach always produces solutions of acceptable, and most often of excellent, quality. However, the cost per iteration of the bundle code can be significantly larger than that of the subgradient algorithm, due to the master problem cost; in fact, especially for the largest instances the subgradient can be significantly faster, and in particular, it is a factor of 2 faster on "`g2-g144`" instances, almost a factor of 3 faster on "`g2-pm144`" instances, and almost an order of magnitude faster on "`c150`" instances. Note that in the latter two cases the obtained dual precision (not to mention the primal precision) is much worse, whereas in the first case it is comparable. Thus, on selected instances or if a rough bound has to be obtained

| | C0 | C2 | V0 | | | V1 | | | B0 | | | B1 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Time | Time | DGap | Pgap | Time | DGap | Pgap | Time | DGap | Pgap | Time | DGap | Pgap | Time |
| c25 | 0.28 | 0.12 | 2e-7 | 8e-3 | 0.04 | 1e-7 | 6e-3 | 0.47 | | 1e-8 | 0.39 | | | 0.27 |
| c50 | 5.53 | 7.41 | 8e-4 | 7e-3 | 0.54 | 6e-4 | 5e-3 | 2.21 | 7e-9 | 1e-5 | 4.00 | 7e-9 | 7e-6 | 4.80 |
| c75 | 33.77 | 76.49 | 6e-4 | 4e-3 | 1.87 | 4e-4 | 3e-3 | 6.80 | 8e-9 | 6e-6 | 13.33 | 4e-9 | 5e-6 | 19.38 |
| c100 | 144.23 | 336.28 | 5e-4 | 5e-3 | 4.11 | 4e-4 | 5e-3 | 13.88 | 8e-9 | 9e-6 | 34.80 | 3e-9 | 1e-5 | 42.79 |
| c125 | 442.26 | 1757.09 | 5e-4 | 7e-3 | 8.54 | 3e-4 | 3e-3 | 26.43 | 2e-9 | 1e-5 | 86.45 | 1e-9 | 1e-5 | 122.06 |
| c150 | 1192.58 | 4223.48 | 5e-4 | 5e-3 | 14.90 | 3e-4 | 3e-3 | 50.35 | 4e-9 | 1e-5 | 138.03 | 1e-9 | 2e-5 | 256.41 |
| p50 | 6.89 | 15.73 | 5e-8 | 8e-3 | 0.50 | | 3e-3 | 0.99 | | 7e-9 | 4.11 | | | 0.67 |
| p100 | 212.29 | 817.36 | 4e-6 | 3e-2 | 5.58 | | 2e-2 | 6.20 | | 1e-7 | 1557.56 | | | 5.04 |
| p150 | 1907.96 | 8907.30 | 1e-4 | 5e-2 | 24.97 | 2e-8 | 5e-2 | 21.53 | 2e-9 | 5e-6 | 5448.50 | | | 19.78 |
| s21 | 0.13 | 0.10 | | 3e-3 | 0.03 | | 5e-4 | 0.12 | | | 0.05 | | | 0.02 |
| s56 | 12.34 | 22.96 | | 1e-2 | 0.67 | 3e-9 | 2e-2 | 2.14 | | | 4.09 | | | 1.46 |
| s91 | 139.64 | 395.15 | 3e-7 | 2e-2 | 4.07 | | 2e-2 | 6.38 | | | 29.05 | | | 5.17 |
| s136 | 1114.76 | 4272.77 | 3e-5 | 3e-2 | 15.40 | | 1e-1 | 19.75 | | 2e-9 | 3577.35 | | | 31.34 |
| g2-pm25 | 0.29 | 0.36 | 4e-4 | 2e-3 | 0.06 | 2e-7 | 2e-3 | 0.31 | | 2e-9 | 0.42 | | | 0.08 |
| g2-pm49 | 6.41 | 14.09 | 2e-4 | 8e-3 | 0.61 | 1e-8 | 6e-3 | 1.71 | 7e-9 | 4e-7 | 7.31 | | | 1.21 |
| g2-pm81 | 73.96 | 224.16 | 7e-5 | 2e-2 | 2.92 | 7e-8 | 1e-2 | 5.69 | 7e-9 | 2e-6 | 6240.71 | | | 9.05 |
| g2pm-100 | 239.21 | 945.05 | 1e-3 | 5e-2 | 13.64 | 5e-6 | 5e-2 | 13.06 | 6e-9 | 2e-5 | 8960.96 | | | 11.71 |
| g2pm-144 | 1837.05 | 8624.21 | 1e-3 | 6e-2 | 23.03 | 1e-4 | 9e-2 | 50.02 | 3e-6 | 5e-4 | 11024.20 | 6e-9 | 1e-7 | 141.05 |
| g3-pm27 | 0.42 | 0.65 | 6e-4 | 2e-3 | 0.07 | 9e-8 | 4e-3 | 0.57 | 1e-9 | 3e-8 | 4.71 | | | 0.97 |
| g3-pm64 | 23.20 | 59.01 | 7e-5 | 2e-2 | 1.66 | | 4e-2 | 5.22 | 5e-9 | 3e-7 | 132.73 | | | 2.28 |
| g3-pm125 | 867.60 | 3812.58 | 1e-3 | 5e-2 | 16.88 | 2e-5 | 9e-2 | 52.03 | 5e-8 | 4e-6 | 6734.21 | | | 32.99 |
| g2-g25 | 0.31 | 0.21 | | 3e-3 | 0.05 | | 1e-3 | 0.15 | | | 0.10 | | | 0.05 |
| g2-g49 | 7.09 | 11.33 | | 7e-3 | 0.47 | | 5e-3 | 0.98 | | | 1.66 | | | 0.45 |
| g2-g81 | 79.54 | 290.91 | 1e-7 | 3e-2 | 2.77 | | 2e-2 | 4.44 | | | 15.11 | | | 2.99 |
| g2-g100 | 243.02 | 1158.78 | 6e-6 | 5e-2 | 6.02 | | 9e-2 | 6.93 | | | 90.01 | | 1e-8 | 7.37 |
| g2-g144 | 1766.64 | 9788.58 | 7e-5 | 7e-2 | 23.81 | 2e-9 | 1e-1 | 29.36 | | 2e-9 | 1528.10 | | 3e-8 | 60.84 |
| g3-g27 | 0.39 | 0.42 | | 4e-3 | 0.06 | | 5e-4 | 0.26 | | | 0.21 | | | 0.07 |
| g3-g64 | 25.54 | 62.52 | 3e-8 | 2e-2 | 1.39 | | 2e-2 | 3.63 | | | 10.24 | | | 1.98 |
| g3-g125 | 931.75 | 4302.72 | 4e-5 | 6e-2 | 16.20 | 1e-9 | 1e-1 | 39.02 | | | 288.21 | | | 26.85 |

Table 1: Main table of results

quickly, and especially as the size of the instances grow, subgradient-based approaches may still be of interest.

## 6. Conclusions and future research

We have proposed and tested several Lagrangian approaches for solving linear optimization problems over the semimetric polytope $\mathcal{M}(G)$ associated with a given graph $G$. Some of these approaches have been shown to be superior to state-of-the-art general-purpose linear programming codes. In order to provide competitive results, using the recently proposed efficient algorithms for solving linear optimization problems over the rooted semimetric polytope $\mathcal{M}^r(G)$ is in most cases ("structured" instances) instrumental, and a careful use of state-of-the-art Non Differentiable Optimization technology is required. As far as the choice between different NDO approaches goes, if accurate primal solutions are required a bundle approach, exploiting the projection heuristic, is also instrumental. The bundle approach is necessary for obtaining accurate dual solutions in several cases, and either competitive or downright faster in many others. However, on some large-scale instances, or if only a rough dual bound has to be obtained quickly, the subgradient algorithm may provide an interesting alternative.

Although we feel that the obtained results already clearly show the potential of Lagrangian approaches in this field, there are still a number of issues that need to be investigated for being able to properly assessing the value of these techniques for the solution of combinatorial optimization problems related to the semimetric polytope, such as the max-cut problem. In particular, implementing versions of this approach for very large, sparse graphs requires to substitute the (trivial) separation routine for the (polynomially many) triangle inequalities (3) with (less trivial) separation routine of the (exponentially many) cycle inequalities (1). The effect of this change on the relative efficiency of the approaches will have to be examined. Furthermore, a number of issues arise when embedding a Lagrangian-based approach within an enumerative approach, such as branch and cut, for a combinatorial problem [10] that will need to be properly resolved in order to being able to replace standard LP technology with the proposed algorithms. Finally, different—either general-purpose or specialized—NDO approaches may prove to be even more efficient than the ones that we have been using so far.

## References

[1] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, Englewood Cliffs, 1993.

[2] L. Bahiense, N. Maculan, and C. Sagastizábal, *The volume algorithm revisited: relation with bundle methods*, Mathematical Programming, 94 (2002), pp. 41–70.

[3] F. Barahona and R. Anbil, *The Volume Algorithm: Producing primal solutions with a subgradient method*, Mathematical Programming, 87 (2000), pp. 385–400.

[4] F. Barahona and A. Mahjoub, *On the cut polytope*, Mathematical Programming, 36 (1986), pp. 157–173.

[5] T. Crainic, A. Frangioni, and B. Gendron, *Bundle-based relaxation methods for multicommodity capacitated fixed charge network design problems*, Discrete Applied Mathematics, 112 (2001), pp. 73–99.

[6] M. Deza and M. Laurent, *Geometry of Cuts and Metrics*, vol. 15 of Algorithms and Combinatorics, Springer-Verlag, Berlin, 1997.

[7] O. du Merle, J.-L. Goffin, and J.-P. Vial, *On Improvements to the Analytic Center Cutting Plane Method*, Computational Optimization and Applications, 11 (1998), pp. 37–52.

[8] A. Frangioni, *Solving semidefinite quadratic problems within nonsmooth optimization algorithms*, Computers & Operations Research, 21 (1996), pp. 1099–1118.

[9] ——, *Generalized Bundle Methods*, SIAM Journal on Optimization, 13 (2002), pp. 117–156.

[10] ——, *About lagrangian methods in integer optimization*, Annals of Operations Research, submitted (2003).

[11] A. Frangioni and G. Gallo, *A bundle type dual-ascent approach to linear multicommodity min cost flow problems*, INFORMS Journal on Computing, 11 (1999), pp. 370–393.

[12] A. Frangioni, F. Glover, A. Lodi, and G. Rinaldi, *Optimal Semicuts*, Tech. Report OR-04-3, DEIS, Università di Bologna, 2004.

[13] A. Frangioni, A. Lodi, and G. Rinaldi, *Optimizing over semimetric polytopes*, in Integer Programming and Combinatorial Optimization - IPCO 2004, D. Bienstock and G. Nemhauser, eds., vol. 3064 of Lecture Notes in Computer Science, Springer-Verlag, 2004, pp. 431–443.

[14] B. Gavish, *Augmented Lagrangian Based Algorithms for Centralized Network Design*, IEEE Transactions on Communications, 33 (1985), pp. 1247–1257.

[15] A. Goldberg, *An efficient implementation of a scaling minimum-cost flow algorithm*, Journal of Algorithms, 22 (1997), pp. 1–29.

[16] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer Verlag, 1988.

[17] M. Guignard, *Efficient cuts in Lagrangean 'Relax-and-Cut' schemes*, European Journal of Operational Research, 105 (1998), pp. 216–223.

[18] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms*, vol. 306 of Grundlehren Math. Wiss., Springer-Verlag, New York, 1993.

[19] D. E. Knuth, *The Stanford GraphBase: a platform for combinatorial computing*, ACM Press, 1993.

[20] T. Larsson, M. Patriksson, and A.-B. Strömberg, *Ergodic, primal convergence in dual subgradient schemes for convex programming*, Mathematical Programming, 86 (1999), pp. 283–312.

[21] C. Lemaréchal, *Lagrangian relaxation*, in Computational Combinatorial Optimization, M. Jünger and D. Naddef, eds., Springer-Verlag, Heidelberg, 2001, pp. 115–160.

[22] F. Liers, M. Jünger, G. Reinelt, and G. Rinaldi, *Computing exact ground-states of hard Ising spin-glass problems by branch and cut*, in New Optimization Algorithms in Physics, A. Hartmann and H. Rieger, eds., Wiley-VCH Verlag, Berlin, 2004. In press.

[23] M. Lomonosov, *Combinatorial approaches to multiflow problems*, Discrete Applied Mathematics, 11 (1985), pp. 1–93.

[24] G. Rinaldi, *Rudy*. `http://www-user.tu-chemnitz.de/~helmberg/sdp_software.html`.