Optimizing over Semimetric Polytopes

Antonio Frangioni[1], Andrea Lodi[2], Giovanni Rinaldi[3]

[1] Dipartimento di Informatica, Univertità di Pisa,
Largo B. Pontecorvo 1, 56127 Pisa, Italy
frangio@di.unipi.it

[2] D.E.I.S., University of Bologna,
Viale Risorgimento 2, 40136 Bologna, Italy
alodi@deis.unibo.it

[3] Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti" del CNR,
Viale Manzoni 30, 00185 Roma, Italy
rinaldi@iasi.cnr.it

1 Introduction

Let $G = (V, E)$ be a complete graph. Then the *semimetric polytope* $\mathcal{M}(G)$ associated with G is defined by the following system of inequalities

$$\left.\begin{array}{r} x_{ij} + x_{ik} + x_{jk} \leq 2 \\ x_{ij} - x_{ik} - x_{jk} \leq 0 \\ -x_{ij} + x_{ik} - x_{jk} \leq 0 \\ -x_{ij} - x_{ik} + x_{jk} \leq 0 \end{array}\right\} \quad \text{for all distinct } i, j, k \in V, \tag{1}$$

called the *triangle inequalities*. The paper deals with the problem of finding efficient algorithms to optimize a linear function over such a polytope.

We briefly mention two reasons to seek for an efficient way to optimize a linear function over $\mathcal{M}(G)$.

1. The polytope $\mathcal{M}(G)$, for $|V| > 4$, properly contains the *cut polytope* associated with G (see, e.g., [7] for the details). Thus, if an edge cost function $c \in \mathbb{R}^E$ is given, maximizing c over $\mathcal{M}(G)$ produces an upper bound on the maximum c-value cut of G, which can be exploited in branch and bound or branch and cut schemes 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 $\mathcal{M}(G)$, or, more precisely, its projection that will be described shortly later. Consequently, most of the computation time for finding a maximum cut is spent into a (possibly long) series of linear optimizations over $\mathcal{M}(G)$.
2. If an edge capacity function $c \in \mathbb{R}^E$ and an edge demand function $d \in \mathbb{R}^E$ are given, the existence of a feasible *multiflow* in the network defined by G, c, and d is established by the Japanese Theorem (see, e.g., [15]). According to this theorem, a feasible multiflow exists if and only if $\mu \cdot (c - d) \geq 0$ holds for every *metric* μ on V, i.e., for every point of the cone defined by all the homogeneous equations of (1). It is not hard to see that this is equivalent

to the condition $\min\{(c-d)x \mid x \in \mathcal{M}(G)\} \geq 0$. In some approaches to network design problems [2, 6] such a feasibility problem has to be solved several times. Again, this calls for an effective solution algorithm.

Although these problems are just standard linear programs with a polynomial number of constraints ($4\binom{|V|}{3}$), they turn out to be surprisingly difficult to solve with standard LP tools such as the simplex method or interior-point methods, even if state-of-the-art software is used. It is therefore of considerable interest to develop alternative algorithmic techniques that make it possible to compute (even with some degree of approximation) optimal primal and dual solutions of these LP's, faster than it is currently possible with standard methods.

An alternative technique is the Lagrangian approach where one "dualizes" all the triangle inequalities and leaves only the upper and lower bounds on the variables (which are actually redundant in the system (1)) as explicit constraints. Such an approach has been successfully applied, e.g., in [3], where a subgradient-type approach is used to solve the Lagrangian dual.

We propose to extend this approach in two ways. First, we dualize only a subset of the triangle inequalities, leaving, as explicit constraints, all the inequalities associated with the triangles of G that have a selected node in common. Such a system of inequalities defines the *rooted semimetric polytope*. We show later how a linear problem defined on this polytope can be solved efficiently. Then, we test the use of a bundle-type algorithm [8] as an alternative to subgradient-type approaches to solve the Lagrangian dual; since subgradient-type approaches can be seen as "special cases" of bundle-type ones [1], this alternative can be considered an extension. We show that, in most cases, bundle-type approaches, in comparison with subgradient-type ones, either produce primal and/or dual solutions of substantially higher accuracy, or reduce the running times significantly, or achieve both these results. The best Lagrangian approach we implemented is shown to obtain, on a large set of instances, primal and dual solutions of accuracy comparable with that produced by standard LP algorithms but in a small fraction of their computation time.

2 Optimizing over the Rooted Semimetric Polytope

If G is not complete, the semimetric polytope associated with G can be obtained by projecting the set defined by (1) into $\mathbb{R}^E$. The resulting polytope is defined as follows. Let $\mathcal{C}$ be the set of all chordless cycles of G and $\bar{E}$ the subset of the edges of G that do not belong to a 3-edge cycle (a *triangle*) of G. The following system of linear inequalities

$$x(C \setminus F) - x(F) \leq |F| - 1 \qquad F \subseteq C \text{ with } |F| \text{ odd and } C \in \mathcal{C} \tag{2}$$

$$0 \leq x_e \leq 1 \qquad e \in \bar{E} \tag{3}$$

defines the *semimetric polytope* $\mathcal{M}(G)$ associated with G. Obviously, $\mathcal{M}(G)$ contains the cut polytope associated with G and every its integral point is the incidence vector of a cut of G. The inequalities (2) are called *cycle inequalities*.

Although exponentially many, they can be separated in polynomial time (see [5]).

Let r be a selected node of G that will be donoted as the *root*. Without loss of generality, assume that node r is adjacent to every other node of G. If this is not the case, one can make r adjacent to each other node by adding new edges to the graph and by extending the edge weight vector c assigning a zero weight to the new edges. A cut in the new graph is readily associated with a cut in the original graph and the two cuts have the same weight.

Let us partition the set $\mathcal{C}$ into two subsets: the set $\mathcal{C}_r$ of the chordless cycles that contain node r and its complement in $\mathcal{C}$. Under the above assumption, the edge set $\bar{E}$ is empty and $\mathcal{C}_r$ contains only triangles. The *rooted semimetric polytope* $\mathcal{M}^r(G)$ associated with G is defined by the subsystem of the cycle inequalities defined by $\mathcal{C}_r$, i.e.,

$$\left.\begin{aligned} x_{ri} + x_{rj} + x_{ij} &\leq 2 \\ x_{ri} - x_{rj} - x_{ij} &\leq 0 \\ -x_{ri} + x_{rj} - x_{ij} &\leq 0 \\ -x_{ri} - x_{rj} + x_{ij} &\leq 0 \end{aligned}\right\} \quad \text{for all } ij \in E^r, \tag{4}$$

where E^r is the edge set of the subgraph of G induced by $V^r = V \setminus \{r\}$.

Despite the fact that the system (4) has much less inequalities than (1), it shares with (1) the property that it is satisfied by the incidence vector of any cut of G, while every its integral solution is the incidence vector of a cut of G. Therefore, the system (4) along with the integrality requirements provides an integer linear programming formulation for the max-cut problem. Thus, the optimization of $c \cdot x$ over $\mathcal{M}^r(G)$ yields an upper bound on the value of an optimal cut.

The system (4) has only $4|E^r|$ inequalities, but optimizing over the rooted semimetric polytope with a general purpose linear optimizer, although not as difficult as optimizing over $\mathcal{M}(G)$, is not as easy as one may expect. We first give a characterization of the vertices of $\mathcal{M}^r(G)$, then we outline a method to effectively solve the problem exploiting this characterization.

For any two disjoint subsets W and Z of V, the notation $(W : Z)$ stands for the set of edges with one endpoint in W and the other in Z. For a node set $W \subseteq V$, by $E(W)$ denote the set of all the edges in E with both the endpoints in W. If $Z = V \setminus W$, then the edge set $(W : Z) = (W : V \setminus W)$ is a *cut* of G and is also denoted by $\delta(W)$.

Definition 1. *A* semicut *of G is an ordered pair $\Omega = (F, J)$ of disjoint subsets of the edges of G satisfying the following condition: there exists an ordered pair (S, T) of disjoint subsets of V, with $r \in S$, such that*

(a) $F = (S : T) \cup H$, where $H \subseteq E(V \setminus (S \cup T))$, and
(b) $J = \delta(S \cup T)$.

The edges in F are called regular*, while those in J are called* weak*.*

If $S \cup T = V$, then the set of weak edges is empty and the semicut is the ordered pair of a cut of G and of the empty set. Since a semicut is fully specified by the ordered pair of node sets (S, T) and by the edge set H, it will be denoted by $\Delta(S, T, H)$.

With a semicut $\Omega = \Delta(S, T, H)$ a representative vector $\psi^\Omega \in \mathbb{R}^E$ is associated, where for each $e \in E$ the component ψ_e^Ω is equal to 1 if e is a regular edge, is equal to 1/2 if it is a weak edge, and is equal to 0 otherwise. If $S \cup T = V$, then H is empty and the characteristic vector of Ω is the incidence vector of a cut, namely, the cut $\delta(S)$. Conversely, the incidence vector of a cut $\delta(W)$ is the representative vector of the semicut $\Delta(W, V \setminus W, \emptyset)$. The *weight of a semicut* Ω is given by the inner product $c \cdot \psi^\Omega$.

A semicut of G is called *extreme* if its representative vector cannot be written as a convex combination of representative vectors of distinct semicuts of G. Not all semicuts are extreme. A characterization of the extreme semicuts and of the vertices of $\mathcal{M}^r(G)$ is given by the following theorems.

Theorem 1. *A semicut $\Omega = \Delta(S, T, H)$ is not extreme if and only if $U = V \setminus (S \cup T)$ is nonempty and there is a connected component $G_i = (U_i, E_i)$ of the subgraph of G induced by U for which $E_i \cap H$ is a cut of G_i.*

Corollary 1. *In a complete graph of n nodes the number of extreme semicuts for which $r \in S$ is given by*

$$2^{n-1} + \sum_{k=3}^{n-1} \binom{n-1}{k} (2^{\binom{k}{2}} - 2^{k-1}) 2^{n-k-1}$$

Theorem 2. *The set of vertices of $\mathcal{M}^r(G)$ coincides with the set of representative vectors of all the extreme semicuts of G for which $r \in S$.*

Let us now turn to the solution of

$$\max\{c \cdot x \mid x \in \mathcal{M}^r(G)\} \tag{5}$$

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:

$$\min \sum_{i \in V^r} (u_{ri} + u_{ri'}) + \sum_{ij \in E^r_+} (v_{ij'} + v_{ji'})$$

subject to

$$\left.\begin{aligned} u_{ri} + \sum_{ij \in E^r_+} v_{ij'} + \sum_{\substack{ki \in E^r_- \\ k<i}} w_{ki} - \sum_{\substack{ki \in E^r_- \\ k>i}} w_{ik} - q_{ir} &= d_i \\ -u_{ri'} - \sum_{ij \in E^r_+} v_{ji'} - \sum_{\substack{ki \in E^r_- \\ k<i}} w_{k'i'} + \sum_{\substack{ki \in E^r_- \\ k>i}} w_{i'k'} + q_{i'r} &= -d_i \end{aligned}\right\} i \in V^r$$

$$\left.\begin{aligned} v_{ij'} &\le 2|c_{ij}| \\ v_{ji'} &\le 2|c_{ij}| \end{aligned}\right\} ij \in E^r_+$$

$$\left.\begin{aligned} w_{ij} &\le 2|c_{ij}| \\ w_{i'j'} &\le 2|c_{ij}| \end{aligned}\right\} ij \in E^r_-$$

$$u, v, w, q \ge 0 \quad ,$$

where

$$d_i = c_{ri} + \sum_{ij \in E^r} c_{ij} - 2 \sum_{\substack{hi \in E^r_- \\ h<i}} c_{hi}. \tag{6}$$

Then we have the following

Theorem 3. *The optimal objective function value of MCFP equals twice the optimal value of (5).*

A sketch of the proof of this theorem is given in the Appendix. In addition, we can show that an optimal solution of (5) can be easily obtained from an optimal dual solution of MCFP, which in turn can be obtained from a primal optimal solution by exploiting the semicut structure. The characterization of the solutions of (5) as semicuts is useful also to devise an efficient algorithm for ranking all the solutions of (5) in the spirit of the scheme given, e.g., in [17] for ranking the s-t cuts in a graph.

Several polynomial time algorithms are available to solve MCFP. In particular, Cost Scaling algorithms run in $O(n^2 m \log nC)$ time, where n is the number of nodes of the graph, m the number of arcs, and C the largest absolute value of the components of the cost vector. Since in our case C is equal to 1, Cost Scaling algorithms solve (5) in strongly polynomial time.

We have performed extensive computational tests on several large-scale instances. In Table 1 we report a brief outcome of our testing on *clique* graphs, *planar* graphs, *simplex* graphs and *toroidal 2D and 3D-grid* graphs for spin glass problems, with either ± 1 or Gaussian costs (costs in $\mathbb{R}$ drawn from a Gaussian distribution), for a total of 39 instances. The name of each instance begins with "c", "p", "s", "g2-pm", "g2-g", and "g3-g", respectively, the rest of the name gives the size of the node set. We tested different algorithms for MCFP implemented under the `MCFClass` interface [9], as well as with a general purpose LP

solver. Specifically, we tested ILOG-Cplex 8.1 Network (CPXNET), the min-cost flow network simplex implementation in [14] (ZIB) and the Cost Scaling algorithm in the implementation of [11] (CS2), and the general purpose LP solver ILOG-Cplex 8.1 Barrier directly applied to (5). We do not report on the performance of general purpose Primal Simplex and Dual Simplex algorithms as prior tests have shown that for this kind of instances they run much slower than Barrier. Computing times are in seconds on a Pentium M 1.6 GHz notebook, and each entry is an average over three different instances.

The results in Table 1 clearly show that for these classes of instances the Cost Scaling algorithm actually provide better performances than simplex-type approaches for MCFP, and that all the min-cost flow algorithms outperform the general-purpose LP algorithm. The last column of Table 1 reports the speed factor of CS2 with respect to ILOG-Cplex 8.1 Barrier.

It is known that (5) provides a rather weak upper bound to the max-cut problem. Therefore, the algorithm described here to solve (5) has been designed primarily with the purpose of optimizing over (1), as it will be explained later. Nevertheless, there are instances of max-cut for which one may expect the bound given by (5) to be of some help in practical computation. These are, for example, instances where the weights of the edges incident with a selected node are, in absolute value, sensibly larger than all the other weights. These instances have received a considerable attention in the computational studies of max-cut (or of its equivalent problem, the unconstrained quadratic $0-1$ problem) (see, e.g., [4] and [16]). Using our algorithm embedded in a vanilla branch and bound scheme we obtained the results reported in Table 2.

Table 2 reports the result of the branch and bound approach based on the rooted semimetric relaxation on classical instances from the literature [16]. More precisely, instances denoted as "a" and "c" are the ones from [16] generated as in [4] and [18], while instances denoted as "b" are the ones from [16] generated as in [12]. Table 2 reports information about the problem size (n), the density of the graph (d), and the intervals of the random generated weights for the edges incident with the selected nodes and for the other edges, respectively. The last two columns of the tables give the outcome in terms of nodes and CPU time of the branch and bound algorithm. The results show that for those problems the rooted semimetric relaxation provides a very good bound allowing their effective solution.

3 Optimizing over the Semimetric Polytope

Let us denote the constraints (4) for one given root node r by $A^r x \leq b^r$. Then, linear optimization over (1) can be written as

$$\max_x \{cx \mid A^r x \leq b^r, \text{ for } r = 1, \ldots, n-1\} \tag{7}$$

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, only $n-1$ blocks suffice to "cover" all constraints in (1).

	directed graph		CPXNET	ZIB	CS2	ILOG-Cplex 8.1, Barrier			
Name	#nodes	# arcs	time	time	time	# var.s	# cons.s	time	speed-up
c150	299	22278.4	0.1	0.1	0.0	10990.2	43364.8	1.0	69.2
c500	999	246363.2	3.2	7.6	0.3	122682.6	488734.4	24.8	82.6
p10000	19999	96760.0	1.7	4.9	0.8	38380.7	113528.0	2.6	3.5
p20000	39999	193516.7	5.3	11.5	2.5	76758.0	227041.3	5.8	2.4
s39711	79421	603219.3	256.1	336.2	9.7	261899.7	888758.7	206.7	21.2
g2-pm22500	44999	179988.0	2.5	2.8	2.1	67495.0	179984.0	4.7	2.2
g2-pm62500	124999	499988.0	10.2	13.5	8.8	187495.0	499984.0	24.2	3.0
g3-pm64000	127999	639984.0	80.8	68.2	9.5	255993.0	767976.0	757.7	79.6
g3-pm8000	15999	79984.0	1.4	0.7	0.6	31993.0	95976.0	10.7	21.8
g2-g22500	44999	179986.7	8.1	12.4	2.9	67494.3	179981.3	5.9	2.0
g2-g62500	124999	499986.0	38.8	62.8	10.3	187494.0	499980.0	24.5	2.4
g3-g8000	15999	79983.3	3.8	5.1	0.8	31992.7	95974.7	11.9	15.1
g3-g64000	127999	639984.0	396.5	786.4	11.6	255993.0	767976.0	857.2	73.6

Table 1. Optimizing over the rooted semimetric polytope

Name	n	d	off-diagonal weights	diagonal weights	nodes	time
1a	50	.1	[-100,100]	[-100,100]	3	0.03
2a	60	.1	[-100,100]	[-100,100]	1	0.03
3a	70	.1	[-100,100]	[-100,100]	159	0.12
4a	50	.2	[-100,100]	[-100,100]	135	0.09
5a	30	.4	[-100,100]	[-100,100]	119	0.06
6a	30	.5	[-100,100]	[-100,100]	75	0.07
7a	100	.0625	[-100,100]	[-100,100]	1	0.03
1b	20	1.	[-100,0]	[0,63]	29	0.03
2b	30	1.	[-100,0]	[0,63]	67	0.05
3b	40	1.	[-100,0]	[0,63]	105	0.11
4b	50	1.	[-100,0]	[0,63]	115	0.20
5b	60	1.	[-100,0]	[0,63]	145	0.42
6b	70	1.	[-100,0]	[0,63]	177	0.87
7b	80	1.	[-100,0]	[0,63]	261	1.81
8b	90	1.	[-100,0]	[0,63]	397	5.12
9b	100	1.	[-100,0]	[0,63]	655	14.57
10b	125	1.	[-100,0]	[0,63]	885	51.53
1c	40	.8	[-50,50]	[-100,100]	2815	1.90
2c	50	.6	[-50,50]	[-100,100]	23275	19.29
3c	60	.4	[-50,50]	[-100,100]	6247	5.82
4c	70	.3	[-50,50]	[-100,100]	8737	8.08
5c	80	.2	[-50,50]	[-100,100]	2191	2.48
6c	90	.1	[-50,50]	[-100,100]	11	0.04
7c	100	.1	[-50,50]	[-100,100]	1	0.03

Table 2. Branch and bound based on the solution of the rooted semimetric relaxation

We 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

$$\min_y \left\{ \sum_{r=1}^{n-1} y^r b^r \mid \sum_{r=1}^{n-1} y^r A^r \geq c \right\} \tag{8}$$

for the dual of (7).

A possible way for solving (7) is to form the Lagrangian relaxation of (7) with respect to all other blocks of constraints but one

$$\max_x \left\{ cx - \sum_{h \neq r} y^h (b^h - A^h x) \mid A^r x \leq b^r \right\} \tag{9}$$

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

$$\min_y \{ v(9) \mid y \geq 0 \} \tag{10}$$

where $v(\cdot)$ denotes the optimal objective function value of a problem. This is equivalent to solving problem (8); the Lagrangian multipliers in (10) are precisely the dual variables of (8), except for the $O(n^2)$ dual variables y^r that are *implicitly* handled by the Lagrangian relaxation.

However, some issues arise:

- Problem (10) is a challenging large-scale Non Differentiable Optimization (NDO) problem, therefore it is natural to ask whether such approach can ever compete with a LP-based one?
- How should we choose the root r? Should we just select r and keep it fixed or a "root hopping" strategy would be preferable?
- Each of the constraint blocks $A^r x \leq b^r$ involves only a "few" ($O(n^2)$) of the "many" ($O(n^3)$) constraints of (7) or, put it in dual terms, the optimization over the single block in (9) can only set a "few" of the "many" variables of (8). Consequently, one wonders if solving (10) has a clear advantage over solving the equivalent Lagrangian Dual

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

  of (7) with respect to *all* blocks of constraints?
- Which NDO algorithm has to be used for solving the Lagrangian Duals?
- Are there alternative Lagrangian approaches based on reformulations of (7), such as Lagrangian Decomposition, that provide more convenient ways for solving (8)?

We experimentally evaluated the proposed approaches with a large-scale comparison. Due to the large number of constraints, each algorithm was embedded into a cutting plane scheme, where first a formulation with no explicit constraint is taken and then it is iteratively enlarged by adding explicit constraints violated by the current primal solution.

The test-bed was made by 175 instances of the same type of graphs described above, but in this context sparse graphs were "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.

Our experiments showed that these large-scale NDO problems, at least for the instances we tested, appear to be relatively easy to solve, even if a fixed root r is chosen with a simple heuristic. This is shown in Table 3, where the results obtained by the approach of [3] for solving (10) and (11) (columns "V1" and "V0"), those of a corresponding bundle-type approach (columns "B1" and "B0") and those of a finely tuned code using the state-of-the-art general-purpose LP solver ILOG-Cplex 8.1 (column "ILOG-Cplex") are compared. Computing times are in seconds on an Athlon MP 2400+.

The table clearly shows that the Lagrangian approaches, in particular the bundle-based one using (5) as subproblem (B1), provides solutions with very

	ILOG-Cplex	V0			V1			B0			B1		
	time	DGap	Pgap	time	DGap	Pgap	time	DGap	Pgap	time	DGap	Pgap	time
c25	0.28	2e-7	8e-3	0.04	1e-7	6e-3	0.47		1e-8	0.39			0.27
c50	8.97	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	69.03	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	386.60	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	1409.20	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	3729.51	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	13.80	5e-8	8e-3	0.50		3e-3	0.99		7e-9	4.11			0.67
p100	762.15	4e-6	3e-2	5.58		2e-2	6.20		1e-7	1557.56			5.04
p150	8877.42	1e-4	5e-2	24.97	2e-8	5e-2	21.53	2e-9	5e-6	5448.50			19.78
s21	0.08		3e-3	0.03		5e-4	0.12			0.05			0.02
s56	25.96		1e-2	0.67	3e-9	2e-2	2.14			4.09			1.46
s91	462.03	3e-7	2e-2	4.07		2e-2	6.38			29.05			5.17
s136	5492.87	3e-5	3e-2	15.40		1e-1	19.75		2e-9	3577.35			31.34
g2-pm25	0.28	4e-4	2e-3	0.06	2e-7	2e-3	0.31		2e-9	0.42			0.08
g2-pm49	15.01	2e-4	8e-3	0.61	1e-8	6e-3	1.71	7e-9	4e-7	7.31			1.21
g2-pm81	233.91	7e-5	2e-2	2.92	7e-8	1e-2	5.69	7e-9	2e-6	6240.71			9.05
g2pm-100	1218.55	1e-3	5e-2	13.64	5e-6	5e-2	13.06	6e-9	2e-5	8960.96			11.71
g2pm-144	9436.36	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	0.70	6e-4	2e-3	0.07	9e-8	4e-3	0.57	1e-9	3e-8	4.71			0.97
g3-pm64	59.23	7e-5	2e-2	1.66		4e-2	5.22	5e-9	3e-7	132.73			2.28
g3-pm125	3427.77	1e-3	5e-2	16.88	2e-5	9e-2	52.03	5e-8	4e-6	6734.21			32.99
g2-g25	0.28		3e-3	0.05		1e-3	0.15			0.10			0.05
g2-g49	13.98		7e-3	0.47		5e-3	0.98			1.66			0.45
g2-g81	187.43	1e-7	3e-2	2.77		2e-2	4.44			15.11			2.99
g2-g100	788.55	6e-6	5e-2	6.02		9e-2	6.93			90.01		1e-8	7.37
g2-g144	9050.67	7e-5	7e-2	23.81	2e-9	1e-1	29.36		2e-9	1528.10		3e-8	60.84
g3-g27	0.49		4e-3	0.06		5e-4	0.26			0.21			0.07
g3-g64	58.45	3e-8	2e-2	1.39		2e-2	3.63			10.24			1.98
g3-g125	3564.98	4e-5	6e-2	16.20	1e-9	1e-1	39.02			288.21			26.85

Table 3. Main table of results

low primal and dual relative gaps (columns "PGap" and "DGap", an empty entry in those columns corresponding to a gap not larger than `1e-10`) in a small fraction of the time required by the LP solver.

The table also shows that "x1" approaches are in general much more efficient than "x0" ones, with the notable exception of complete (clique) graphs. As far as the "Vx" versus "Bx" comparison is concerned, most often the bundle method obtains much better dual precision in comparable or even less time; furthermore, the subgradient approach never produces primal solutions of even moderately good quality, whereas the bundle approach always produces solutions of acceptable — and most often of excellent — quality. However, especially for the largest instances the subgradient can be significantly faster.

We remark that alternative, more complex Lagrangian approaches — that we do not describe here for space reasons — have been tested, but they have not been found to be competitive with those presented here.

The above analysis allows us to conclude that Lagrangian approaches for linear optimization over (1) are competitive with LP-based. On all "structured" instances, exploiting the efficient algorithms for (5) is instrumental. If accurate primal solutions are required a bundle approach is also instrumental; the bundle approach is also 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.

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. Barahona, F., *Network Design Using Cut Inequalities*, SIAM Journal on Optimization, 6 (1996), pp. 823–837.
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, M. Jünger, and G. Reinelt, *Experiments in quadratic 0–1 programming*, Mathematical Programming, 44 (1989), pp. 127–137.
5. F. Barahona and A. Mahjoub, *On the cut polytope*, Mathematical Programming, 36 (1986), pp. 157–173.
6. Bienstock, D., Chopra, S., Günlük, O., and Tsai, C., *Minimum Cost Capacity Installation for Multicommodity Network Flows*, Mathematical Progamming, 81 (1998), pp. 177–199.
7. M. Deza and M. Laurent, *Geometry of Cuts and Metrics*, vol. 15 of Algorithms and Combinatorics, Springer-Verlag, Berlin, 1997.
8. A. Frangioni, *Generalized bundle methods*, SIAM Journal on Optimization, 13 (2002), pp. 117–156.
9. Frangioni, A. and Manca, A., *A Computational Study of Cost Reoptimization for Min Cost Flow Problems*, INFORMS Journal on Computing, to appear.
10. D. Fulkerson, A. Hoffman, and M. McAndrew, *Some properties of graphs with multiple edges*, Canadian Journal of Mathematics, 17 (1965), pp. 166–177.

11. A. Goldberg, *An efficient implementation of a scaling minimum-cost flow algorithm*, Journal of Algorithms, 22 (1997), pp. 1–29.
12. V. Gulati, S. Gupta, and A. Mittal, *Unconstrained quadratica bivalent programming problem*, European Journal of Operational Research, 15 (1984), pp. 121–125.
13. D. Hochbaum, *Monotonizing matrices with up to two nonzeroes per column*, Operations Research Letters, 32 (2003), pp. 49–58.
14. A. Löbel, *Solving large-scale real-world minimum-cost flow problems by a network simplex method*, Technical Report SC-96-7, ZIB Berlin, 1996.
15. M. Lomonosov, *Combinatorial approaches to multiflow problems*, Discrete Applied Mathematics, 11 (1985), pp. 1–93.
16. P. Pardalos and G. Rodgers, *Computational aspects of a branch and bound algorithm for quadratic zero-one programming*, Computing, 45 (1990), pp. 131–144.
17. V. Vazirani and M. Yannakakis, *Suboptimal cuts: Their enumeration, weight and number*, in Proceedings of the 19th International Colloquium on Automata, Languages, and Programming, W. Kuich, ed., vol. 623 of Lecture Notes in Computer Science, New York, NY, 1992, Springer-Verlag, pp. 366–377.
18. A. Williams, *Quadratic 0–1 programming using the roof dual with computational results*, Tech. Report RUTCOR-8-85, State University of New Yersey, 1985.

Appendix

Sketch of the proof of Theorem 3. With a couple of simple variable transformations it is possible to rewrite system (5) as

$$\begin{aligned} &\max\ d \cdot z \\ &\text{subject to} \\ &\quad z_{ri} + z_{rj} - z_{ij} \le 1, \quad ij \in E^r_+ \\ &-z_{rh} + z_{rk} - z_{ij} \le 0, \quad ij \in E^r_- \text{ and } \begin{cases} h = \min\{i,j\} \\ k = \max\{i,j\} \end{cases} \\ &\qquad\qquad z_{ri} \le 1, \quad i \in V^r \\ &\qquad\qquad z \ge 0, \end{aligned} \tag{12}$$

where d_{ri}, for $i \in V^r$, coincides with d_i as defined in (6), and $d_{ij} := 2|c_{ij}|$, for $ij \in E^r$. Then, it is not difficult to see that the dual of (12) can be interpreted as a network flow problem on a mixed network in which there are undirected edges (associated with constraints corresponding to $ij \in E^r_+$), and directed arcs (associated with constraints corresponding to $ij \in E^r_-$), plus directed arcs from/to the root node r to/from all other nodes $i \in V^r$. A simple example of a complete graph of four nodes is shown in Figure 1, while the corresponding mixed network is given in Figure 2. Specifically, in Figure 2 node labels in boldface and in italics denotes node identifiers and node supply/demand, respectively; edge labels denote edge capacities. Directed arcs from node r to the other nodes have cost 1, as well as the edges of the mixed network, while the directed arcs between nodes $i, j \in V^r$ have cost 0. Finally, arcs from/to node r have infinite capacity.

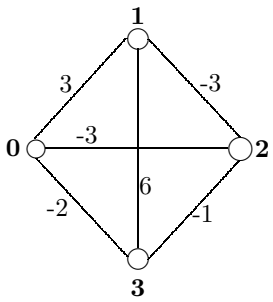

Fig. 1. Weighted undirected graph of four nodes

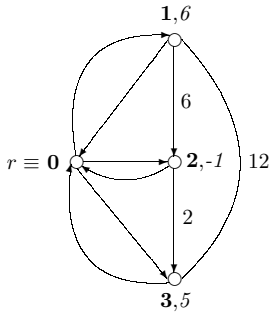

Fig. 2. Network interpretation of the dual of (12)

Obviously, in the special case in which $E^r_+ = \emptyset$, the mixed network becomes a directed network and the dual of (12) is already a min-cost flow problem.

When $E^r_+ \neq \emptyset$, to overcome the difficulty associated with the presence of undirected edges in the network representing the dual of system (12) we make use of a transformation proposed in [10] and recently pointed out in [13]. The transformation works as follows: (a) for each node $i \in V^r$ we make a replica, say i', connected to node r with two arcs ri' and $i'r$; (b) for each arc ij in the mixed network we have an additional arc $j'i'$; and (c) each edge ij in the mixed network is replaced by two arcs $i'j$ and $j'i$. Costs and capacities of the arcs in the directed network are naturally inherited from edges and arcs in the mixed network, while the supply/demand associated with each duplicated node is the opposite of the original one. The directed graph in the special case of the example of figures 1 and 2 is shown in Figure 3.

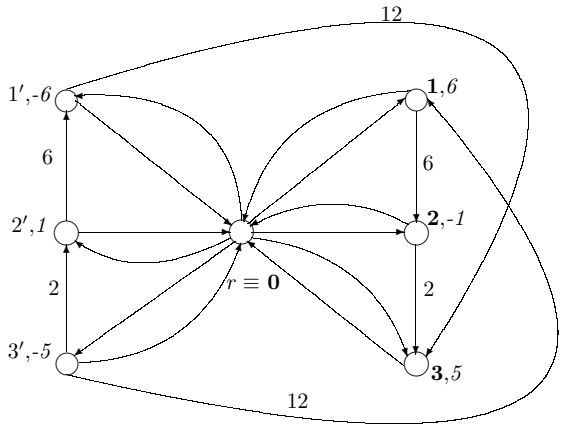

Fig. 3. The directed network corresponding to the mixed network of Figure 2