Mixed Integer NonLinear Programs featuring "On/Off" constraints: convex analysis and applications

Hassan Hijazi · Pierre Bonami · Gérard Cornuéjols · Adam Ouorou

October 2009

1 Computational testings

We compare the four models introduced in [6] on real world networks (denoted rdatax) as well as on randomly generated networks (denoted adatax), with up to 100 vertices and 1000 commodities. All models are implemented in C++ and solved with Bonmin (release 1.1.3) [2] an open source convex MINLP solver (see `http://www.coin-or.org/Bonmin`). The time limit for Bonmin is set to 2 hours. The underlying MILP solver used is CBC [5] and the nonlinear programming solver is Ipopt [8]. All tests were performed on an Intel Xeon 1.6 Ghz CPU.
Bonmin offers the possibility to choose one of five solution algorithms: a nonlinear programming based Branch & Bound [3], an Outer Approximation decomposition [4], and three branch-and-cut algorithms based on the Quessada Grossmann algorithm [7], a vanilla implementation of this algorithm, a hybrid method including a preliminary phase of Outer Approximation Decomposition and periodically adding outer approximation cuts, and finally, a method based on adding Extended Cutting Plane cuts [9] (similar to the method proposed in [1]). Here we report results obtained with the hybrid method since it appeared to be consistently better than the others (with all four models) in preliminary experiments. In the following tables, we compare the computing time to optimality and the number of nodes developed in the branch and bound with the four models, results are reported in tables with the following form: **(cpu time ; number of nodes)**. If optimality is not reached within the time limit, the gap between the current best

First and second authors are supported by ANR grant ANR06-BLAN-0375.

Third author is supported by NSF grant CMMI0653419, ONR grant N00014-03-1-0133 and ANR grant ANR06-BLAN-0375.

H. Hijazi
Orange Labs R&D/CORE-MCN, 38-40 rue du Général Leclerc, 92794 Issy-Les-Moulineaux cedex 9, France; and LIF, CNRS-Aix Marseille Université, Parc Scientifique et Technologique de Luminy, Marseille, France.

P. Bonami
LIF, CNRS-Aix Marseille Universite, Parc Scientifique et Technologique de Luminy, Marseille, France.

G. Cornuéjols
Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA 15213, USA;
and LIF, CNRS-Aix Marseille Universite, Parc Scientifique et Technologique de Luminy, Marseille, France.

A. Ouorou
Orange Labs R&D/CORE-MCN, 38-40 rue du Général Leclerc, 92794 Issy-Les-Moulineaux cedex 9, France.

Table 1 Mono-routing analysis, 2 paths per commodity

	$\|V\|$	$\|E\|$	$\|K\|$	P_{bigM}	P_{proj}	P_{high}	P_{red}
rdata1	60	280	100	**(0.4 ; 0)**	**(0.4 ; 0)**	(35 ; 0)	(2 ; 0)
rdata2	61	148	122	(190 ; 5193)	**(155 ; 1012)**	(2997 ; 21312)	(1948 ; 15129)
adata3	100	600	200	(144 ; 335)	**(57 ; 0)**	(206 ; 556)	(159 ; 84)
rdata4	34	160	946	**(3 ; 0)**	**(3 ; 0)**	(2040 ; 11691)	(1485 ; 5845)
rdata5	67	170	761	([∞] ; 14788)	**(251 ; 3357)**	(1549 ; 2793)	([0.03%] ; 15697)
adata6	100	800	500	**(1065 ; 27991)**	(1470 ; 42244)	([1%] ; 6831)	([0.3%] ; 2499)

Table 2 Mono-routing analysis, 3 paths per commodity.

	$\|V\|$	$\|E\|$	$\|K\|$	P_{bigM}	P_{proj}	P_{high}	P_{red}
rdata1	60	280	100	(2.7 ; 0)	**(2.4 ; 0)**	(2.8 ; 0)	(11.9 ; 0)
rdata2	61	148	122	(25 ; 0)	**(13 ; 0)**	(994 ; 3671)	(1396; 7815)
adata3	100	600	200	([0.28%] ; 157748)	(344; 5097)	(722 ; 3286)	**(312 ; 1124)**
rdata4	34	160	946	([0.001%] ; 79807)	**(1525 ; 50583)**	([0.04%] ; 28876)	([0.1%] ; 22438)
rdata5	67	170	761	([0.43%] ; 138618)	**([0.03%] ; 202122)**	([0.2%] ; 9472)	([0.14%] ; 9461)
adata6	100	800	500	([0.006%] ; 176413)	**(934 ; 19351)**	([0.6%] ; 16539)	([0.06%] ; 4067)

Table 3 mono-routing analysis, 10 paths per commodity.

	$\|V\|$	$\|E\|$	$\|K\|$	P_{bigM}	P_{proj}	P_{high}	P_{red}
rdata1	60	280	100	(568 ; 13649)	**(231 ; 4762)**	(1415 ; 9263)	(1209 ; 6485)
rdata2	61	148	122	(120 ; 0)	**(66 ; 0)**	(1527 ; 1599)	(1555 ; 2563)
adata3	100	600	200	**(534 ; 5118)**	(644 ; 14216)	(4866 ; 8841)	(6684 ; 11626)
rdata4	34	160	946	([1.9%] ; 79807)	([2.1%] ; 96212)	([∞] ; 3409)	**([1.8%] ; 3156)**
rdata5	67	170	761	([∞] ; 37446)	([∞] ; 30500)	([∞] ; 747)	([∞] ; 1568)
adata6	100	800	500	([2.7%] ; 35520)	**([1.5%] ; 2680)**	([∞] ; 2642)	([∞];1001)

integer feasible solution and the continuous relaxation is displayed inside brackets, ∞ indicates that no integer feasible solution has been found after two hours of computing times. For each instance, the best computing time or the smallest gap is listed in bold characters.
In the following tables, we report results obtained for different networks and different parameters using the hybrid algorithm in Bonmin. In tables 1, 2 and 3, we consider the case when N = 1, that is only one path per commodity can be activated at a time (mono-routing problems). The number of candidate paths per commodity is set to 2, 3 and 10 respectively in tables 1, 2 and 3.

First, let us point out the fact that having zero node explored in the Branch & Bound means that the problem has been solved during the Outer Approximation Decomposition [4] initial phase of the hybrid algorithm (it means in no case that the initial continuous relaxation is integer feasible). The main observation is that the model based on the projected convex hull P_{proj} gives the best performance on these instances. P_{proj} solves 14 instances out of 18 while P_{bigM}, P_{high} and P_{red} solve respectively 10, 11 and 10 instances. If we consider geometric means, P_{proj} is 2.1 times faster than P_{bigM}, 6.7 times faster than P_{high} and 6.3 times faster than P_{red}. The advantage in terms of number of nodes is comparable. If we look only at the four problems which are solved by P_{proj} but not solved by P_{bigM}, P_{proj} is at least one order of magnitude faster than P_{bigM} with at least 5 times less nodes to reach optimality (it is at least 5.45 and 7.17 times faster that P_{high} and P_{red}). One can conclude that even if P_{high} and P_{red} provide better continuous relaxations and usualy require less nodes, they remain slower due to the increased number of variables. Problem adata3 in Table 2 gives a good illustration of this: one can see that optimality is reached with P_{red} in only 1124 nodes, where the Big-M model explored 157748 nodes without reaching the optimum. However, P_{high} and P_{red} often remain slower to solve since they have an important number of additional variables. On the same problem, the continuous relaxation of P_{red} is solved in 3.5 secs while for P_{bigM} it takes only 0.5 secs. The projected model

Table 4 Bi-routing analysis, 2 paths per commodity.

	$\|V\|$	$\|E\|$	$\|K\|$	P_{bigM}	P_{proj}
rdata1	60	280	100	(0.8 ; 0)	**(0.4 ; 0)**
rdata2	61	148	122	**(2.4 ; 0)**	(10.9 ; 0)
adata3	100	600	200	(47.6 ; 0)	**(4.7 ; 0)**
rdata4	34	160	946	(5.6 ; 0)	**(3.4 ; 0)**
rdata5	67	170	761	**(38.3 ; 0)**	(50.4 ; 0)
adata6	100	800	500	**(40.1 ; 0)**	(42.4 ; 0)

Table 5 Bi-routing analysis, 3 paths per commodity.

	$\|V\|$	$\|E\|$	$\|K\|$	P_{bigM}	P_{proj}
rdata1	60	280	100	(16.7 ; 0)	**(2.9 ; 0)**
rdata2	61	148	122	(129.2 ; 18)	**(59 ; 0)**
adata3	100	600	200	(291.5 ; 760)	**(171.3 ; 615)**
rdata4	34	160	946	**(154.9 ; 62)**	(343.8 ; 472)
rdata5	67	170	761	([0.15%] ; 91430)	**(609.1 ; 5290)**
adata6	100	800	500	(1579.8 ; 8176)	**(747.7 ; 7056)**

Table 6 Bi-routing analysis, 10 paths per commodity.

	$\|V\|$	$\|E\|$	$\|K\|$	P_{bigM}	P_{proj}
rdata1	60	280	100	(1909 ; 56788)	**(399 ; 7846)**
rdata2	61	148	122	**(28.8 ; 0)**	(288.8 ; 666)
adata3	100	600	200	**([0.11%] ; 67705)**	([0.13%] ; 77129)
rdata4	34	160	946	([1.2%] ; 23984)	**([0.6%] ; 32939)**
rdata5	67	170	761	([2%] ; 11772)	**([1.2%] ; 16285)**
adata6	100	800	500	([0.7%] ; 6480)	**([0.14%] ; 22364)**

P_{proj} is able to give bounds almost as tight as the extended formulations, without having to deal with the inconvenience of large size problems (still on the same instance the continuous relaxation of P_{proj} is solved in 0.9 secs).
We now consider the bi-routing case, maximum two paths can be activated per commodity, in order to route fractions of the demand ($N = 2$).
From previous results on the mono-routing case, it appears clearly that P_{proj} is consistently better than P_{high} and P_{red} for all instances but one where they are equivalent. Furthermore, since the bi-routing and the multiple-routing case involve adding new variables corresponding to fractions of demands (ϕ_k^i), these high dimensional relaxations would be even larger and more difficult to solve in these cases. For this reason, P_{high} and P_{red} were not implemented in the remaining of the experiments. Tables 4, 5 and 6 reports results obtained for instances with respectively 2, 3 and 10 paths per commodity.

First, we note that bi-routing problems seem in general easier to solve than their mono-routing counterparts. For these problems P_{bigM} and P_{proj} solved respectively 13 and 14 instances. Instances having 2 paths per commodity seem very easy to solve (they are all solved in less than one minute with both formulations) and instances with 3 path seem much easier than before. On average P_{proj} is still faster than P_{bigM} taking about 1761 secs versus 2235 secs.
We finally consider the multiple-routing case where all the paths in $P(k)$ can be activated ($N = \infty$). Tables 7 and 8 reports respectively results obtained for instances with 3 and 10 paths per commodity (instances with 2 paths are similar in the bi-routing and multi-routing cases).

Multi-routing problems bring the same observations as the bi-routing case. For these problems P_{bigM} and P_{proj} solved respectively 8 and 11 instances out of 12. Now, all instances with

Table 7 Multiple-routing analysis, 3 paths per commodity.

	$\|V\|$	$\|E\|$	$\|K\|$	P_{bigM}	P_{proj}
rdata1	60	280	100	(2.7 ; 0)	**(1.2 ; 0)**
rdata2	61	148	122	**(10 ; 0)**	(29.7 ; 0)
adata3	100	600	200	(195.7 ; 164)	**(50.5 ; 0)**
rdata4	34	160	946	**(20.7 ; 0)**	(43 ; 0)
rdata5	67	170	761	(2503.1 ; 34783)	**(362.8 ; 4024)**
adata6	100	800	500	(1482.4 ; 6083)	**(224.7 ; 326)**

Table 8 Multiple-routing analysis, 10 paths per commodity

	$\|V\|$	$\|E\|$	$\|K\|$	P_{bigM}	P_{proj}
rdata1	60	280	100	(799.7 ; 12633)	**(220.8 ; 1922)**
rdata2	61	148	122	**(16.1 ; 0)**	(24.8 ; 0)
adata3	100	600	200	([0.08%] ; 94194)	**(768.6 ; 5207)**
rdata4	34	160	946	([0.4%] ; 40820)	**([0.04%] ; 45492)**
rdata5	67	170	761	([1.2%] ; 16106)	**(5467.7 ; 17347)**
adata6	100	800	500	([0.7%] ; 5880)	**(5392 ; 23867)**

3 paths per commodity seem quite easy to solve with both formulations. On average, for all instances, P_{proj} takes 1649 secs while P_{bigM} takes 2819 secs. For the instance not solved by both formulations, P_{proj} final gap is about one order of magnitude smaller than the P_{bigM} one.

References

1. K. Abhishek, S. Leyffer, and J. T. Linderoth. FilMINT: An outer-approximation-based solver for nonlinear mixed integer programs. Preprint ANL/MCS-P1374-0906, Mathematics and Computer Science Division, Argonne National Laboratory, 2006.
2. P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186–204, 2008.
3. R. J. Dakin. A tree search algorithm for mixed programming problems. *Computer Journal*, 8:250–255, 1965.
4. M. A. Duran and I. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36:307–339, 1986.
5. J. Forrest. CBC, 2004. Available from `http://www.coin-or.org/`.
6. H. Hijazi, P. Bonami, G. Cornuéjols, and A. Ouorou. Mixed integer nonlinear programms featuring "on/off" constraints : convex analysis and applications. *Computational Optimization and Applications*, Submitted October 2009.
7. I. Quesada and I. E. Grossmann. An LP/NLP based branch–and–bound algorithm for convex MINLP optimization problems. *Computers and Chemical Engineering*, 16:937–947, 1992.
8. A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.
9. T. Westerlund and F. Pettersson. A cutting plane method for solving convex MINLP problems. *Computers and Chemical Engineering*, 19:s131–s136, 1995.