Mathematical Programming manuscript No.
(will be inserted by the editor)

A. Frangioni · A. Lodi · G. Rinaldi

New approaches for optimizing over the semimetric polytope

the date of receipt and acceptance should be inserted later

Abstract The semimetric polytope is an important polyhedral structure lying at the heart of several 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, 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 $F \subseteq E$, let $x(F) = \sum_{e \in F} x_e$. The *semimetric polytope* $\mathcal{M}(G)$ associated with G is defined by the following linear system:

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

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

The inequalities (1) are called the *cycle inequalities.*

If G is complete, the only chordless cycles of G are its triangles, therefore $\mathcal{M}(G)$ is defined by the *triangle inequalities*

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

Antonio Frangioni: Dipartimento di Informatica, Università di Pisa, largo Bruno Pontecorvo 3, 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`).

Mathematics Subject Classification (1991): 20E28, 20G40, 20C20

The paper deals with the problem of developing efficient algorithms to solve

$$\max\{\, cx \mid x \in \mathcal{M}(G) \,\} \tag{4}$$

for any given $c \in \mathbb{R}^E$. We briefly mention two applications of (4) related to two difficult combinatorial optimization problems:

- *Max-cut*: 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 G; W and $V \setminus W$ are called the *shores* of the cut. The *max-cut* problem is to find a cut $\delta(W^*)$ of G having maximum weight $c(\delta(W^*))$. The convex hull of the incidence vectors of all cuts of G is the *cut polytope* $CUT(G)$ associated with G (see, e.g., [7]); then, max-cut can be formulated as the linear program

  $$\max\{\, cx \mid x \in CUT(G) \,\} \ .$$

  It is not difficult to see that $CUT(G) \subseteq \mathcal{M}(G)$, the inclusion being strict for $|V| > 4$. Thus, (4) produces an upper bound on the optimal value of max-cut, that can be exploited at each node of a branch and cut scheme. Actually, in all the computational studies concerning instances of max-cut for very large sparse graphs based on a branch and cut scheme (see, e.g., [3, 21]), 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 function $d \in \mathbb{R}^E$ are given, a feasible *multiflow* in the network defined by G, q, and d exists if and only if the *metric inequality* $\mu \cdot (q - d) \geq 0$ holds for every *metric* μ on V, i.e., for every point of the cone defined by all the homogeneous inequalities in (3) [22]. 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.

In the case of a complete graph, (4) is a linear program with a polynomial number ($4\binom{|V|}{3}$) of constraints; still, it turns out to be surprisingly difficult to solve with standard LP tools, such as simplex or barrier algorithms, even if state-of-the-art software is used (cf. Section 5). 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 to (4) faster than it is currently doable with standard methods.

A possible technique is the Lagrangian approach where all the triangle (or cycle) inequalities are "dualized" leaving, as explicit constraints, only the (possibly redundant) upper and lower bounds on the variables. This technique has been successfully applied in [2], where the "Volume Algorithm" is used to solve the Lagrangian dual. We propose a two-pronged modification of this approach. First, we dualize only a subset of the cycle inequalities, leaving, as explicit constraints, the inequalities that define the *rooted semimetric polytope*, i.e., those associated with a given node of G, the root. Then, to solve the Lagrangian dual, besides the Volume Algorithm we also use a bundle algorithm. Combining these tools with a projection-type heuristic for quickly constructing feasible primal solutions, we obtain very accurate primal and dual solutions to (4) in a small fraction of the time required for the same task by 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.

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 general scheme for exploiting the available efficient algorithm for optimization over the rooted semimetric polytope in order to solve (4). Then, in Section 4 the relevant aspects of the implementation of the proposed approaches are discussed, in Section 5 the obtained computational results are presented and, finally, in Section 6 some conclusions are drawn.

2. Semimetric polytopes

The triangle inequalities for a complete graph with n vertices are $O(n^3)$; therefore, already for graphs of a few hundred nodes, they are too many to be handled explicitly, and the use of a row generation 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 a $O(1)$ violation check for each inequality in a set of polynomial size. For a general graph G, the cycle inequalities are exponentially many; however, separation is still polynomial [4], as it amounts to finding at most n shortest paths in a graph that has twice the size of G. By the polynomial equivalence between separation and optimization [14], it follows that (4) is solvable in polynomial time. As noticed, this task may be very time consuming for standard LP codes combined with cutting plane techniques. On the other hand, at present it seems very difficult to design a purely combinatorial algorithm, which would be extremely desirable for more effective computations. To the contrary, purely combinatorial algorithms have 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 (otherwise we 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 having much less defining inequalities, $\mathcal{M}^r(G)$ still is an integer linear programming formulation of max-cut; that is, every integral point in $\mathcal{M}^r(G)$ is the incidence vector of a cut of G. This formulation is minimal, i.e., the removal of any of its inequalities allows integral feasible solutions that are not incidence vectors of cuts of G.

Therefore, denoting by $A^r x \leq b^r$ the set of constraints that define $\mathcal{M}^r(G)$, the linear program

$$\max\{\ cx \mid A^r x \leq b^r\ \} \tag{5}$$

yields an upper bound on the value of an optimal cut, as (4) does. However, in most practical cases the latter bound is far weaker than the former. On the other hand, it can be shown that (5) can be solved by means of either a single min-cost flow computation [13] or a single max-flow computation [5] on different auxiliary directed graphs which have roughly twice the size of G. Thus, (5) is solvable by purely combinatorial algorithms, that turn out to be very efficient in practice. However, we are rather interested in solving (4), that is

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

Note that any two blocks of constraints $A^r x \leq b^r$ and $A^q x \leq b^q$ are not disjoint; in fact, it is easy to verify that every $n-1$ blocks (without loss of generality, the first $n-1$) contain all the constraints (3). In order not to overburden the notation, we will assume that only one of each replica is actually considered; accordingly, we will write

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

for the dual of (6), although in principle any two subvectors y^r and y^q of the vector of dual variables y, corresponding to blocks r and 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 (few) remaining ones. The presence of “easy” structures embedded into a “more complex” problem is a common occurrence in optimization, and it is often exploited by means of *Lagrangian* approaches such as those described next.

3. Solving the semimetric problem

We will use Lagrangian relaxation in order to exploit the algorithmic results on (5) for solving (6); the reader is therefore assumed to be familiar with Lagrangian duality (see, e.g., [11,16,20]). When an optimization problem exhibits a structure whereby its constraints can be partitioned into a number of “easy” blocks, there are two basic routes for exploiting such a structure:

- *Lagrangian relaxation*: keep one block of the constraints, relax all the others;
- *Lagrangian decomposition*: introduce one copy of the variables for each “easy” block plus constraints ensuring that all the copies attain the same value, then relax these constraints.

It should be remarked that in (6) the blocks are “many”, “small”, and they all possess the same structure; for instance, in a graph with 100 nodes we have 99 blocks of constraints, each containing roughly 1% of the constraints. This contrasts with most of the usual applications (e.g., [16]) where the blocks are "few", "large", and typically have different structures. Thus, while the choice among the different possibilities is usually dictated by considerations about the different structures of the blocks (plus those on the quality of the obtained bounds, that do not apply here because we are solving a convex program), in this case there is no reason, a priori, to prefer one route over the other. Thus, we have experienced with a family of relaxations which combine the two ideas. For this we select a set $R \subseteq V$ of root nodes—without loss of generality, the first k nodes—and consider the following equivalent form of (6):

$$\begin{array}{lll} \max \bar{c} \sum_{r \in R} x^r & & \\ \quad A^r x^r \leq b^r & r \in R & \\ \quad x^r = x^{r+1} & r = 1, \ldots, k-1 & \\ \quad A^q \left(\frac{1}{k} \sum_{r \in R} x^r \right) \leq b^q & q \notin R & \end{array} \tag{8}$$

where $\bar{c} = \frac{c}{k}$. The problem has k copies of the original variables x linked by $k-1$ blocks of equality constraints, plus all the constraints (3) not "covered" by the blocks in R; in these, it is convenient to express the identical value of all variables blocks x^r in terms of the average of the duplicated variables. The Lagrangian relaxation of (8) with respect to all the $x^r - x^{r+1} = 0$ constraints *and* the blocks out of R

$$z(\pi, y) = \sum_{r \in R} \max_{x^r} \left\{ \bar{c}^r x^r - \frac{1}{k} \sum_{q \notin R} y^q (b^q - A^q x^r) \mid A^r x^r \leq b^r \right\} \tag{9}$$

$$\text{where} \qquad \bar{c}^r = \begin{cases} \bar{c} + \pi^1 & \text{if } r = 1 \\ \bar{c} - \pi^k & \text{if } r = k \\ \bar{c} + \pi^i - \pi^{i-1} & \text{otherwise} \end{cases}$$

can be solved by means of k optimizations over $\mathcal{M}^r(G)$, with k distinct roots. The corresponding Lagrangian dual

$$\min \left\{ z(\pi, y) \mid y \geq 0 \right\} \tag{10}$$

is a large-scale Non Differentiable Optimization problem, with $O(kn^2)$ unconstrained variables π^r, corresponding to the $x^r = x^{r+1}$ constraints, plus $O((n-k)n^2)$ constrained variables y^q, corresponding to all the blocks $q \notin R$. This approach offers an "handle", namely $|R|$: for $|R| = n-1$ we obtain the Lagrangian decomposition, for $|R| = 1$ we obtain the Lagrangian relaxation, and for any value in between we obtain a hybrid approach. Furthermore, $R = \emptyset$ gives

$$z(y) = \max_{x} \left\{ cx - \sum_{r=1}^{n-1} y^r (b^r - A^r x) \mid x \in \{0,1\}^E \right\} , \tag{11}$$

i.e., the relaxation of [2]. In the following, we will refer to problem (10) with $|R| = r$ as (D_r).

Clearly, the availability of such a family of relaxations immediately rises some questions: what is a good value for $|R|$? Furthermore, once the cardinality is chosen, how should one select the elements of R? We remark that, especially for (D_1) (the "pure" Lagrangian relaxation), it would in principle be possible to change the root r during the algorithm: in fact, one is seeking for a full optimal dual vector $[y^1, \ldots, y^{n-1}]$ for (7), and dynamically changing r only influences which of the blocks of dual variables is "controlled" by the solver of the Lagrangian problem rather than by the NDO algorithm. However, for each r a different Lagrangian function z_r is defined, and it is not entirely straightforward to adapt the NDO algorithms to deal with such a family of related functions.

In general, the answers to the above questions may vary according to the way in which the Lagrangian dual is solved. Thus, in the next section we will report on the most important aspects of our implementation of the proposed approach.

4. The Lagrangian approach

The proposed approach requires to solve a Lagrangian dual, i.e., a (large-scale) Non Differentiable Optimization problem; thus, the choice of the proper NDO algorithm (and some implementation details) is crucial for its effectiveness.

4.1. Non Differentiable Optimization algorithms

Most NDO algorithms are based on a well-known principle in nonlinear optimization: construct a (local) *model* of the function $z(y) : \mathbb{R}^m \to \mathbb{R}$ (for simplicity of notation we do not distinguish the two types of dual variables) to be minimized, then use the information provided by the model to determine a direction of improvement and possibly a stepsize along it. The most popular model in NDO is the *cutting plane* approximation of z, iteratively constructed by solving (9) at a sequence of points $\{\bar{y}_i\}$. The corresponding solutions $\{[\bar{x}_i^r]_{r \in R}\}$ (for $R = \emptyset$, the solution of (11)) provide the function values $z(\bar{y}_i)$, and, via the violation of the relaxed constraints, the *subgradients* $g_i \in \mathbb{R}^m$, from which the linear underestimators $l_i(y) = z(\bar{y}_i) + g_i(y - \bar{y}_i)$ of z can be constructed. Thus, $z_i(y) = \max\{l_h(y), h \leq i\}$ is a polyhedral lower approximation to z. Using the minimum of z_i as the next point $\bar{y}_{i+1}$ where to evaluate z gives the classical cutting-plane algorithm, perhaps better known for its specialized version for structured linear programs: Dantzig-Wolfe's decomposition method. Minimizing z_i corresponds to solving a linear program, the *master problem*, with i columns and m rows; this may end up being very costly. Furthermore, due to the relatively poor performances of the cutting-plane method in practice, a number of related approaches have been developed to improve upon it. The form of the master problem is the most prominent aspect that differentiates these approaches: like the cutting-plane algorithm, some form of *bundle methods* use a linear program [10], most bundle methods [17,20] use a convex quadratic program, while other bundle methods [10] and algorithms based on "centers" [8,20] require a general nonlinear program, e.g., with logarithmic objective function. Therefore, even for an identical set of subgradients, the cost of solving the master problem can be very different according to the NDO algorithm employed.

In order to reduce the master problem cost, some bundle methods may reduce the number of stored subgradients by performing "aggregations": at each iteration, any number of the previously obtained subgradients can be discarded provided that the approximate subgradient $\tilde{g}_i$ to z corresponding to $[\tilde{x}_i^r = \sum_{h \leq i} \theta_h^r \bar{x}_i^r]_{r \in R}$, where the convex multipliers θ_i^r are produced by (the dual of) the master problem, is inserted in the master problem to replace them. Always keeping $\tilde{g}_i$ only gives rise to methods where the next tentative direction is a linear combination of the latest obtained subgradient and the direction used at the previous iteration [1]; in this case, the master problem becomes solvable by a closed formula. Algorithms of this type are known as *subgradient methods*, and have a long history of theoretical analysis [18] and practical application, in particular to combinatorial optimization (e.g., [2,6,15,16]). Subgradient methods and cutting-plane type approaches are technically different in many respects, as several important algorithmic details—step-size selection rules, stopping conditions, and the like—must be chosen in different ways for ensuring convergence. However, for the purpose of the current analysis it is reasonable to treat subgradient methods just as "poor-man" versions [20] of cutting-plane type algorithms. Cutting-plane type methods will typically show a faster convergence rate than subgradient methods at the expense of a higher cost per iteration [6,20]; since the trade-off for our application was hard to evaluate a priori, we have implemented the Lagrangian approach with two NDO solvers:

– a general-purpose *generalized bundle* solver (developed by the first author);
– a version of the Volume algorithm of Barahona and Anbil [2], derived by the code made available by the authors.

4.2. Finding primal feasible solutions

An important characteristic of both approaches is that the sequence of solutions $\{x_i^* = \sum_{r \in R} \tilde{x}_i^r / |R|\}$ that they produce is guaranteed (possibly taking subsequences) to converge to a primal optimal solution to (6). Each x_i^* is typically not feasible with respect to the relaxed constraints, although most often it tends to become quickly at least "almost feasible". Thus, the NDO algorithms asymptotically provide optimal solutions to (6); however, when they are finitely stopped the available primal solution x_i^* is most often not even feasible. This is especially true for subgradient approaches which, as shown in the next section, often terminate relatively far from dual optimality. The situation for the bundle algorithm is different, as its theoretical finite convergence properties often show up in practice by providing an actually feasible x_i^*; however, this is not always the case, and the algorithm may terminate with a x_i^* that is not feasible within the typical relative precision provided by linear programming solvers (say, `1e-8` for barrier and `1e-12` for simplex). The final accuracy depends on some of the algorithmic parameters: however, normally a very accurate dual solution is found much earlier than an accurate primal solution, thus setting the parameters in such a way as to require an accurate x_i^* may result in a considerable increase in the number of iterations.

To overcome these difficulties, we developed and tested a very simple projection approach for producing a feasible primal solution out of x_i^*: the primal solution is iteratively projected, using simple closed formulae, upon each violated (to within `1e-12`) inequality (3) until no more violated inequalities are found. Note that this process does not depend on which NDO approach has been used, nor from which $|R|$ has been chosen. Since when G is complete, as it is the case in our computational study (see Section 5), each inequality has only three nonzero coefficients, checking the violation and projecting one point over an inequality requires $O(1)$. This simple approach proved to be quite efficient: in all our experiments it always found a feasible solution with objective function value very close to that of the starting infeasible x_i^* in a very small fraction of the time required by the NDO algorithm.

4.3. Lagrangian variables generation

A relevant 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 can be more than two millions dual variables. However, at the optimum the vast majority of these is expected to be zero; therefore, the solution of (10) may greatly benefit from a dynamic generation of Lagrangian variables (already used with success, e.g., in [12]). This simply amounts at choosing a (small) *active set* $\mathcal{A} \subset \{1, \ldots, m\}$ and performing some iterations of the minimization of z restricted to the subspace of the active variables, that is, $z_{\mathcal{A}}(y_{\mathcal{A}}) = z([y_{\mathcal{A}}, 0])$; after that, a check is performed to find if new variables have to be added to $\mathcal{A}$. This corresponds to (approximately) solving a relaxation of (6) where all constraints but those in $\mathcal{A}$ (and, of course, those of the blocks associated with R) are completely disregarded, and then find if some of these are violated; hence, this is an ordinary row generation approach for the solution of (6).

Generating new Lagrangian variables corresponds to separating violated inequalities (3); for this, a primal solution is needed. The solution of the latest (9) could be used, but there was no guarantee that it provides a “good” input for separation routines [15]. Instead, we have found that x_i^* provides a completely satisfactory input for the separation routines.

Our preliminary results have shown that such a strategy has a considerable impact on the overall efficiency of the algorithms. The improvement is 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 the improvement for the bundle approach is in part due to the use of the specialized code of [9], which employs a two-level active-set approach particularly well-suited for efficiently dealing with the changes in the master problem corresponding to Lagrangian variables creation/destruction. Bundle approaches using non-specialized codes for solving the master problem might have benefited less from the dynamic generation of Lagrangian variables.

5. Computational results

We empirically evaluated the proposed approaches on 5 different types of graphs: a) *clique* graphs, b) random *planar* graphs with density between 50% to 100% of the maximum, c) *simplex* graphs [19], 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 made adjacent. For the first three types the edge costs were drawn from a uniform random distribution with proper ranges. The last two types of instances come from the Statistical Physics problem of analyzing the properties of a spin glass [21]; as it is customary, we experimented both with uniform random ± 1 costs and with Gaussian costs. Thus, we had a total of seven groups of instances. Within each group we produced instances with different number of nodes (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. All the instances have been produced by the machine-independent `rudy` random generator [23]; the corresponding parameters are available upon request from the authors.

In the instances, sparse graphs are "completed" by adding zero-cost edges, thereby producing a new instance in which the shores of a maximum cut define a maximum cut of the original (sparse) instance. This allowed us to use a trivial separation procedure, avoiding any possible artifact in the results due to the degree of sophistication of the algorithm used to separate the cycle inequalities. Furthermore, this 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), hence the instances in our test-bed are to be considered difficult.

5.1. Tuning of solution algorithms

Both NDO solvers have numerous algorithmic parameters that can be tuned to maximize their performances. In order to avoid distortion of the results, we refrained from instance- or even class-specific tuning; the only, unavoidable exception is a stopping parameter which depends on the scaling of the Lagrangian function. After preliminary experiments, all the instances have been run with the same set of parameters, mostly set to the "default" values advised for a generic problem.

We remark that for the subgradient algorithm the "default" parameters normally produce an upper bound of medium-to-good quality reasonably fast, but none 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. The bundle algorithm, instead, was always able to produce solutions with relative precision in excess of `1e-8`. For this to be true, however, a “large” maximum number of subgradients had to be allowed; although in principle that parameter provides a “knob” to finely tune the balance between the cost of the master problem solution and the overall convergence speed [6], in this case we basically had to allow the algorithm keep all the subgradients it would keep.

Apart from comparing the two NDO approaches, we also tested their efficiency against the state-of-the-art general-purpose linear programming code CPLEX 9.0. Although solving (6) with CPLEX may appear to be an easy exercise, we had to cope with the following nontrivial choices: a) should one supply a full formulation of (6) to the solver, or should a row generation approach akin to the Lagrangian variables generation be used? and b) which to choose among the three LP algorithms (primal or dual simplex, barrier)?

Preliminary tests showed that, somewhat surprisingly, solving the full formulation of the problem with the barrier 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. This is why in Section 5.3 we report the results of both the “static” approach and of the best of the row generation ones, that—again—uses the barrier algorithm (although dual simplex was often competitive, while primal simplex never was).

5.2. Preliminary results

From a preliminary test performed on a selected set of instances, we gathered the following understandings of the behavior of the approaches:

- The rate of convergence of the Lagrangian approaches suffers a very sharp decrease passing from $|R| = 1$ to $|R| > 1$, and further slowly deteriorates as $|R|$ increases; thus, the only computationally viable choices for $|R|$ are 0 and 1. This is probably explained by the fact that the optimal Lagrangian multipliers for most of the inequalities (3) are zero (the initial value), and possibly they never change. By contrast, the optimal Lagrangian multipliers for the equality constraints in (8) are most likely to be nonzero, and therefore they have to be (painfully) found by the NDO algorithm.
- Choosing the root node in (D_1) with a simple heuristic—the node with largest sum of the costs of the incident edges—appear to consistently provide results comparable with the best possible choice of r, as determined by running the algorithm n times with all possible roots and picking the best run. Therefore, dynamically changing r during the course of the algorithm does not appear to be promising, and it has not been tested.

Therefore, after the preliminary experiments we could discard the hybrid approaches with $|R| > 1$, leaving only the approaches (D_0) and (D_1) for the last phase of the experiments.

5.3. Results of the large-scale experiments

All the codes have been written in C/C++ and compiled with `gcc 3.3` using `-O2` optimization. CPLEX 9.0 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 Table 1, each row is labeled by $\langle\text{type}\rangle n$, where $n = |V|$ and $\langle\text{type}\rangle$ denotes the instance class: "`c`" for clique graphs, "`p`" for planar graphs, "`s`" for simplex graphs, "`g2-pm`" and "`g2-g`" for toroidal 2D-grid graphs with, respectively, ± 1 and Gaussian costs, and, analogously, "`g3-pm`" and "`g3-g`" for toroidal 3D-grid graphs. For all columns, the entries of each row correspond to the average among all the instances of the corresponding class.

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 finding the feasible primal solution with the projection heuristic; this was always, however, a very small fraction of the total. For the barrier algorithm, the reported time does not include any "crossover" procedure; the solutions obtained by this approach are of completely comparable quality with those obtained by the (best of the) Lagrangian ones. For the latter approaches, the columns labelled "`DGap`" and "`PGap`" report the obtained (relative) dual and primal gaps, respectively, against the optimal objective function value of (6) computed "exactly" with the dual simplex method. An empty entry corresponds to a gap not larger than `1e-10`.

In Table 1 we report the comparison between solving (6) with CPLEX, either providing it the full formulation (column "`C0`") or by row generation (column "`C2`"), and the most promising of the Lagrangian approaches: "`V0`" and "`V1`" are the results of the subgradient approach for solving (D_0) and (D_1), respectively, and similarly "`B0`" and "`B1`" for the bundle approach.

The following facts clearly emerge from the results:

– 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`" approach. The results are even more impressive against the row generation "`C2`" approach, that rapidly becomes the only available choice due to the memory requirements of the barrier algorithm.

– Solving (D_1) is in general convenient with respect to solving (D_0), especially when using the bundle method. This is due to the fact that approaches for (D_1) usually show a much faster rate of convergence, so that the extra cost of solving (5) at each iteration is largely compensated by the reduction in the number of iterations (for the bundle algorithm) and the increase in obtained dual precision (for the subgradient method). The only exception to this rule are the "completely unstructured" clique graphs, where the two approaches obtain comparable precisions in a comparable number of iterations, so that avoiding the extra cost for solving (5) turns out to be more convenient.

	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

- The best bundle variant (i.e., "`B1`" except for clique graphs) often obtains much better dual precision in comparable or less time than the best subgradient variant (ditto). This is due partly to the faster convergence of the bundle method, but most importantly to its much more effective stopping criterion. 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 takes less than 500 iterations, the last 500 ones producing no improvement in the dual solution.
- The subgradient approach never produces primal solutions of even moderate quality, whereas the bundle approach always produces primal solutions of acceptable, and most often of excellent, quality.
- The cost per iteration of the bundle code is significantly larger than that of the subgradient algorithm, so that the latter ends up being significantly faster on some of the largest instances: `g2-g144`, `g2-pm144`, and `c150`. Note that in the latter two cases the obtained dual accuracy (not to mention the primal accuracy) is much worse, whereas in the first case it is comparable.

Thus, on the selected instances Lagrangian approaches based on the rooted semimetric relaxation appear to be quite promising, especially when the Lagrangian dual is solved by a bundle method. If a rough bound has to be obtained quickly, and especially as the size of the instances grow, subgradient 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 most cases ("structured" instances), relaxations using efficient algorithms for linear optimization over the rooted semimetric polytope $\mathcal{M}^r(G)$ are the most efficient. Careful use of state-of-the-art Non Differentiable Optimization technology is required: a bundle approach is superior if accurate primal solutions are required, but it is also either competitive or downright faster in obtaining accurate dual solutions in many cases. On some large-scale instances, however, or if only a rough dual bound has to be obtained quickly, a 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 properly assessing the value of these techniques for the solution of combinatorial optimization problems such as max-cut. In particular, implementing this approach for sparse graphs requires to substitute the (trivial) separation routine for triangle inequalities with the (more complex) separation routine for cycle inequalities, whose effect on the relative efficiency of the approaches will have to be examined. Furthermore, a number of issues arise when embedding a Lagrangian approach within an enumerative algorithm (such as branch and cut) [11] that will need to be properly addressed if the Lagrangian approach is to replace standard LP technology. Finally, different NDO approaches may prove to be even more efficient than the ones that we have been using so far.

References

1. L. Bahiense, N. Maculan, and C. Sagastizábal, *The volume algorithm revisited: relation with bundle methods*, Mathematical Programming, 94 (2002), pp. 41–70.
2. F. Barahona and R. Anbil, *The Volume Algorithm: Producing primal solutions with a subgradient method*, Mathematical Programming, 87 (2000), pp. 385–400.
3. F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt, *An application of combinatorial optimization to statistical physics and circuit layout design*, Operations Research, 36 (1988), p. 493.
4. F. Barahona and A. Mahjoub, *On the cut polytope*, Mathematical Programming, 36 (1986), pp. 157–173.
5. A. Boros and P. Hammer, *Pseudo-boolean optimization*, Discrete Applied Mathematics, 123 (2002), pp. 155–225.
6. 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.
7. M. Deza and M. Laurent, *Geometry of Cuts and Metrics*, vol. 15 of Algorithms and Combinatorics, Springer-Verlag, Berlin, 1997.
8. 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.
9. A. Frangioni, *Solving semidefinite quadratic problems within nonsmooth optimization algorithms*, Computers & Operations Research, 21 (1996), pp. 1099–1118.
10. ———, *Generalized Bundle Methods*, SIAM Journal on Optimization, 13 (2002), pp. 117–156.
11. ———, *About lagrangian methods in integer optimization*, Annals of Operations Research, to appear (2005).
12. 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.
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. M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer Verlag, 1988.
15. M. Guignard, *Efficient cuts in Lagrangean 'Relax-and-Cut' schemes*, European Journal of Operational Research, 105 (1998), pp. 216–223.
16. ———, *Lagrangean relaxation*, TOP, 11 (2003), pp. 151–228.
17. J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms*, vol. 306 of Grundlehren Math. Wiss., Springer-Verlag, New York, 1993.
18. K.C. Kiwiel, *Convergence of Approximate and Incremental Subgradient Methods for Convex Optimization*, SIAM Journal on Optimization, 14 (2004), pp. 807–840.
19. D. E. Knuth, *The Stanford GraphBase: a platform for combinatorial computing*, ACM Press, 1993.
20. C. Lemaréchal, *Lagrangian relaxation*, in Computational Combinatorial Optimization, M. Jünger and D. Naddef, eds., Springer-Verlag, Heidelberg, 2001, pp. 115–160.
21. 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, pp. 47–69.
22. M. Lomonosov, *Combinatorial approaches to multiflow problems*, Discrete Applied Mathematics, 11 (1985), pp. 1–93.
23. G. Rinaldi, *Rudy*. `http://www-user.tu-chemnitz.de/~helmberg/sdp_software.html`.