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

Matteo Fischetti · Domenico Salvagnin · Arrigo Zanette

Minimal Infeasible Subsystems and Benders' cuts

Received: date / Revised Sept. 2009 /Accepted: date

Abstract In this paper we propose a new cut selection criterion for Benders' cuts, based on the correspondence between minimal infeasible subsystems of an infeasible LP and the vertices of the associated alternative polyhedron. The choice of the "most effective" violated Benders' cut corresponds to the selection of a suitable vertex of the alternative polyhedron, hence a clever choice of the dual objective function is crucial—whereas the classic Benders' approach uses a completely random selection policy, at least when feasibility cuts are generated. Computational results are presented, showing that our new cut selection policy can yield a considerable speedup with respect to the standard ones.

We also prove that finding a violated optimality Benders' cut is a strongly NP-hard problem when violated feasibility cuts exist, thus giving a theoretical explanation of the practical difficulty of handling optimality and feasibility cuts together.

Keywords Benders' Decomposition, Cutting Planes, Mixed-Integer Programs, Computational Analysis

1 Introduction

There are situations in mathematical programming where cutting planes can be generated by solving a certain *Cut Generation Linear Program* (CGLP) whose feasible solutions define a family of valid inequalities for the problem

M. Fischetti
DEI, University of Padova E-mail: fisch@dei.unipd.it

D. Salvagnin
DMPA, University of Padova E-mail: dominiqs@gmail.com

A. Zanette
DMPA, University of Padova E-mail: zanettea@math.unipd.it

at hand. Disjunctive cuts [4] and Benders' cuts [5] are two familiar examples arising when solving Mixed-Integer Programs (MIPs)

Benders' cuts were originally proposed as a machinery to convert a generic MIP involving integer variables x and continuous variable y into an integer program involving the x variables only, possibly plus a single continuous variable η taking into account the overall contribution to the objective function of the continuous variables (say $d^T y$). The continuous y variables are removed by a standard projection technique based on dynamic cutting-plane generation. At each iteration, one solves the current *master problem* relaxation in the (x, η) space, and sends the optimal solution (x^*, η^*) to the so-called *slave problem*. This is an LP in the y space that tries to define a suitable vector y^* such that (x^*, y^*) is feasible for the original problem, and $\eta^* = d^T y^*$. If the slave problem is feasible, we are done. Otherwise, a so-called (feasibility or optimality) *Benders' cut* in the (x, η) space is generated by using Farkas' characterization of infeasible LPs, the cut is added to the master problem, and the method is iterated.

The effective use of the cuts is a major issue in any cutting plane method, as three main topics need to be addressed:

i) *When to cut?* Possible answers range from "only when an integer superoptimal solution is available" (as in the original Benders' proposal where cuts are applied only to cut an optimal integer solution of the current master problem) to "whenever a fractional (or integer infeasible) solution is available" (as in modern branch-and-cut frameworks).
ii) *What to cut?* The usual choice in cutting plane methods for integer programming is to cut an optimal *vertex* of an LP relaxation. However this may lead to an unstable behavior and slow convergence, hence internal points to be cut may be preferred as, e.g., in the method recently described in [20]. Alternatively, stabilization through penalty functions may be applied, following a common practice, e.g., in bundle methods.
iii) *How to choose the cut?* Given the point x^* to be separated, choose the "best possible" cut(s) among the violated ones; see [6,8] for recent papers on this topic in the context of disjunctive programming, and also [23] where a weighted combination of the dual solutions is taken before generating the actual Benders' cut.

All three points above play a crucial role in the design of an effective solution method. In the present paper we focus on (iii), and in particular we address the topic of selecting Benders' cuts for general MIPs in an effective way. More specifically, we aim at understanding the properties that qualify a single Benders' cut as "a good one".

The present paper is organized as follows. In Section 2 we briefly review the theory behind the Benders approach. In Section 3 we investigate some fundamental implementation issues. In Section 4 we prove that finding a violated *optimality* Benders' cut is a strongly NP-hard problem (assuming violated feasibility cuts can exist), and propose a new selection criterion for Benders' cuts. Our approach is based on the correspondence between minimal infeasible subsystems of an infeasible LP, and the vertices of the associated *alternative polyhedron*. The choice of the "most effective" violated Benders' cut then corresponds to the selection of a suitable vertex of the alternative

polyhedron, and a clever choice of the dual objective function is of crucial importance. Computational results are presented in Section 5, showing that our new cut selection criterion often allows for a considerable speedup with respect to the standard ones. Some conclusions are finally drawn in Section 6.

An extended abstract of a previous version of the present paper was presented at the *Combinatorial Optimization* meeting held in Oberwolfach, November 2008 [12].

2 The Benders approach

Suppose we are given a generic MIP of the form

$$\begin{aligned} \min\, & c^T x + d^T y \\ & Ax \geq b \\ & Tx + Qy \geq r \\ x \geq 0 \quad & \text{integer} \\ & y \geq 0 \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$, $y \in \mathbb{R}^t$, and matrix Q has m rows. The standard Benders' approach starts by reformulating the MIP above as

$$\begin{aligned} \min\, & c^T x + \eta \\ & Ax \geq b \\ \eta \geq u^T(r - Tx), \quad & u \in U \\ v^T(r - Tx) \leq 0, \quad & v \in V \\ x \geq 0,\ & x \text{ integer} \end{aligned} \tag{2}$$

where the additional continuous variable η takes into account the objective function term $d^T y$, while sets U and V contain the vertices and extreme rays (respectively) of the polyhedron D defined by:

$$\begin{aligned} \pi^T Q &\leq d^T \\ \pi &\geq 0 \end{aligned} \tag{3}$$

Formulation (2) has exponentially many inequalities, so an iterative solution method based on cutting planes is applied. The original Benders' proposal can be outlined as follows.

1. Solve the following *master problem*:

$$\begin{aligned} \min\, & c^T x + \eta \\ & Ax \geq b \\ < & \text{ previously generated Benders' cuts } > \\ & x \geq 0,\ x \text{ integer} \end{aligned} \tag{4}$$

   including (some of) the Benders cuts generated so far (none at the very beginning). By construction, the master problem is a relaxation of the

original MIP reformulation (2), that however keeps the integrality requirement on the x variables. Let (x^*, η^*), with integer x^*, be an optimal solution of the master problem, found, e.g., by an enumerative algorithm.

2. Solve the following *dual slave problem*, that acts as a CGLP:

$$\begin{aligned} \max \pi^T (r - Tx^*) \\ \pi^T Q \leq d^T \\ \pi \geq 0 \end{aligned} \tag{5}$$

3. If the dual slave problem is unbounded, choose any unbounded extreme ray $\overline{\pi}$, add the following *Benders' feasibility cut*

$$\overline{\pi}^T (r - Tx) \leq 0$$

to the master, and repeat from Step 1. Otherwise, let z^* and $\overline{\pi}$ denote the optimal value and an optimal vertex of the dual slave problem, respectively. If $z^* \leq \eta^*$ then stop: the current (x^*, η^*) is feasible and hence optimal for (2). Otherwise, add the following *Benders' optimality cut*

$$\eta \geq \overline{\pi}^T (r - Tx)$$

to the current master problem, and repeat from Step 1.

Modifications of the above scheme lead to variants that are often refereed to, in a wide sense, as "Benders' methods". E.g., by removing the integrality requirement on the x variables in the master, the method leads to a pure cutting plane scheme for solving the LP relaxation of the original MIP (2). In this framework, an optimal (fractional) solution (x^*, η^*) of the LP relaxation of the current master is used, in Step 2, to generate a violated valid inequality to be added to the relaxation—the resulting approach can be significantly faster than solving directly the original LP, mainly in the cases where the dual slave problem decomposes into a series of independent subproblems as, e.g., in scenario-based Stochastic Programming. In addition, the condition to be added to the master need not to be obtained by just solving an LP, but it can exploit an ad-hoc infeasibility analysis [7], possibly based on Constraint Programming [22], or can even be described by logic conditions (instead of linear inequalities) as in the Logic Benders' method pioneered by John Hooker [14, 15].

Coming back to the original Benders' scheme, some observations are in order. First of all, at each iteration MIP (4) needs to be exactly solved, a task that becomes more and more difficult as new cuts are added. Secondly, according to the scheme, at each iteration a single cutting plane is generated and added to the master. Much of the research on the Benders method has been devoted to fix the two issues above. McDaniel and Devine [17] observed that Benders' cuts can be generated also from solutions obtained by solving the linear relaxation of (4), thus providing a set of cutting planes to warm start the method. Analogously, Benders' cuts can be generated from any feasible (but not necessarily optimal) integer master solutions. Following the above observation, Rei, Cordeau, Gendreau, and Patrick [21] recently used a local branching scheme [10] to generate multiple solutions (and hence multiple cuts) for a same master problem.

Example

To illustrate how Benders' cuts can fit into different solution schemes, suppose to apply Benders' method to solve the well-known Asymmetric Traveling Salesman Problem (ATSP). A standard compact MIP formulation involves binary variables x_{ij} associated with the arcs (i, j) of a digraph $G = (V, A)$, and continuous flow variables y^k_{ij} that describe a flow of value 1 from a fixed source node (say node 1) to sink node k, for all $k \in V \setminus \{1\}$. In this example, system $Ax \geq b$ corresponds to in- and out-degree restrictions on the arc binary variables, whereas system $Tx + Qy \geq r$ is made by $|V| - 1$ independent blocks corresponding to the flow-conservation equations for each k, plus the coupling constraints $y^k_{ij} \leq x_{ij}$ for all $k \in V \setminus \{1\}$ and $(i, j) \in A$. In this example, the slave problem decomposes nicely into $n-1$ independent flow problems. Benders' cuts are of the feasibility type only, and correspond to the classical Subtour Elimination Constraints (SECs) in their cut form $\sum_{(i,j)\in\delta^+(S)} x_{ij} \geq 1$ with $1 \in S \subset V$. These cuts are known to be facet-defining for the convex hull of the integer points in the x space (assuming G is a complete digraph). Now, there are two search schemes that can be seen as different implementations of the Benders idea.

(a) [the original Benders' method] keep the integrality requirement on the binary arc variables in the master, and solve it to proven optimality by using an external MIP (actually, ILP) solver: if the optimal solution x^* corresponds to a single circuit, then it gives an optimal ATSP solution; otherwise, find a Benders' cut (SEC) to break one of the subtours associated to x^*, add it to the master, and repeat.
(b) [a modern branch-and-cut method] relax the integrality requirement on the x variables, and solve the current master by using an LP solver: if the optimal (possibly fractional) solution x^* violates any SEC, then add a violated SEC to the master LP problem and repeat. If no violated SEC exists and x^* is integer, then we have a provable optimal ATSP solution; otherwise, the only missing requirement for x^* is integrality, that can be enforced by branching.

Note that in this specific example, scheme (b) above is typically more effective than (a), but there are cases where the opposite holds.

Of course, hybrid search techniques can also be designed, e.g., by implementing a scheme akin to (a) where the ILP solver is not viewed as a black-box but generates violated SECs (if any) on the fly, each time the incumbent integer solution is updated (as suggested by Miliotis [18,19]). Or by using a truncated version of scheme (b) as a preprocessing tool for scheme (a), so as to start with a significant set of SECs generated in a computationally cheap way (very much in the spirit of the general proposal of McDaniel and Devine [17]). □

In the example, both Benders' methods (a) and (b) lack a sensible cut selection criterion: once a violated SEC exists the dual slave becomes unbounded and *any* violated SEC can be returned by the separation procedure—

whereas we know that SEC density (among other characteristics) plays a crucial role in speeding-up the overall convergence.

The choice of good *optimality* Benders' cuts was addressed by Magnanti and Wong in [16], who proposed to accelerate the convergence of the method by generating *Pareto-optimal* cuts. Their method only works with optimality cuts, namely $\eta \geq \overline{\pi}^T(r - Tx)$, and requires the availability of a *core-point* q in the relative interior of the convex hull of the feasible solutions of (2). Given the master point x^* to be separated, one first looks for a Benders' cut that is maximally violated by x^* by solving the standard CGLP (5). If alternative optimal solutions of the dual slave exist, the actual cut to be generated is the one with minimum slack with respect to q. It is easy to see that this policy is equivalent to finding a maximally-violated cut with respect to a new "slightly interior" point where x^* is replaced by $(1-\epsilon)x^* + \epsilon q$ for a very small $\epsilon > 0$. Besides the nice theoretical properties described in [16], this change in the point to be cut has the important effect of avoiding dealing with zero components. Indeed, these components do not contribute to cut violation and hence can lead to uncontrolled cut coefficients in the generated cut—a similar beneficial effect could be obtained by slightly increasing the zero entries in x^*, a simple yet effective trick often used for the separation of other classes of MIP cuts.

In practice, the Magnanti and Wong procedure has some potential drawbacks:

- The procedure requires the dual slave to have a bounded optimal value, hence it cannot be applied in a completely general context involving feasibility cuts.
- The user has to provide a point in the relative interior of the master feasible set. This may be a simple task if the master has a special structure, as in the cases addressed by Magnanti and Wong in their study, but it is NP-hard in general if the problem is a MIP, because we need a point in the relative interior of the convex hull of the integer feasible points.
- The method may be computationally heavy because it requires to solve two LPs to generate a single cut, the second LP being often quite time-consuming due to the presence of an additional equation that fixes the degree of violation of the cut in (x^*, η^*)—for fractional x^*, this equation may be quite dense and numerically unstable. A possible remedy is to use a technique recently exploited by Zanette, Fischetti and Balas [24] in the context of lexicographic optimization, where the second separation LP does not contain an explicit constraint to fix the violation in x^*, but just fixes to zero all the nonbasic variables (including slacks) that have a nonzero reduced cost in the first separation LP.
- The criterion benefits from the existence of several equivalent optimal solutions of the dual slave problem (i.e., several maximally-violated optimality cuts), which is unlikely to be the case when fractional (as opposed to integer) points of the master have to be cut.

3 Implementing the Benders scheme

Any implementation of the Benders approach has to face a number of implementation issues that heavily affect the overall performance of the method. Though major modifications of the big picture are usually well documented [16,17,21], there are apparently minor implementation choices that may have a large impact on the practical performance of the method.

A first naive approach is to implement Benders' separation as described in Section 2, i.e., by maximizing the violation of the given point x^*. Although straightforward, this implementation has serious drawbacks:

- As already mentioned, there is no criterion at all for the choice of the unbounded ray used to generate a violated feasibility cut. This means picking violated feasibility cuts in a very blind way, which is a serious issue on some classes of problems where feasibility cuts play an important role.
- As long as a violated feasibility cut exists, the dual slave is unbounded and no optimality cut can be generated, though these cuts may be essential for improving the lower bound and one would like to generate them as early as possible.

A simple way to mitigate the last drawback was proposed by Benders himself in his seminal paper [5]: if we solve the dual slave with the primal simplex method, when we discover an unbounded ray we are "sitting on a vertex" of polyhedron D, hence we can generate also the optimality cut corresponding to this dual vertex, with no additional computational effort. Note however that the optimality cut produced is by no means guaranteed to be violated, and in any case its choice is quite random as the corresponding vertex is not necessarily a one maximizing a certain quality index such as cut violation, depth, etc. Nevertheless, the effect of the additional optimality cut can be quite substantial on some cases, as we verified in our computational tests.

Since feasibility cuts seem to be the Achilles' heel of the method, a solution would be to truncate polyhedron D into a polytope $D(M) := \{u \in D : \sum_i \pi_i \leq M\}$ for a sufficiently large $M > 0$, so as to work with optimality cuts only. This corresponds to the addition of an artificial continuous variable z with a large cost M in the primal slave, that becomes $\min\{d^T y + Mz : Qy + ez \geq r - Tx^*,\ y, z \geq 0\}$, where $e = (1, \cdots, 1)$, and the modified slave reads:

$$\begin{aligned} \max \pi^T (r - Tx^*) \\ \pi^T Q \leq d^T \\ \pi^T e \leq M \\ \pi \geq 0 \end{aligned} \tag{6}$$

Working with the modified formulation one can therefore apply the Magnanti-Wong procedure, hence the choice of the violated cut is no longer arbitrary. However, a number of potential drawbacks arise:

- A big-M coefficient has been introduced. This is quite tricky and potentially dangerous for the correctness of the method, not to mention the associated numerical issues.
- Feasibility cuts are replaced by "surrogate" optimality cuts that are however inherently weaker, because they depend on M and involve the continuous variable η, with the risk of losing a possible combinatorial structure of their feasibility counterpart.

The issues above were also addressed in the original Benders' paper, where a variant of the separation LP that works directly with the primal slave is described. This variant can be rephrased in the dual space as follows:

- Dynamically update the big M coefficient whenever there is evidence that it is too small. Such an evidence can readily be found by standard LP sensitivity analysis; in particular, the current value for M is not too small if the current basis remains optimal when $M \to +\infty$.
- Decompose the optimal vertex $u(M)$ of $D(M)$ into $u(M) = u + \lambda v$, where $u \in U$, $v \in V$ and $\lambda \geq 0$, so as to generate both an optimality and a feasibility cut for the original problem before truncation. This is always possible due to the tight correlation between D and $D(M)$, and requires at most one additional pivot operation; see [5] for details.

4 Benders' cuts and Minimal Infeasible Subsystems

An important aspect for the practical effectiveness of Benders' cuts is the relative contribution of optimality and feasibility cuts. Indeed, according to our computational experience, feasibility and optimality cuts behave quite differently for two important respects:

- For many problems where term $d^T y$ gives a significant contribution to the overall optimal value, optimality cuts can be much more effective in improving the bound than feasibility cuts, because they involve the η variable explicitly.
- Optimality cuts are typically quite bad from a numerical point view because they tend to exhibit a higher density and *dynamism* (ratio between the maximum and minimum absolute value of the cut coefficients) with respect to the feasibility ones.

In this section we propose a new selection criterion for Benders' cuts that leads to a more clever choice of the separated cuts, in particular when *both* feasibility and optimality violated cuts exist. To emphasize the intrinsic difficulty of handling violated optimality and feasibility cuts together, we observe that finding a violated optimality cut in presence of violated feasibility cuts is equivalent to finding an optimal vertex of a polyhedron with unbounded rays. This is in fact a strongly NP-hard problem, in that it contains the problem of finding a min-cost path in a digraph with negative cycles as a special case. The above considerations can be formalized into to the following (somehow surprising) result:

Theorem 1 *Finding a violated* optimality *Benders' cut for a given point* (x^*, η^*) *is a strongly NP-hard problem, even in case* Q *is totally unimodular,* $d \in \{0, -1\}^t$*,* $x^* = \mathbf{1}$*, and* η^* *is a nonnegative integer.*

Proof We will use a reduction from the following well-known strongly NP-complete *Hamiltonian Path* (HP) problem: Does a given digraph $G = (V, A)$ contain a Hamiltonian directed path from node 1 to node $n := |V|$? To answer this question, we define the following polyhedron

$$D := \left\{ \pi \in \mathbb{R}^A_+ : \sum_{j:(h,j)\in A} \pi_{hj} - \sum_{i:(i,h)\in A} \pi_{ih} \leq d_h \ \forall h \in V \setminus \{1\} \right\} \tag{7}$$

where $d_2 = \cdots = d_{n-1} = 0$ and $d_n := -1$. By construction, D coincides with the feasible solution set of the CGLP (5) when Q^T is the node-arc incidence matrix of the subgraph of G induced by $V \setminus \{1\}$—hence Q is totally unimodular. It is well known the vertices of D are in 1-1 correspondence with the characteristic vectors of the directed *simple* paths in G from 1 to n, while the extreme rays of D correspond to the directed simple cycles of G. Our order of business is therefore to find a vertex of D (i.e., a simple $(1-n)$-path) whose support has maximum cardinality, and check whether this cardinality is strictly greater than $n-2$. To this end, it is enough to consider the CGLP (5) where $r := 0$, $T := -I$, and $x^*_{ij} := 1$ for all $(i, j) \in A$ (so the objective function reads $\sum_{(i,j)\in A} \pi_{ij}$), while setting $\eta^* := n - 2$. By construction, the existence of the required Hamiltonian path is equivalent to the existence of a violated optimality Benders' cut, which completes the proof. □

Our order of business is to define a sound unified framework for the separation of feasibility and optimality cuts. To this end, we observe that Benders' separation can always be rephrased as a pure feasibility problem: given a master solution (x^*, η^*), a violated cut can be generated if and only if the following (extended) primal slave LP is infeasible:

$$\begin{aligned} d^T y &\leq \eta^* \\ Qy &\geq r - Tx^* \\ y &\geq 0 \end{aligned} \tag{8}$$

or equivalently, by LP duality, if the following (homogenized) dual slave problem is unbounded:

$$\begin{aligned} \max \pi^T (r - Tx^*) - \pi_0 \eta^* \\ \pi^T Q \leq \pi_0 d^T \\ \pi, \pi_0 \geq 0 \end{aligned} \tag{9}$$

The cut associated with a given ray $(\overline{\pi}, \overline{\pi}_0)$ of (9) reads

$$\overline{\pi}^T (r - Tx) - \overline{\pi}_0 \eta \leq 0 \tag{10}$$

In practice, one is interested in detecting a "minimal source of infeasibility" of (8), so as to detect a small set of constraints in the slave that suffices

to cut the master solution. According to Gleeson and Ryan [13], the rows of any *Minimal* (with respect to set inclusion) *Infeasible Subsystem* (MIS) of (8) are indexed by the support of the vertices of the following polyhedron, sometimes called the *alternative polyhedron*:

$$\begin{aligned} \pi^T Q &\leq \pi_0 d^T \\ \pi^T (r - Tx^*) - \pi_0 \eta^* &= 1 \\ \pi, \pi_0 &\geq 0 \end{aligned} \tag{11}$$

where the unbounded objective function—namely, the cut violation to be maximized—has been fixed to a normalization positive value.

By choosing a generic objective function

$$\min \sum_{i=1}^{m} w_i \pi_i + w_0 \pi_0 \tag{12}$$

with appropriate nonnegative weights w_i's, it is therefore possible to optimize over the alternative polyhedron so as to select a violated cut corresponding to a MIS of (8) with certain useful properties. Of course, the choice of the objective function coefficients w_i's is of crucial importance, in that it models the quality measure that one wants to apply for a clever cut selection.

It is worth noting that the original Benders' CGLP (6) arises as a particular case by setting $w_0 = 1$ and $w_1 = w_2 = \cdots = w_m = 0$, i.e., by minimizing variable π_0 only. If the optimal solution $\overline{\pi}$ has $\overline{\pi}_0 = 0$, then (10) gives a proof of infeasibility not involving variable η, thus a feasibility cut is generated. Otherwise, $\overline{\pi}_0 > 0$ and it is not difficult to see that $\overline{\pi}/\overline{\pi}_0$ gives an optimal solution of the standard dual slave (5), hence the same maximally-violated optimality cut as in the original method is generated (barring degeneracy).

In a previous version of the present paper [12], the objective function weights $w_0, \cdots, w_m$ were all set to 1, with the aim of trying to reduce the cardinality of the support of the optimal vertex, and hence to heuristically find a minimum-cardinality MIS (which is an NP-hard problem in general; see, e.g., Amaldi, Pfetsch, and Trotter [1]). By analyzing some specific pathological instances we later observed that matrix T often has null rows, meaning that there are "static" conditions in the slave that are always active and do not depend on x. For these rows, there is no reason to penalize the corresponding multiplier π_i in the CGLP objective function, so one should set $w_i = 0$. According to our computational experience, this simple change leads to a quite substantial improvement of the generated cuts in many cases, so our final choice is to consider the following CGLP objective function

$$\min \sum_{i \in I(T)} \pi_i + w_0 \pi_0 \tag{13}$$

where w_0 is a scaling factor taking into account the possibly wider range of variable η, and $I(T)$ contains the indices of the nonzero rows in T.

A final comment is in order. As we are only interested in solutions with a positive cut violation, and because $\{(\pi, \pi_0) \geq 0 : \pi^T Q \leq \pi_0 d^T\}$ is a cone, one can swap the role of the objective function (12) and of the normalization

condition in (11). The swap yields the following equivalent CGLP, akin to the one used for disjunctive cuts by Balas, Ceria, and Cornuéjols [4] (see also Fischetti, Lodi and Tramontani [11]):

$$\begin{aligned} \max \pi^T (r - Tx^*) - \pi_0 \eta^* \\ \pi^T Q \le \pi_0 d^T \\ \sum_{i=1}^{m} w_i \pi_i + w_0 \pi_0 = 1 \\ \pi, \pi_0 \ge 0 \end{aligned} \tag{14}$$

Formulation (14) can be preferable from a computational point of view because normalization constraint $\sum_{i=1}^{m} w_i \pi_i + w_0 \pi_0 = 1$, though very dense, may be numerically more stable than its "cut violation" counterpart $\pi^T (r - Tx^*) - \pi_0 \eta^* = 1$, in particular when x^* is fractional.

At first glance, our CGLP (14) looks similar to the modified slave (6), as both of them make use of a certain normalization condition. In fact, for feasibility cuts ($\pi_0 = 0$), normalization $\sum_i \pi_i \le M$ in (6) can trivially be rewritten as $\sum_i \pi_i = 1$, which is a different (often weaker) version of the one we propose, namely (13). When both optimality and feasibility violated cuts exist, however, the two methods can behave quite differently, as shown in the next section.

5 Computational experiments

The main Benders' variants addressed in the present paper have been implemented in C++. IBM ILOG Cplex 11.2 was used as LP/MIP solver, with all parameters left at their default values.

All tests have been performed on an PC Intel Core2 Q6600 (2.40 GHz) with 4GB of RAM, with a time limit of 3,600 seconds for each run.

To speedup the solution of the several master MIPs generated by the method, as well as to generate multiple cuts that hopefully reduce the overall number of master MIPs to be solved, we implemented the following simple strategy. At each main iteration (i.e., master MIP solution), when the master incumbent solution is updated we generate, on the fly, a corresponding violated Benders' cut (if any). However, this cut is not added to the master problem until the next main iteration, and the master processing continues until the current incumbent solution violates a Benders' cut and it was not updated in the last $K = 1,000$ enumeration nodes. In this way we avoid wasting computing time on proving the optimality of an integer solution that is already known to be infeasible, and at the same time we avoid to restart the master too many times—namely, as soon as a new incumbent solution is found. In this way, several Benders' cuts are typically generated at each main iteration, and our scheme can be seen as a "light" version of the local-branching one proposed in [21].

In our computational study we compared four different MIP solution methods:

`cpx` a state-of-the-art Branch&Cut solver (IBM ILOG Cplex 11.2);
`std` a standard Benders' approach based on CGLP (5);
`std2` a more elaborated Benders' approach using the modified CGLP (6), plus the enhancements described at the end of Section 3;
`mis` our Benders' approach with the new CGLP(14) and normalization condition as in (13), i.e., $w_i = 0$ if i indexes a null row of matrix T, $w_i = 1$ otherwise, and $w_0 = 1$.

Both `std` and `std2` (but not `mis`) implement the Magnanti-Wong [16] acceleration procedure with the nonbasic variable fixing described in [24]. The warm-start procedure of McDaniel and Devine [17] was instead deactivated for all methods because, for the instances in our testbed, it resulted into a generalized performance degradation. Note that each call to the CGLP in `std` and `std2` can generate *two* distinct violated cuts (associated with a vertex and an unbounded ray of the original slave, respectively), whereas it generates (at most) one single cut for `mis`. It is also worth stressing that the three Benders' implementations above are completely general purpose, and only differ in the way the cuts are selected—all other features are identical.

The four MIP methods have been compared on two sets of MIP instances whose structure is known to be well suited for Benders' methods. All instances are available, on request, from the second author.

Our first testbed consists of instances of the *multicommodity-flow network design problem* [3], where one has to allocate capacity to the arcs of a given network by ensuring that all commodities can simultaneously be routed from source to destination. Along with a capacity plan, a routing of all commodities has to be determined. The objective is to minimize the cost of the installed capacity in the network and of the routing of the commodities. More specifically, given a directed graph $G = (V, A)$, a set of commodities K (each commodity being described by a source node s^k, a destination node t^k, a demand d^k, and arc routing costs c^k_{ij}), a base capacity unit C, and arc capacity installation costs f_{ij}, our network loading problem can be formulated as:

$$\min \sum_{(i,j)\in A} f_{ij} y_{ij} + \sum_{k\in K} \sum_{(i,j)\in A} c^k_{ij} x^k_{ij}$$

$$\sum_{j\in V} x^k_{ij} - \sum_{j\in V} x^k_{ji} = \begin{cases} 1 & i = s^k \\ -1 & i = t^k \\ 0 & \text{otherwise} \end{cases} \qquad k \in K, i \in V \tag{15}$$

$$\sum_k d^k x^k_{ij} \le C y_{ij}, \qquad (i,j) \in A \tag{16}$$

$$x^k_{ij} \ge 0, y_{ij} \in \mathbb{Z}^+_0,\ k \in K, (i,j) \in A \tag{17}$$

Network design problems, as well as many other network design problems, are well suited for a Benders' approach because there is natural partition between first-stage integer variables (arc capacities to setup) and second-stage continuous variables (network flows). The reader is referred to Costa [9] for a recent survey on Benders' algorithms successfully applied to network design problems.

We generated two different types of random instances of the above network design problem, the underlying network topology being defined as follows:

`grid`: we first generated 5x5 and 5x6 grid networks, and then we randomly deleted arcs with probability 0.1
`random`: we first generated random networks with 20 and 25 nodes, each node being connected with its 5 nearest nodes, and then we randomly deleted arcs with probability 0.1.

We then generated a demand for some pairs of nodes picked with 0.15 probability, with value taken uniformly in interval $(5, 10)$. Each unit of capacity is of size 20 and has a cost of 5. Finally, we considered two different scenarios: in the `feas` case, routing costs were set to zero and only feasibility cuts were generated by the Benders' algorithms, while in the `opt` case each unit of flow has a cost of 1 on each arc, hence both feasibility and optimality Benders' cuts are generated.

Table 1 summarizes the main characteristics of the instances in our network design testbed.

Topology	Nodes	Type	# constr.s	# var.s	# int.var.s
grid	5x5	feas	2,318	6,281	34
grid	5x5	opt	2,236	5,974	34
grid	5x6	feas	4,282	11,622	41
grid	5x6	opt	4,014	10,527	40
random	20	feas	1,148	5,125	46
random	20	opt	1,192	4,993	43
random	25	feas	2,432	10,405	54
random	25	opt	2,208	9,754	56

Table 1 Testbed characteristics (network design problem); 5 instances for each class

Table 2 reports the outcome of our experiments on the network design testbed; results for `cpx` are not reported because this method exceeded the 3,600-seconds time limit in 36 (out of 40) runs. The table gives average results (geometric means) over the 5 instances of each class for both the `feas` and `opt` scenarios. For each method the table reports the computing time (in CPU seconds: column *time*), the percentage of computing time spent in the CGLP (*sep*), the number of main iterations (i.e., of master MIPs generated: column *iter.s*), and the total number of feasibility (*feas*) and optimality (*opt*) cuts generated. In the time column, the number of time-limit instances (if any) is given in parenthesis; these instances contribute to the overall statistics by taking the current figures (time, iterations, etc.) when the run was aborted.

According to Table 2, all methods spent a large percentage of their computing time in the CGLP, which is not surprising because of the large number of continuous variables; in addition, the master MIPs were solved quickly because they involve a small number of integer variables.

type	class	method	time (s)	sep (%)	iter.s	# feas	# opt
feas	g_5_5	std	22.8	92.9%	125	249	0
		std2	585.2	96.2%	266	592	0
		mis	14.9	98.1%	75	106	0
	g_5_6	std	100.0	98.4%	138	250	0
		std2	$2,303.4^{(3)}$	97.3%	290	701	0
		mis	57.7	98.6%	102	161	0
	r_20_5	std	52.8	90.2%	129	270	0
		std2	$1,095.4^{(1)}$	59.0%	436	955	0
		mis	35.5	93.3%	92	166	0
	r_25_5	std	$313.8^{(1)}$	90.8%	166	426	0
		std2	$2,897.4^{(3)}$	87.4%	344	953	0
		mis	147.7	77.9%	92	182	0
	all	std	$78.4^{(1)}$	93.0%	138	291	0
		std2	$1,438.2^{(7)}$	83.4%	328	784	0
		mis	46.1	91.6%	90	151	0
opt	g_5_5	std	67.9	24.5%	162	525	528
		std2	275.2	64.9%	152	585	589
		mis	18.7	98.4%	77	104	7
	g_5_6	std	177.1	32.7%	182	790	794
		std2	1,439.9	78.3%	295	1,026	1,036
		mis	69.9	99.1%	97	136	21
	r_20_5	std	498.5	9.8%	181	594	812
		std2	$1,554.3^{(1)}$	19.0%	230	765	1,127
		mis	109.0	58.7%	127	124	302
	r_25_5	std	$3,600.5^{(5)}$	12.7%	311	1,315	1,541
		std2	$3,600.6^{(5)}$	43.6%	220	1,339	1,380
		mis	$1,440.2^{(2)}$	64.3%	227	225	893
	all	std	$383.2^{(5)}$	17.8%	202	754	851
		std2	$1,220.3^{(6)}$	45.3%	218	886	987
		mis	$119.7^{(2)}$	77.9%	121	141	80

Table 2 Network design results.

A surprising outcome is that the "more elaborated" standard Benders' implementation (`std2`) is much worse than the "simple" one (`std`) in terms of both computing time and number of main iterations required. The difference is striking for the `feas` scenario, where `std2` requires on average about twice more iterations and cuts than `std`, and takes about 20 times more computing time. Evidently, the trivial normalization condition $\sum_i \pi_i = 1$ implicitly used by `std2` actually hurts, as it turns out to perform even worse than the random choice of the unbounded ray performed by `std`—thus confirming that a clever choice of normalization is a key issue in practice. As to `mis`, it outperforms `std` by a factor of about 2 in the `feas` scenario, and of about 3 in the `opt` one. The fact that the speedup is due to a more effective choice of the cuts

is confirmed by the greatly reduced number of cuts (and of main iterations) required.

We also tested the algorithms on 30 *network expansion* hard instances from the literature [2]—the easiest instances have been removed from the testbed. For these instances only feasibility cuts can be generated, hence the three Benders' implementations only differ for the normalization used when solving the slave problem.

Table 3 reports the main characteristics of each class of instances in this second testbed, whereas Table 4 gives the corresponding average results; again, `cpx` is not reported because it exceeded the time limit in 24 (out of 30) runs.

Class	# instances	# constr.s	# var.s	# int.var.s
100.20.2	5	2080	2970	1980
100.20.4	5	2080	4950	3960
100.20.8	5	2080	8910	7920
150.20.2	5	4620	6705	4470
150.20.4	5	4620	11175	8940
150.20.8	5	4620	20115	17880

Table 3 Testbed characteristics (hard network expansion instances from [2])

class	method	time (s)	sep (%)	iter.s	# feas
100.20.2	std	205.6	0.5%	195	387
	std2	19.8	5.1%	136	218
	mis	22.1	5.3%	139	214
100.20.4	std	441.7	0.2%	140	392
	std2	65.3	1.5%	122	270
	mis	67.3	2.0%	121	266
100.20.8	std	422.7	0.2%	139	409
	std2	63.9	1.7%	137	321
	mis	60.5	2.4%	136	338
150.20.2	std	1,790.6	0.1%	291	659
	std2	105.1	2.9%	231	346
	mis	108.0	5.6%	241	357
150.20.4	std	$3,194.6^{(1)}$	0.1%	169	672
	std2	$1,430.3^{(1)}$	0.3%	207	541
	mis	$1,086.7^{(1)}$	0.9%	213	542
150.20.8	std	$3,568.5^{(4)}$	0.0%	167	580
	std2	1,492.3	0.4%	204	623
	mis	$1,537.0^{(1)}$	0.6%	202	617

Table 4 Network expansion results

As far as the two standard Benders' implementations are concerned, this second testbed behaves just the opposite way as the previous one: separation time is almost negligible, and `std2` is by far faster than `std`—hence its normalization condition helps in separating the "right" cuts. In this setting, the performance of `mis` is very similar to that of `std2`, which is an indication that the different normalizations they use are equally effective for this problem class.

On the whole, the results show that our new cut selection criterion in more robust than those from the literature, in that `mis` is almost always the best (possibly with ties) of the three methods under comparison, while `std` and `std2` exhibit a much more erratic behavior that heavily depends on the structure of the underlying problem. As expected, `mis` obtains its best speedup when both optimality and feasibility cuts are be separated, due to the fact that these cuts are treated in a sound unified framework.

6 Conclusions

By exploiting the correspondence between minimal infeasible subsystems of an infeasible LP and the vertices of the associated alternative polyhedron, we have been able to define a simple yet effective cut selection criterion for Benders' cuts. As a by product we have obtained a novel interpretation of a widely-used disjunctive cut normalization, based on combinatorial objects (minimal infeasible subsystems), thus providing a theoretical explanation of its practical effectiveness.

Computational results show that the proposed method may allow for a substantial speedup with respect to the standard one, mainly when feasibility together with optimality cuts are generated.

From a theoretical point of view, we proved a surprising (at first glance) complexity result, namely that finding a violated Benders' cut is a strongly NP-hard problem if one restricts to optimality cuts only.

Acknowledgements This work was supported by the Future and Emerging Technologies unit of the EC (IST priority), under contract no. FP6-021235-2 (project "ARRIVAL") and by MiUR, Italy (PRIN 2006 project "Models and algorithms for robust network optimization"). Thanks are due to two anonymous referees for their constructive comments, and to Laurence Wolsey for discussions about the complexity of separating optimality Benders' cuts.

References

1. Amaldi, E., Pfetsch, M.E., Jr, L.T.: On the maximum feasible subsystem problem, IISs and IIS-hypergraphs. Mathematical Programming **95**(3), 533–554 (2003)
2. Atamtürk, A., , Nemhauser, G.L., Savelsbergh, M.W.P.: Valid inequalities for problems with additive variable upper bounds. Mathematical Programming **91**, 145–162 (2001)
3. Atamturk, A., Rajan, D.: On splittable and unsplittable flow capacitated network design arc-set polyhedra. Mathematical Programming **92**, 315–333 (2002)

4. Balas, E., Ceria, S., Cornuéjols, G.: Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. Management Science **42**, 1229–1246 (1996)
5. Benders, J.: Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik **4**, 238–252 (1962)
6. Cadoux, F.: Computing deep facet-defining disjunctive cuts for mixed-integer programming. Mathematical Programming **122**(2), 197–223 (2009)
7. Codato, G., Fischetti, M.: Combinatorial Benders' cuts for mixed-integer linear programming. Operations Research **54**(4), 758–766 (2006)
8. Cornuèjols, G., Lemaréchal, C.: A convex analysis perspective on disjunctive cuts. Mathematical Programming **106**(3), 567–586 (2006)
9. Costa, A.M.: A survey on Benders decomposition applied to fixed-charge network design problems. Computers & Operations Research **32**(6), 1429–1450 (2005)
10. Fischetti, M., Lodi, A.: Local branching. Mathematical Programming **98**(1–3), 23–47 (2003)
11. Fischetti, M., Lodi, A., Tramontani, A.: On the separation of disjunctive cuts. Mathematical Programming (2009). (to appear)
12. Fischetti, M., Salvagnin, D., Zanette, A.: Minimal infeasible subsystems and benders' cuts. Oberwolfach Report 51/2008 pp. 8–11 (2008)
13. Gleeson, J., Ryan, J.: Identifying minimally infeasible subsystems of inequalities. ORSA Journal on Computing **2**(1), 61–63 (1990)
14. Hooker, J.N.: Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction. Wiley (2000)
15. Hooker, J.N., Ottosson, G.: Logic-based Benders decomposition. Mathematical Programming **96**(1), 33–60 (2003)
16. Magnanti, T., Wong, R.: Accelerating Benders decomposition: algorithmic enhancement and model selection criteria. Operations Research **29**, 464–484 (1981)
17. McDaniel, D., Devine, M.: A modified Benders' partitioning algorithm for Mixed Integer Programming. Management Science **4**, 312–319 (1977)
18. Miliotis, P.: Integer programming approaches to the travelling salesman problem. Mathematical Programming **10**, 367–378 (1976)
19. Miliotis, P.: Using cutting planes to solve the symmetric travelling salesman problem. Mathematical Programming **15**, 177–178 (1978)
20. Naoum-Sawaya, J., Elhedhli, S.: An interior-point branch-and-cut algorithm for mixed integer programs. Tech. rep., Department of Management Sciences, University of Waterloo (2009). Presented at ISMP 2009
21. Rei, W., Cordeau, J.F., Gendreau, M., Soriano, P.: Accelerating Benders Decomposition by Local Branching. INFORMS Journal on Computing (21), 333–345 (2009)
22. Thorsteinsson, E.S.: Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming. Lecture Notes in Computer Science **2239**, 16–30 (2001)
23. Wentges, P.: Accelerating Benders' decomposition for the capacitated facility location problem. Mathematical Methods of Operations Research **44**, 267–290 (1996)
24. Zanette, A., Fischetti, M., Balas, E.: Can pure cutting plane algorithms work? In *IPCO*, volume 5035 of *Lecture Notes in Computer Science*, Springer pp. 416–434 (2008)