Math. Program., Ser. B (2010) 124:175–182
DOI 10.1007/s10107-010-0365-7

FULL LENGTH PAPER

A note on the selection of Benders' cuts

Matteo Fischetti · Domenico Salvagnin · Arrigo Zanette

Received: 14 June 2008 / Accepted: 12 March 2010 / Published online: 12 May 2010

Abstract A new cut selection criterion for Benders' cuts is proposed and computationally analyzed. The results show that the new criterion is more robust—and often considerably faster—than the standard ones.

Keywords Benders' decomposition · Cutting planes · Mixed-integer programs

Mathematics Subject Classification (2000) 90C11 Mixed integer programming · 90C05 Linear programming · 49M27 Decomposition methods

1 Introduction

Consider a generic MIP

$$\min \left\{ c^T x + d^T y : \ Ax \geq b, \ Tx + Qy \geq r, \ x \in \mathbb{Z}^n_+, \ y \in \mathbb{R}^t_+ \right\}. \tag{1}$$

Benders' decomposition [5] works as follows. An artificial variable $\eta = d^T y$ is introduced along with a lower bound $\overline{\eta}$, and the *master problem* relaxation (assumed to be bounded)

$$\min \left\{ c^T x + \eta : \ Ax \geq b, \ \eta \geq \overline{\eta}, \ x \in \mathbb{Z}^n_+ \right\} \tag{2}$$

M. Fischetti (✉)
DEI, University of Padova, Padua, Italy
e-mail: fisch@dei.unipd.it

D. Salvagnin · A. Zanette
DMPA, University of Padova, Padua, Italy
e-mail: salvagni@math.unipd.it

A. Zanette
e-mail: zanettea@math.unipd.it

is solved, e.g., by an enumerative method. The optimal solution found, say (x^*, η^*) with x^* integer, is sent to the so-called *dual slave problem*

$$\max \left\{ \pi^T (r - Tx^*) : \pi^T Q \leq d^T, \pi \geq 0 \right\}. \tag{3}$$

If the dual slave problem is unbounded, an unbounded extreme ray $\overline{\pi}$ is chosen, and the *Benders' feasibility cut* $\overline{\pi}^T (r - Tx) \leq 0$ is added to the master, which is solved again. 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 the current (x^*, η^*) is feasible and hence optimal for (1). If not, the *Benders' optimality cut* $\eta \geq \overline{\pi}^T (r - Tx)$ is added to the current master problem, and the method is repeated.

In the present work we address the topic of selecting Benders' cuts for general MIPs in an effective way. The choice of effective Benders' cuts was already addressed by Benders [5], Magnanti and Wong [12], and Wentges [15], among others. Cut selection policies are of fundamental importance for any cutting plane method, and have been investigated deeply in the general context of convex optimization. A notable case arises in disjunctive programming, where valid cuts are generated by solving a certain *Cut Generating LP* (CGLP) akin to Benders' dual slave (3). In this context, the CGLP is usually stated in a form where cut violation acts as the objective function to be maximized, and a (linear) normalization condition is added to deal with unboundedness. As the actual choice of the violated cut heavily depends on the normalization used, different normalization conditions lead to different cut selection criteria. We refer the reader to Cornuéjols [7] and to Cornuéjols and Lemaréchal [8] for a treatment of disjunctive cuts, and in particular to [8, Sect. 4], for a thorough discussion on the role of normalization in the very general context where cuts are generated by optimal solutions of convex programs—thus generalizing CGLPs. Nonlinear conditions for disjunctive cuts have been recently investigated in Cadoux [6], where the Euclidean distance cut-off is used to measure cut quality.

2 A new selection criterion for Benders' cut

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. In this setting, finding a most-violated optimality cut is equivalent to finding an optimal vertex of a polyhedron with unbounded rays, which is a strongly NP-hard problem [10].

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:

$$d^T y \leq \eta^*, \;\; Qy \geq r - Tx^*, \;\; y \geq 0 \tag{4}$$

or, equivalently, if the following modified dual slave problem is unbounded:

$$\max \left\{ \pi^T (r - Tx^*) - \pi_0 \eta^* : \ \pi^T Q \leq \pi_0 d^T, \ \pi, \pi_0 \geq 0 \right\}. \tag{5}$$

The cut associated with a given ray $(\overline{\pi}, \overline{\pi}_0)$ of (5) reads $\overline{\pi}^T (r - Tx) - \overline{\pi}_0 \eta \leq 0$.

In practice, one is interested in detecting a "minimal source of infeasibility" of (4), so as to detect a small set of constraints in the slave that suffices to cut the master solution. According to Gleeson and Ryan [11], the rows of any *Minimal* (with respect to set inclusion) *Infeasible Subsystem* (MIS) of (4) are indexed by the support of the vertices of the following polyhedron, sometimes called the *alternative polyhedron*:

$$\left\{ (\pi, \pi_0) \geq 0 : \ \pi^T Q \leq \pi_0 d^T, \ \pi^T (r - Tx^*) - \pi_0 \eta^* = 1 \right\} \tag{6}$$

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 $\sum_{i=1}^{m} w_i \pi_i + w_0 \pi_0$ to minimize (where m denotes the number of rows of Q), it is therefore possible to optimize over the alternative polyhedron so as to select a violated cut corresponding to a MIS of (4) with certain useful properties.

As observed in [8], one can swap the role of the objective function and of the normalization condition in (6) to obtain an equivalent CGLP akin the one introduced for disjunctive cuts by Balas, Ceria, and Cornuéjols [4]:

$$\max \left\{ \pi^T (r - Tx^*) - \pi_0 \eta^* : \pi^T Q \leq \pi_0 d^T, \sum_{i=1}^{m} w_i \pi_i + w_0 \pi_0 = 1, (\pi, \pi_0) \geq 0 \right\}. \tag{7}$$

This latter formulation is often preferable from a computational point of view.

Of course, the choice of 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. The original Benders' CGLP (3) arises as a particular case by setting $w_0 = 1$ and $w_1 = \cdots = w_m = 0$. A more clever choice is however to set $w_0 = \cdots = w_m = 1$, with the aim of reducing the cardinality of the support of the optimal vertex of (6), and hence to heuristically find a minimum-cardinality MIS–which is an NP-hard problem [1]. This is in fact the choice typically used for disjunctive CGLPs, whose practical effectiveness can be explained because it tends to produce a "minimum-cardinality certificate" for x^* infeasibility.

In order to obtain even better cuts, we observe 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, so one should set $w_i = 0$. According to our computational experience, this simple change leads to a substantial improvement of the generated cuts in many cases, so our final choice is to consider the following CGLP normalization condition in (7):

$$\sum_{i \in I(T)} \pi_i + w_0 \pi_0 = 1 \tag{8}$$

where w_0 (just set to 1 in our tests) is a possible scaling factor taking into account a wider range of variable η, and $I(T)$ indexes the nonzero rows of T.

3 Computational experiments

The Benders' scheme addressed in the present paper has 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 Intel Core2 Q6600 PC running at 2.40 GHz with 4 GB of RAM, with a time limit of 10 h 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), whenever 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 is 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 also 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 [14]. 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 (3); this is equivalent to using CGLP (7) with the "blind" normalization $\pi_0 = 1$;
- `std2` the enhanced approach in [5] where (3) is truncated by $\sum_{i=1}^{m} \pi_i \leq M$, M is a dynamically updated big-M coefficient; when feasibility cuts are generated, this is equivalent to using CGLP (7) with $\sum_{i=1}^{m} \pi_i + \pi_0 = 1$;
- `mis` our method using CGLP (7) with normalization $\sum_{i \in I(T)} \pi_i + \pi_0 = 1$.

The three Benders' implementations above are completely general purpose, and only differ in the way the cuts are selected—all other features are identical. Each call to the CGLP in `std` and `std2` can generate *two* distinct violated cuts, associated respectively with a vertex and an unbounded ray of the (3), whereas it generates at most one cut for `mis`. Both `std` and `std2` (but not `mis`) implement the Magnanti-Wong [12] cut selection rule. The warm-start procedure of McDaniel and Devine [13] was instead deactivated for all methods because it resulted into a generalized performance degradation in our testbed.

The four MIP methods above have been compared on two sets of MIP instances whose structure is known to be well suited for Benders' methods [9]. All instances and detailed tables of results are available, on request, from the authors. Our first testbed

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

Topology	Nodes	Type	# constr.s	# var.s	# int.var.s
Grid	5×5	feas	2,318	6,281	34
Grid	5×5	opt	2,236	5,974	34
Grid	5×6	feas	4,282	11,622	41
Grid	5×6	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

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. We generated two different types of random instances of the above network design problem, the underlying topology being defined either as a `grid` or as a `random` network. In addition, 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, hence both feasibility and optimality Benders' cuts were generated. Table 1 summarizes the main characteristics of the instances in our network design testbed.

Table 2 reports the outcome of our experiments on the network design testbed; results for `cpx` are not reported because this method exceeded the 10-hour time limit in most runs. The table gives average results (geometric means) over 5 instances for each class and scenario. 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.

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 three times more iterations and cuts than `std`, and takes about 25 times more computing time. Evidently, the simple 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 3.5 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.

Table 3 reports the detailed results on the instances where at least one of the Benders' variants reached the 10-hour time limit. Column *gap* reports the integrality gap

Table 2 Network design results

Type	Class	Method	Time (s)	sep (%)	iter.s	# feas	# opt
feas	g_5_5	std	22.77	93.0	125	249	0
		std2	548.86	96.1	266	592	0
		mis	14.05	98.1	75	106	0
	g_5_6	std	100.44	98.6	138	250	0
		std2	3,213.48	97.7	408	948	0
		mis	51.41	98.6	102	161	0
	r_20_5	std	52.74	90.2	129	270	0
		std2	1,167.61	57.3	489	1,021	0
		mis	33.65	93.2	92	166	0
	r_25_5	std	347.38	84.8	199	462	0
		std2	6,571.72$^{(1)}$	64.5	581	1,418	0
		mis	148.56	76.5	92	182	0
	all	std	80.46	91.5	145	297	0
		std2	1,918.02	76.7	419	949	0
		mis	43.59	91.1	90	151	0
opt	g_5_5	std	65.58	24.3	162	525	528
		std2	263.61	65.2	152	585	589
		mis	18.57	98.3	77	104	7
	g_5_6	std	156.54	32.3	182	790	794
		std2	1,346.45	78.4	295	1,026	1,036
		mis	69.86	99.2	97	136	21
	r_20_5	std	480.15	10.1	181	594	812
		std2	2,327.40$^{(1)}$	13.9	296	782	1,324
		mis	109.00	58.0	127	124	302
	r_25_5	std	14,194.35$^{(2)}$	6.7	492	1,749	2,140
		std2	36,001.24$^{(5)}$	11.6	593	2,562	3,361
		mis	3,070.51$^{(1)}$	38.7	351	268	1,162
	all	std	514.32	15.2	226	810	924
		std2	2,335.27	30.1	298	1,047	1,284
		mis	144.34	68.4	135	147	86

when the algorithm was stopped. For these hard instances, `mis` performed even better when compared to `std` and `std2`. In addition, `mis` reached the time limit only for one instance, `r_25_5_4_opt`, for which all other methods reached the time limit as well but closed a significantly smaller gap.

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. The corresponding average results are reported in Table 4; again, `cpx` is not reported because it exceeded the time limit in most cases.

Table 3 Detailed results on network design instances where at least one method reached the 10-hour time limit

Instance	Method	Time (s)	gap (%)	sep (%)	iter.s	# feas	# opt
r_25_5_3_feas	std	9,248.69	0.00	52.4	1,239	2,207	0
	std2	36,000.00	2.22	38.8	1,412	3,036	0
	mis	1,652.23	0.00	27.6	214	422	0
r_20_5_4_opt	std	2,872.06	0.00	3.3	339	657	1,304
	std2	36,000.00	2.00	2.3	1,008	1,110	4,019
	mis	617.20	0.00	33.1	247	186	813
r_25_5_1_opt	std	10,194.60	0.00	6.7	443	1,255	1,864
	std2	36,000.00	0.52	8.2	760	1,970	3,449
	mis	1,447.93	0.00	58.7	277	187	904
r_25_5_2_opt	std	5,699.31	0.00	5.4	241	1,365	1,466
	std2	36,000.00	0.47	4.3	542	1,857	2,193
	mis	327.36	0.00	70.9	109	159	350
r_25_5_3_opt	std	36,000.00	1.13	7.1	1,469	3,320	3,335
	std2	36,000.00	2.97	25.1	539	3,802	3,865
	mis	17,063.10	0.00	26.7	795	375	2,794
r_25_5_4_opt	std	36,000.00	1.36	5.5	590	3,142	3,607
	std2	36,000.00	2.73	17.5	511	3,472	3,838
	mis	36,000.00	0.71	10.5	1,117	539	3,532
r_25_5_5_opt	std	7,651.76	0.00	9.2	313	917	1,365
	std2	36,000.00	0.65	13.9	644	2,286	3,824
	mis	937.40	0.00	74.5	198	230	679

Table 4 Network expansion results

Class	Method	Time (s)	sep (%)	iter.s	# feas
100.20.2	std	236.73	0.4	195	387
	std2	20.97	4.8	136	218
	mis	23.84	5.0	139	214
100.20.4	std	450.83	0.2	140	392
	std2	66.39	1.5	122	270
	mis	68.90	1.9	121	266
100.20.8	std	456.60	0.2	139	409
	std2	65.55	1.7	137	321
	mis	61.84	2.4	136	338
150.20.2	std	2,135.75	0.1	291	659
	std2	109.59	2.9	231	346
	mis	114.69	5.5	241	357
150.20.4	std	4,944.44(1)	0.0	169	672
	std2	2,086.67(1)	0.2	207	541
	mis	1,645.82(1)	0.6	213	542
150.20.8	std	5,393.15	0.0	181	655
	std2	1,493.35	0.4	204	623
	mis	1,675.16	0.6	202	617

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`. 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 class.

On the whole, the results show that our new cut selection criterion in more robust than its competitors, 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.

Acknowledgments 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.

References

1. Amaldi, E., Pfetsch, M.E., Trotter, L.E. Jr.: On the maximum feasible subsystem problem, IISs and IIS-hypergraphs. Math. Program. **95**(3), 533–554 (2003)
2. Atamtürk, A., Nemhauser, G.L., Savelsbergh, M.W.P.: Valid inequalities for problems with additive variable upper bounds. Math. Program. **91**, 145–162 (2001)
3. Atamturk, A., Rajan, D.: On splittable and unsplittable flow capacitated network design arc-set polyhedra. Math. Program. **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. Manage. Sci. **42**, 1229–1246 (1996)
5. Benders, J.: Partitioning procedures for solving mixed-variables programming problems. Numer. Math. **4**, 238–252 (1962)
6. Cadoux, F.: Computing deep facet-defining disjunctive cuts for mixed-integer programming. Math. Program. **122**(2), 197–223 (2009)
7. Cornuéjols, G.: Valid inequalities for mixed integer linear programs. Math. Program. Ser. B **112**(1), 3–44 (2008)
8. Cornuéjols, G., Lemaréchal, C.: A convex analysis perspective on disjunctive cuts. Math. Program. **106**(3), 567–586 (2006)
9. Costa, A.M.: A survey on Benders decomposition applied to fixed-charge network design problems. Comput. Oper. Res. **32**(6), 1429–1450 (2005)
10. Fukuda, K., Liebling, T.M., Margot, F.: Analysis of backtrack algorithms for listing all vertices and all faces of a convex polyhedron. Comput. Geome. **8**, 1–12 (1997)
11. Gleeson, J., Ryan, J.: Identifying minimally infeasible subsystems of inequalities. ORSA J. Comput. **2**(1), 61–63 (1990)
12. Magnanti, T., Wong, R.: Accelerating Benders decomposition: algorithmic enhancement and model selection criteria. Oper. Res. **29**, 464–484 (1981)
13. McDaniel, D., Devine, M.: A modified Benders' partitioning algorithm for mixed integer programming. Manage. Sci. **4**, 312–319 (1977)
14. Rei, W., Cordeau, J.F., Gendreau, M., Soriano, P.: Accelerating Benders decomposition by local branching. INFORMS J. Comput. **21**, 333–345 (2009)
15. Wentges, P.: Accelerating Benders' decomposition for the capacitated facility location problem. Math. Methods Oper. Res. **44**, 267–290 (1996)