Implementing Automatic Benders Decomposition in a Modern MIP Solver

Pierre Bonami[1], Domenico Salvagnin[2], and Andrea Tramontani[3]

[1] CPLEX Optimization, IBM, Madrid, Spain
[2] DEI, Via Gradenigo, 6/B, 35131 Padova, Italy
[3] CPLEX Optimization, IBM, Bologna, Italy

Abstract. We describe the automatic Benders decomposition implemented in the commercial solver IBM CPLEX. We propose several improvements to the state-of-the-art along two lines: making a numerically robust method able to deal with the general case and improving the efficiency of the method on models amenable to decomposition. For the former, we deal with: unboundedness, failures in generating cuts and scaling of the artificial variable representing the objective. For the latter, we propose a new technique to handle so-called generalized bound constraints and we use different types of normalization conditions in the Cut Generating LPs. We present computational experiments aimed at assessing the importance of the various enhancements. In particular, on our test bed of models amenable to a decomposition, our implementation is approximately 5 times faster than CPLEX default branch-and-cut. A remarkable result is that, on the same test bed, default branch-and-cut is faster than a Benders decomposition that doesn't implement our improvements.

1 Introduction

Benders decomposition was originally proposed in [5] to solve Mixed Integer Programs (MIP). The decomposition consists in splitting the original problem between a *master problem*, that consists of the integer variables of the original problem and possibly some additional continuous variables, and a *Cut Generating Linear Program (CGLP)* formulated in the space of the remaining variables. As originally stated, Benders method is iterative. At each step, the master is first solved to optimality. The CGLP is then constructed using the solution of master and solved: from its solution cuts for the master are derived (Benders cuts herein). This process is repeated until no cuts are found by the CGLP. The theorem of Benders guarantees that at this point the original problem is solved to optimality.

Benders decomposition is more often applied to MIPs where the CGLP constraint matrix has a block diagonal structure and can be further decomposed into smaller problems. Among the numerous applications the most notable are Facility Location [10,11], Network Design [14] and Stochastic Optimization [6].

In the last decades, a vast body of research has examined every step of Benders decomposition. A recent and comprehensive survey can be found in [23]. Here, we overview the literature most relevant to our work. First, a feasible solution of the master problem is not needed to generate a Benders cut. An initial good set of cuts can usually be found by solving the initial LP relaxation of the problem by Benders decomposition [20]. More generally, in branch-and-cut, Benders cuts can be separated at any node of the search tree. Second, several computational studies (e.g., [4,10]) have shown that a simple stabilization mechanism for the cutting plane loop allows to significantly improve the effectiveness of the method. Finally, most of the research has been devoted to separating non-dominated or even facet defining Benders cuts. For the case where the CGLP is feasible and bounded, [19] proposes to solve two LPs to guarantee non-domination. In [22], this approach was improved and it was shown that non-domination can be obtained with a single LP. For the infeasible case, in [13] a normalization was proposed based on the concept of *minimal infeasible subsystem*.

For the applications mentioned above, Benders decomposition is often the only method able to solve problems of realistic size. Most implementations in the scientific literature or in the industry are ad-hoc implementations for a specific class of problems. In this paper, we report on the automatic Benders decomposition solver implemented in CPLEX [16]. Our goal is to have a numerically robust implementation that can be used as a black-box on any MIP, and that is competitive on the classes of problems where Benders decomposition has been reported to be useful in practice.

To achieve this goal our main contributions are: applying a generic stabilization procedure to solve the initial Benders cut loop efficiently, dealing with cases of unboundedness, dealing with numerical stability of Benders cuts and artificial variables, handling of linking constraints with special structure to simplify the CGLP, and applying normalizations to find "good" Benders cuts when the

CGLP is infeasible. For this last item in particular, we implement for the first time, to the best of our knowledge, a new normalization proposed in [7]. Our computational results show that the resulting algorithm is considerably faster than default branch-and-cut on models amenable to decomposition, and that the algorithmic enhancements we propose have a dramatic effect.

The outline of the paper is as follows: in Section 2, we outline the overall Benders decomposition algorithm, setting up the required notation. In Section 3, we detail enhancements to the construction of the master problem and overall numerical stability of the procedure. In Section 4, we describe the improvements to solving the CGLP mentioned above. Finally, in Section 5 we computationally evaluate our algorithm and we analyze the effect of the different algorithmic ideas we propose.

2 Benders Decomposition

In this section we briefly outline the Benders decomposition algorithm. Most textbooks use the so-called dual space of the CGLP, which has the advantage of directly expressing coefficients for the Benders cuts. However, as noted in the original paper [5], it is often computationally more convenient to work in the primal space, i.e., the space in which the problem is originally formulated. The implementation in CPLEX is entirely done in the primal space, and we believe it is also a simpler way to view the method. We will use this point of view in the remainder. We consider a MIP of the form:

$$\min cx + dy \tag{1}$$
$$Ax \geq b, \tag{2}$$
$$Tx + Qy \geq r, \tag{3}$$
$$x, y \geq 0 \text{ and } x \in \mathbb{Z}^n, \tag{4}$$

where $c \in \mathbb{Q}^n$, $d \in \mathbb{Q}^p$, $A \in \mathbb{Q}^{m_1 \times n}$, $T \in \mathbb{Q}^{m_2 \times n}$, and $Q \in \mathbb{Q}^{m_2 \times p}$. The decomposition starts by creating the master problem. It involves the x variables and an additional variable η representing the contribution to the objective of the y variables:

$$\min cx + \eta \tag{5}$$
$$Ax \geq b, \tag{6}$$
$$< \text{Benders cuts} >, \tag{7}$$
$$x \in \mathbb{Z}^n_+, \eta \text{ free.} \tag{8}$$

Some of the x variables could be continuous without significantly changing the method. Note that the set of Benders cuts (7) is initially empty. Also, at any point we can assume that (5)–(8) is feasible, otherwise the original problem is proven infeasible and the method is stopped.

Given a solution (x^*, η^*) to the master problem, the CGLP tries to find a feasible y satisfying the linking constraints (3) for fixed $x = x^*$. The following lemma details how the CGLP is defined and how it is used to either derive a Benders cut, or conclude optimality.

Lemma 1 (Benders Theorem [5]). *Let (x^*, η^*) be an optimal solution of* (5)–(8) *and define the CGLP:*

$$\min dy \tag{9}$$

$$Tx^* + Qy \geq r \tag{10}$$

$$y \geq 0. \tag{11}$$

(i) If (9)–(11) *is infeasible, then there exists $\pi \in \mathbb{Q}_+^{m_2}$ such that*

$$\pi Tx \geq \pi r \tag{12}$$

is valid and $\pi Tx^* < \pi r$.

(ii) If (9)–(11) *has a finite optimum y^*, then: Either $dy^* \leq \eta^*$ and (x^*, y^*) is optimal for* (1)–(4)*. Otherwise, there exists $\pi \in \mathbb{Q}_+^{m_2}$ such that*

$$\eta + \pi Tx \geq \pi r \tag{13}$$

is valid and cuts off (x^*, η^*).

Although the result is well known, we give a short proof in the appendix.

The two cuts (12) and (13) defined in Lemma 1 are the so-called Benders cuts. The former is the *feasibility cut* and the latter the *optimality cut*. After the CGLP is solved, if a cut has been derived, it can be added to the master problem, and the method is iterated. Otherwise, (1)–(4) is solved. Note that we have neglected the particular case where the master problem is unbounded. We will deal with it in Section 3.

Before proceeding to our implementation of this procedure, two remarks are in order. First, as noted in the introduction, the CGLP is often itself decomposable (Q is block diagonal). In this case, introducing an η variable for each block, the CGLP can be split into smaller problems and each one may give a cut. Second, the procedure outlined doesn't require solving (5)–(8) to optimality before solving the CGLP. Instead, in a branch-and-cut algorithm, the CGLP can be solved every time an integer feasible solution for (5)–(8) is encountered. Either this solution is cut and the search can proceed or it is not cut and it is indeed also feasible for (1)–(4).

3 The Master Problem

The overall Benders decomposition algorithm implemented in CPLEX follows the algorithm described in the previous section. In this section, we detail in particular how the master problem is constructed. Here is the main workflow of the method:

1. A decomposition of the complete model (1)–(4) is detected. If the user provided a decomposition (via annotations [16]), then CPLEX will decompose the model according to it. Otherwise, it performs an automatic split, in which integer variables are assigned to the master, while continuous variables are assigned to the CGLP. The latter is eventually split into several independent CGLPs if a block decomposition can be detected. This detection is very efficient, as it is linear in the number of nonzeros of the CGLP matrix.
2. The complete model (1)–(4) is presolved using CPLEX regular MIP presolve (only reductions that may invalidate the decomposition previously identified are disabled). This step is applied before actually decomposing the problem because more reductions are typically found on the complete model and presolve is usually cheap enough that it can be done on the full model.
3. The presolved model is decomposed according to the decomposition identified at Step 1.
4. An initial stabilized Benders cut loop is executed on the LP relaxation of the problem. The rationale is to warm start the Benders search with a tighter approximation of the projection of the complete model, so that the subsequent search can benefit from it. We detail the stabilization procedure below.
5. Once the initial Benders cut loop is over, a regular branch-and-cut is started from the current master. This is pretty much a regular MIP solve, but in which Benders cuts are separated on the fly as lazy constraints.

To deal with all challenges in maintaining a numerically stable master problem, several additional ingredients are needed. We detail them in the next paragraphs.

Stabilization of Initial Cut Loop Stabilization is obtained using an in-out strategy [4,10]. Briefly, this consists in not trying to separate the optimal solution of the LP relaxation (x^*, η^*) of (5)–(8), but using instead a suitable convex combination of (x^*, η^*) and a point (x^0, η^0) that is in the relative interior of the projection of the feasible region of (1)–(4) onto the variables (x, η). The point (x^0, η^0) is called the *core point*. The in-out strategy performs a binary search on the line segment joining (x^0, η^0) and (x^*, η^*), until a violated cut is found. In applications, the corepoint is usually obtained by exploiting the specific structure of the problem. In our generic framework, we compute it by solving the LP relaxation of the complete model without the objective and using the barrier algorithm without crossover. This is to attempt to get a point close to the analytic center of the LP relaxation of (1)–(4).

Cut Violation A key decision for the soundness (and numerical stability) of the overall decomposition is the strategy used to decide when a master solution (x^*, η^*) is violated or not by a Benders cut. As CPLEX is based on floating point arithmetic, tolerances are needed. For optimality cuts, the violation of a cut has a natural interpretation: it is the amount by which the artificial variable η^* underestimates the contribution of the CGLP variables to the objective function. As such, for optimality cuts we use the regular optimality tolerance used by the solver. For feasibility cuts, there is no such natural interpretation, and any scaling of feasibility cuts is arbitrary. Therefore, we consider a master solution (x^*, η^*) to be violated if and only if the corresponding CGLP is infeasible (according to the regular feasibility tolerances) regardless of whether we are actually able to derive a sufficiently violated feasibility cut out of it. This is also consistent with what a regular B&C algorithm would have done on the complete model.

Another key aspect is what to do if we fail to derive a cut. When the issue is on the CGLP side, sometimes it can pay off to resolve the CGLP from scratch (possibly forcing a different LP algorithm), and reconstruct the cut. However, in some cases the cut is inherently bad, e.g., when the cut returned by the CGLP is violated but its dynamism is such that it cannot be added to master. In this case, the current master solution is still flagged as invalid, and we act as follows. If the master solution comes from a heuristic, we just discard the solution. If it comes from a node, we have no choice but to branch on a non-fractional variable, unless we already reached a leaf of the enumeration tree. If we are at a leaf of the tree, and there are no continuous variables (except the artificial η) in master, we can still prune the node. However, this would not be correct if there are structural continuous variables in master, so in this latter case we have no choice but to abort with a numerical failure.

Scaling of the Variable η in Optimality Cuts Another aspect where the method may fail for numerical reasons is the scaling of η in optimality cuts. Note that in (13) the coefficient of η is 1 by construction. Depending on the contribution of the y variables to the objective this can pose severe problems in the numerical behavior of the method. If the coefficients πT for the variables x in the cuts are very large or very small, an LP solver using floating point arithmetic might not be able to handle them correctly. In such a case it is possible to scale η to get more numerically stable cuts. In particular, we can define η so that instead of being equal to dy, it is a fraction of it, i.e., $\alpha\eta = dy$. The derivation of a cut is the same as before, except that in the aggregation used to construct the optimality cut we now use the inequality $dy \leq \alpha\eta$ leading to the cut $\alpha\eta + \pi T x \geq \pi r$. Of course all optimality cuts must share the same scaling for η while the coefficients for x may be vastly different among them. Therefore choosing an appropriate value for α is not trivial. In our implementation, we define α to be equal to the largest coefficient for an x variable in the first optimality cut derived. Note that α could be dynamically rescaled in the procedure but our simple attempt did not find any advantage to it.

Cutting a Ray Finally, a case that we left out in Section 2 is the separation of cuts when master is unbounded. In most of the literature this is excluded by construction, but CPLEX has to deal with the general case. Denote with P and R the continuous relaxations of (1)–(4) and (5)–(8), respectively. When R is unbounded, the next lemma shows that a variant of the CGLP can be used to conclude that P is unbounded as well (and hence (1)–(4) is either infeasible or unbounded), or to separate a cut to truncate an unbounded ray of R.

Lemma 2. *Let (u^*, u_0^*) be an unbounded ray of R and consider the modified CGLP*

$$\min ds \tag{14}$$
$$Tu^* + Qs \geq 0 \tag{15}$$
$$s \geq 0. \tag{16}$$

(i) If (14)–(16) is unbounded, then P is unbounded.
(ii) If (14)–(16) is infeasible, then there exists $\pi \in \mathbb{Q}_+^{m_2}$ such that $\pi Tx \geq \pi r$ is valid and $\pi Tu^* < 0$.
(iii) If (14)–(16) has a finite optimum s^*, then: If $ds^* \leq u_0^*$, P is unbounded. Otherwise, $\exists \pi \in \mathbb{Q}_+^{m_2}$ such that $\eta + \pi Tx \geq \pi r$ is valid and $u_0^* + \pi Tu^* < 0$.

The proof is in the appendix. Note that the only differences w.r.t. the case in which we cut a point are (i) fix the master variables to the values in the ray (rather than a point), and (ii) zero out the right hand side of the constraints[4].

4 CGLP Improvements

In this section we describe several improvements to the CGLP: exploiting linking constraints with special structure in Section 4.1, and normalization conditions to separate "good" feasibility cuts in Section 4.2.

4.1 Generalized Bound Constraints

Benders decomposition can be effective when the problem structurally simplifies after fixing variables x. The most common case is when Q is block diagonal. However, the simplification can be significant in other cases as well. A relevant case arises when many linking constraints (3) involve only one y variable. We denote those constraints as Generalized Bound Constraints (GBCs) since in the CGLP, with the x variables fixed, they boil down to simple bound constraints on the y variables. A prime example of GBCs is the one of variable bound constraints like, e.g., constraints of the form $y_j \leq x_i$ that are prevalent in facility location problems. However, other and more complex GBCs, involving several x variables at a time, can also arise in practice, as, for instance, in the case of partial set covering location problems [8].

Translating GBCs to simple bound constraints is key to solving the CGLP faster. However, fully exploiting the presence of GBCs requires some dedicated machinery. Blindly fixing the x variable in the CGLP and having LP presolve do the necessary simplifications potentially destroys the warm-starting capabilities of the simplex method. On the other hand, disabling presolve is of course not an option either as in this case GBCs would not be turned into simple bounds anymore, negating all the benefits of the method. For this reason, we treat GBCs explicitly when we set up the CGLP: we don't add them to the CGLP

[4] This also applies to bounds: bounded y variables turn into directions s fixed to zero.

formulation, but rather compute on the fly the corresponding simple bounds and directly change those. Note that we need to keep track of which GBC (if any) is active for each CGLP variable y_j, as we need to multiply it with the corresponding dual multiplier when computing the cut coefficients.

4.2 CGLP Normalization

It was noted in [13] that the textbook implementation of the Benders CGLP gives little control on which feasibility cut is returned. Actually, any dual feasible solution is optimal and any arbitrary unbounded dual ray will be returned by an LP solver. To select "good" feasibility cuts, we need to add a normalization condition that truncates the dual cone: clearly, the choice of the normalization is critical. In the following, we will describe two such normalizations. Note that, by adding the objective as a constraint, we can always reduce ourselves to the case where the CGLP is infeasible, and treat feasibility and optimality cuts in a unified way. However, this has several drawbacks: the objective is often dense, numerically shaky and badly scaled w.r.t. the other constraints in the model. A preliminary implementation of this unified approach indicated it is not effective. For this reason, we adopt a two-stage approach. We first solve (9)–(11): if it is feasible, an optimality cut is derived. Otherwise, we temporarily remove the objective and add a normalization. Once the cut is obtained, the normalization is removed and the objective restored[5]. It is important to note that the addition of the normalization can be done in both cases without hindering the warm-start capabilities of the simplex method. Finally, we note that GBCs (see Section 4.1) do not simplify to simple bounds if they are used in the normalization. Therefore, we do not apply the normalization to those constraints. As a result cuts are potentially weaker, but separation would be orders of magnitude slower on some models classes otherwise.

L^1 Normalization Assume that (9)–(11) is infeasible. A normalization is simply introduced by adding a penalty variable z_0 as follows:

$$\min z_0 \tag{17}$$
$$Tx^* + Qy + z_0 \geq r \tag{18}$$
$$y, z_0 \geq 0 \tag{19}$$

Note that the addition of z_0 acts as normalization condition in the dual space, specifically the L^1-norm of the dual multipliers is constrained to be 1. This normalization is the one used in [13] but expressed in the primal space. It was originally proposed in the context of lift-and-project cuts in [2] and, as shown in [12], it has nice numerical properties as it favors the separation of cuts with a sparse support.

[5] There is a notable exception to this strategy: if the CGLP has no objective (i.e. $d = 0$) we never remove the normalization, as any violated Benders cut will be a feasibility one by construction.

As we assumed that (9)–(11) is infeasible, the optimal solution of (17)–(19) has $z_0^* > 0$. Using the vector π of optimal dual multipliers we can obtain (12). By strong duality it holds that $z_0^* = \pi(r - Tx^*)$, hence the cut is violated by x^*.

CW Normalization The L^1 normalization is known to have nice numerical properties, but it does not give any theoretical guarantee on the strength of the feasibility cuts. A better approach in this sense is described by Conforti and Wolsey in [7]. Let x^0 be the core point defined in Section 3. The geometric idea is to find the point on the line segment (x^0, x^*) which is feasible for (9)–(11) and further away from x^0. It is shown in [7] that this approach separates facet defining inequalities with probability 1. Defining the convex combination as $x^* + \lambda(x^0 - x^*)$, after introducing the variable λ, we can write the CGLP as:

$$\begin{aligned} &\min \lambda \\ Tx^* + Qy + \lambda[T(x^0 - x^*)] &\geq r \\ y &\geq 0 \\ 0 \leq \lambda &\leq 1 \end{aligned} \tag{20}$$

Note that the dual constraint associated with λ reads $\pi T(x^0 - x^*) = 1$, which is the well known Balas–Perregaard normalization on the polar space [3]. Given the optimal dual multipliers π, a feasibility cut is derived as previously.

Although the CW normalization is theoretically stronger than the L^1 normalization, (20) is typically harder to solve than (17)–(19). Therefore, CPLEX chooses at runtime which normalization to apply, with the rationale of trying to use the CW normalization when feasibility cuts appear to be important w.r.t. optimality cuts.

5 Computational Results

In this section we report on some computational experiments aimed at assessing the importance of the algorithmic components previously described. In particular, we focus on i) in-out techniques to stabilize and accelerate the convergence of the initial Benders cut loop, ii) CW normalization to separate stronger feasibility cuts, and iii) special handling of GBCs in the CGLP.

To this end, we considered a benchmark test bed of instances that are suitable for Benders decomposition. Specifically, we collected 209 two-stage stochastic models from various applications (capacitated facility location [18,6], network interdiction [21,6], fixed charge multi-commodity network design [9], chance-constrained programs [17], and others from CPLEX internal library) and 166 non-stochastic models also coming from different applications (capacitated and uncapacitated facility location [24,11,15,10], network expansion [1], partial set covering location [8], and others from CPLEX internal library), for a total of 375 benchmark instances on which Benders decomposition is expected to be effective. All tests were conducted by running CPLEX 12.10 [16] on a cluster of identical 12 core Intel Xeon CPU E5430 machines running at 2.66 GHz and equipped with 24GB of RAM. A time limit of 10,000 seconds was enforced on each run.

Table 1 compares the default CPLEX branch-and-cut ("B&C") to two versions of CPLEX automatic Benders search: the default method ("Benders default") and the much weaker variant where we disabled in-out, CW normalization[6], and the special handling of GBCs ("Benders no all"). The table reports aggregated results on all 375 instances which are grouped in each row based on the *hardness of the models*. First, the set "all" consists of all the models for which no method had a failure and all methods gave consistent objective values. Then, the set "all" is subdivided in classes "[n, 10k]" ($n \in \{0, 1, 10, 100\}$), containing the models for which at least one of the methods took at least n seconds and that were solved to optimality within the time limit by at least one. For each set, we report: the number of models ("#models"), the number of time limit hit by each method ("#tilim"), then for the two methods "Benders default" and "Benders no all", the ratio of the shifted geometric means with respect to the reference "B&C" for solution times ("time") and number of nodes to optimality ("nodes")[7] (a value $t < 1$ indicates that the specific method is faster than the reference one by a factor of $1/t$).

Table 1. Comparison between regular B&C and Benders decomposition.

		B&C	Benders default			Benders no all		
class	#models	#tilim	#tilim	time	nodes	#tilim	time	nodes
all	361	165	72	0.23	58.6	179	1.44	149.
[0,10k]	313	117	24	0.19	44.8	131	1.53	162.
[1,10k]	310	117	24	0.18	45.0	131	1.53	166.
[10,10k]	304	117	24	0.18	47.8	131	1.51	160.
[100,10k]	285	117	24	0.16	55.5	131	1.55	192.

The results reported in Table 1 clearly show that, on a test bed of instances amenable to Benders decomposition, the default variant of CPLEX Benders significantly outperforms regular branch-and-cut. In particular, considering the instances in the class [0,10k], the number of timeouts is reduced from 117 to 24 and the Benders solver is around 5.26 times faster. However, the table also shows that advanced algorithmic components are crucial to achieve good performance. Indeed, by disabling in-out techniques, the CW normalization and the special handling of GBCs, performance dramatically deteriorates. In particular, still considering the instances in [0,10k], the number of timeouts increases to 131 and the Benders solver becomes 1.53 times slower than regular branch-and-cut.

[6] Note that, when the CW normalization is disabled, the L^1 normalization is used instead, and thus feasibility cuts are still separated using some normalization.

[7] The shift applied is of 1 second for "time" and 10 nodes for "nodes".

In order to better assess the performance impact of the individual algorithmic components highlighted in Table 1, we conducted a set of experiments in which we disabled each of them individually. The outcome of these experiments is summarized in Table 2. Each row compares a variant of the CPLEX Benders solver, obtained by disabling one or more features, against the default Benders solver. For each comparison we report only the results for the instances in the class [0,10k]. We remark that each row is independent of the others and the number of models varies a little. The structure of the table is similar to Table 1. We add three columns under the header "affected" to report results only on the models on which the specific solver in the comparison is at least 10% slower or faster than the reference solver (i.e., default Benders decomposition).

Table 2. Impact of the individual features in the Benders solver on the [0,10k] bracket.

		default	all models			affected		
feature	#models	#tilim	#tilim	time	nodes	#models	time	nodes
No InOut	298	1	41	1.31	0.99	243	1.40	0.96
No CW-norm	301	0	0	1.14	1.40	71	1.83	4.36
No InOut and CW-norm	297	0	76	3.35	3.52	266	3.85	4.06
No GBCs	305	3	20	2.31	0.83	214	3.29	0.79
No all	300	6	113	9.03	3.87	275	11.00	4.45

The results reported in Table 2 lead to the following observations:

1. In-out appears to be the most important feature, as it affects 82% of the models and disabling it leads to 40 additional timeouts.
2. The CW normalization seems to be less important than in-out, as it affects only 24% of the models and no timeouts are introduced by disabling it. However, by comparing "No InOut" and "No InOut and CW-norm", we can clearly see that CW normalization becomes fundamental if in-out is disabled. Intuitively, the two techniques are related as they both use the segment joining x^* to x^0 to separate deeper cuts. In this sense, the CW normalization is theoretically superior, but our experiments show that in-out is also essential.
3. Handling of GBCs affects 70% of the models and allows to solve 17 additional instances. Also, it appears to be the most important single feature in terms of overall improvement of computing time.

Acknowledgements

We thank Daniel Junglas and Roland Wunderling for many discussions and helping out with the implementation of some of the ideas. We also thank Michele Conforti for the many discussions on normalizations in Benders CGLPs.

References

1. Atamtürk, A., , Nemhauser, G.L., Savelsbergh, M.W.P.: Valid inequalities for problems with additive variable upper bounds. Mathematical Programming **91**, 145–162 (2001)
2. Balas, E.: A modified lift-and-project procedure. Mathematical Programming **79**, 19–31 (1997)
3. Balas, E., Perregaard, M.: Lift-and-project for mixed 0–1 programming: recent progress. Discrete Applied Mathematics **123**, 129–154 (2002)
4. Ben-Ameur, W., Neto, J.: Acceleration of cutting-plane and column generation algorithms: Applications to network design. Networks **49**(1), 3–17 (2007)
5. Benders, J.F.: Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik **4**, 238–252 (1962)
6. Bodur, M., Dash, S., Günlük, O., Luedtke, J.: Strengthened benders cuts for stochastic integer programs with continuous recourse. INFORMS Journal on Computing **29**(1), 77–91 (2017)
7. Conforti, M., Wolsey, L.A.: Facet separation with one linear program. Mathematical Programming **178**(1–2), 361–380 (2019)
8. Cordeau, J.F., Furini, F., Ljubić, I.: Benders decomposition for very large scale partial set covering and maximal covering location problems. European Journal of Operational Research **275**(3), 882–896 (2019)
9. Crainic, T.G., Hewitt, M., Rei, W.: Partial decomposition strategies for two-stage stochastic integer programs. Tech. Rep. 13, CIRRELT (2014)
10. Fischetti, M., Ljubic, I., Sinnl, M.: Benders decomposition without separability: A computational study for capacitated facility location problems. European Journal of Operational Research **253**(3), 557–569 (2016)
11. Fischetti, M., Ljubić, I., Sinnl, M.: Redesigning benders decomposition for large-scale facility location. Management Science **63**(7), 2146–2162 (2017)
12. Fischetti, M., Lodi, A., Tramontani, A.: On the separation of disjunctive cuts. Mathematical Programming **128**, 205–230 (2011)
13. Fischetti, M., Salvagnin, D., Zanette, A.: A note on the selection of Benders' cuts. Mathematical Programming B **124**, 175–182 (2010)
14. Geoffrion, A.M., Graves, G.W.: Multicommodity distribution system design by benders decomposition. Management Science **20**(5), 822–844 (1974)
15. Görtz, S., Klose, A.: A simple but usually fast branch-and-bound algorithm for the capacitated facility location problem. INFORMS Journal on Computing **24**(4), 597–610 (2012)
16. IBM CPLEX Optimizer: CPLEX User's Manual (2019), https://www.ibm.com/analytics/cplex-optimizer
17. Liu, X., Küçükyavuz, S., Luedtke, J.: Decomposition algorithms for two-stage chance-constrained programs. Mathematical Programming **157**(1), 219–243 (2016)
18. Louveaux, F.V.: Discrete stochastic location models. Annals of Operations Research **6**(2), 21–34 (1986)
19. Magnanti, T., Wong, R.: Accelerating Benders decomposition: algorithmic enhancement and model selection criteria. Operations Research **29**, 464–484 (1981)
20. McDaniel, D., Devine, M.: A modified Benders' partitioning algorithm for Mixed Integer Programming. Management Science **4**, 312–319 (1977)
21. Pan, F., Morton, D.P.: Minimizing a stochastic maximum-reliability path. Networks **52**(3), 111–119 (2008)

22. Papadakos, N.: Practical enhancements to the Magnanti-Wong method. Operations Research Letters **36**(4), 444–449 (2008)
23. Rahmaniani, R., Crainic, T.G., Gendreau, M., Rei, W.: The benders decomposition algorithm: A literature review. European Journal of Operational Research **259**(3), 801–817 (2017)
24. UflLib: Uncapacitated facility location library, http://resources.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/packages.html

A Appendix

A.1 Proof of Lemma 1

(i) Assume that (9)–(11) is infeasible. Then, Farkas lemma implies that there exists a ray $\pi \geq 0$ such that $\pi Q \leq 0$ and $\pi(r - Tx^*) > 0$. Multiplying (3) by π, and eliminating the y variables from the resulting constraint using $\pi Q \leq 0$ and $y \geq 0$, we get the inequality (12). This inequality is violated by x^* by definition of π.

(ii) Suppose now that (9)–(11) has a finite optimal value and let y^* be an optimal solution. If $dy^* \leq \eta^*$ we claim that (x^*, y^*) is optimal for (1)–(4). Indeed, it is feasible, and since (5)–(8) is a relaxation of (1)–(4) and $cx^* + dy^* \leq cx^* + \eta^*$, (x^*, y^*) is optimal. Otherwise, let's consider the optimal dual vector π. It satisfies the conditions $\pi Q \leq d$, $\pi(r - Tx^*) = dy^*$ and $\pi \geq 0$. Multiplying again (3) by π and using $\pi Q \leq d$, $y \geq 0$ and $dy \leq \eta$, we can eliminate the y variables from the resulting constraint and we obtain the inequality (13). This inequality is violated by the point (x^*, η^*) by strong duality.

A.2 Proof of Lemma 2

By definition, (u, u_0) is an unbounded ray of R if

$$u \geq 0, Au \geq 0, cu + u_0 < 0,$$

and (u, s) is an unbounded ray of P if

$$u \geq 0, Au \geq 0, s \geq 0, Tu + Qs \geq 0, cu + ds < 0. \tag{21}$$

Given an unbounded ray (u^*, u_0^*) of R, we want to check whether it can be turned into an unbounded ray (u^*, s^*) of P, meaning that P itself is unbounded, or find a cut that truncates R along (u^*, u_0^*).

(i) Assume (14)–(16) is unbounded, and consider an unbounded ray s^*. By definition, $s^* \geq 0$, $Qs^* \geq 0$ and $ds^* < 0$. Thus, there exists a scalar $\lambda > 0$ such that $(u^*, \lambda s^*)$ satisfies (21). This proves that P is unbounded.

(ii) Assume that (14)–(16) is infeasible. Then by Farkas lemma $\exists \pi \geq 0$ such that $\pi Q \leq 0$ and $\pi T u^* < 0$. Multiplying (3) by π, and eliminating the y variables from the resulting constraint using $\pi Q \leq 0$ and $y \geq 0$, we get the inequality (12). By definition of π we have $\pi T u^* < 0$ and thus the inequality truncates R along (u^*, u_0^*).

(iii) Finally, suppose that (14)–(16) is feasible and bounded and let s^* be an optimal solution. If $ds^* \leq u_0^*$, then (u^*, s^*) satisfies (21) and P is unbounded. Now suppose $ds^* > u_0^*$, and consider the optimal dual vector $\pi \geq 0$ which satisfies the conditions $\pi Q \leq d$ and $\pi T u^* = -ds^*$. Multiplying again (3) by π and using $\pi Q \leq d$, $y \geq 0$ and $dy \leq \eta$, we can eliminate the y variables from the resulting constraint and we obtain the inequality (13). From $\pi T u^* = -ds^*$ and $ds^* > u_0^*$ we get $\pi T u^* + u_0^* < 0$ and thus the inequality truncates R along (u^*, u_0^*).