Mixed Integer Programming vs. Logic-based Benders Decomposition for Planning and Scheduling*

André Ciré, Elvin Coban, and J. N. Hooker

Carnegie Mellon University, USA
acire,ecoban,jh38@andrew.cmu.edu

Abstract. A recent paper by Heinz and Beck (CPAIOR 2012) found that mixed integer software has become competitive with or superior to logic-based Benders decomposition for the solution of facility assignment and scheduling problems. Their implementation of Benders differs, however, from that described in the literature they cite and therefore results in much slower performance than previously reported. We find that when correctly implemented, the Benders method remains 2 to 3 orders of magnitude faster than the latest commercial mixed integer software on larger instances, thus reversing the conclusion of the earlier paper.

1 Introduction

Logic-based Benders decomposition (LBBD) [12, 21, 22] is a generalization of classical Benders decomposition [3] that accommodates an arbitrary optimization problem as the subproblem. LBBD is particularly attractive for planning and scheduling problems, in which the master problem can use MIP to allocate jobs to resources and the subproblem can use CP to schedule jobs on each resource. Implementations of this method have obtained computational results superior to those of state-of-the-art MIP and CP solvers, sometimes by several orders of magnitude [4–6, 8–10, 13, 15–17, 19, 23–27].

A recent study by Heinz and Beck [11] finds, however, that MIP software has improved to the point that it is competitive with or superior to LBBD on a class of planning and scheduling problems on which LBBD previously excelled [19]. MIP software has in fact improved significantly since the earlier results were published (2007). Yet the computation times reported in [11] for LBBD are much longer than those obtained in earlier studies, including [19].

A partial explanation for this discrepancy is that [11] incorrectly implements the Benders cuts described in the references it cites [16, 18, 19]. In addition, it solves the CP subproblems with a significantly slower method than the state of the art. We therefore re-implemented LBBD, for the purpose of reproducing previous results and comparing them with recent MIP software. We found that LBBD remains superior to state-of-the-art MIP on this class of problems, as it

* Partial support from NSF grant CMMI-1130012 and AFOSR grant FA-95501110180.

is 2 to 3 orders of magnitude faster than CPLEX 12.4 on larger instances. These results have subsequently been confirmed by Beck and Ku in unpublished work [2].

2 Previous Work

Table 1 shows computational results for several methods tested by Heinz and Beck [11] as well as previous LBBD results from [19]. The results are for the "c" instances used in [13, 19] and available online [14]. There are n jobs to be assigned to m facilities and then scheduled on each facility, using cumulative scheduling. The objective is to minimize total cost of assigning jobs to facilities, although other objectives have been used [16, 17, 19].

The results from [11] shown in the table are implemented in SCIP on an Intel Xeon E5420 2.5 GHz machine. CIP(CP) and CIP(MIP) refer to CP/MIP hybrids. The LBBD results from [19] were obtained from an implemention in OPL Studio.

The discrepancy in LBBD results may be seen by examining the boldface figures in Table 1. The difference is actually greater than shown, because the SCIP results are shifted geometric means[1] while the earlier LBBD results are averages. There is an even larger discrepancy with results obtained in [28] by the integrated solver SIMPL for similar instances.

One possible explanation for the discrepancy is that [11] uses relatively weak nogood cuts as Benders cuts. When a set J of jobs assigned to facility i is found to have no feasible schedule, the Benders cut $\sum_{j \in J}(1 - x_{ij}) \geq 1$ is added to the master problem, where 0-1 variable $x_{ij} = 1$ when job j is assigned to facility i. The cut prevents this set same of jobs (or any superset) from being assigned to facility i in subsequent iterations. However, earlier studies use strengthened cuts that are obtained heuristically by re-solving the scheduling problem on facility i for subsets of J, so as to find a smaller set J' of jobs that have no feasible schedule. This results in the stronger cut $\sum_{j \in J'}(1 - x_{ij}) \geq 1$. Previous work suggests that the strengthening procedure can bring significant improvement in performance, and this may explain part of the discrepancy. We test this hypothesis in the next section.

3 Computational Results

Table 2 shows our results for the "c" instances. The MIP results are obtained by CPLEX 12.4.01. LBBD is implemented by solving the master problem with CPLEX 12.4.01 and the subproblems with IBM CP Optimizer 12.4.01 using extended filtering, DFS search, and default variable and value selection. All tests are run on an Intel Xeon E5345 2.33 GHz (64 bits) in single core mode with 8 GB RAM.

[1] The shifted geometric mean of $v_1, \ldots, v_n$ is $(\Pi_i(v_i + s))^{1/n} - s$ for shift s. Following [11], we use $s = 10$ seconds. The shifted geometric mean of a set of distinct values is smaller than the average.

Table 1. Computational results for planning and scheduling "c" instances as reported by [11] and [19], showing number of instances solved to optimality (out of 5) and solution time. Boldface figures show contrasting reuslts for LBBD.

		Reported in [11]								LBBD	
Size		MIP (SCIP)		LBBD		CIP(CP)		CIP(MIP)		in [19]	
m	n	Solved	Sec[1]	Solved	Sec[1]	Solved	Sec[1]	Solved	Sec[1]	Solved	Sec[2]
2	10	5	1	**5**	**1**	5	0	5	2	**5**	**0**
	12	5	1	**5**	**1**	5	0	5	4	**5**	**0**
	14	5	1	**5**	**3**	5	1	5	5	**5**	**0**
	16	5	11	**5**	**17**	5	19	5	30	**5**	**2**
	18	4	162+	**5**	**89**	5	18	5	77	**5**	**9**
	20	3	401+	**3**	**158+**	5	3	5	139	**5**	**111**
	22	2	1442+	**2**	**703+**	2	1325+	4	2550+	**<5**	**1805+**
	24	2	2197+	0	-	3	707+	3	1180+		
	26	3	2977+	1	5193+	1	5440+	2	3261+		
	28	2	2503+	3	441+	3	160+	2	2598+		
	30	1	5429+	1	2972+	0	-	1	4180+		
	32	0	-	1	5680+	3	282+	1	6123+		
3	10	5	0	**5**	**1**	5	0	5	1	**5**	**0**
	12	5	1	**5**	**1**	5	1	5	5	**5**	**0**
	14	5	1	**5**	**1**	5	2	5	11	**5**	**0**
	16	5	14	**5**	**10**	5	22	5	60	**5**	**1**
	18	5	429	**5**	**21**	4	139+	4	296+	**5**	**3**
	20	4	1124+	**5**	**6**	5	35	5	253	**5**	**3**
	22	2	6014+	**5**	**149**	2	1352+	4	505+	**5**	**8**
	24	2	3253+	**1**	**2324+**	1	3165+	5	1001	**5**	**19**
	26	0	-	4	1351+	3	727+	1	4467+		
	28	2	1829+	0	-	2	1261+	2	3057+		
	30	0	-	0	-	0	-	2	5435+		
	32	0	-	0	-	1	6918+	1	6639+		
4	10	5	0	**5**	**1**	5	1	5	1	**5**	**0**
	12	5	1	**5**	**1**	5	1	5	4	**5**	**0**
	14	5	1	**5**	**1**	5	2	5	12	**5**	**0**
	16	5	2	**5**	**10**	5	1	5	11	**5**	**0**
	18	5	38	**5**	**21**	5	4	5	67	**5**	**1**
	20	5	309	**5**	**6**	5	27	5	106	**5**	**1**
	22	4	324	**5**	**149**	5	46	5	544	**5**	**3**
	24	0	-	**1**	**2324+**	2	1446+	2	4184+	**5**	**39**
	26	0	-	**4**	**1351+**	1	4070+	2	5530+	**5**	**29**
	28	1	5034+	0	-	1	2804+	2	3885+		
	30	0	-	0	-	1	2105+	0	-		
	32	0	-	0	-	0	-	0	-		

[1]Shifted geometric mean [2]Average
+Computation terminated after 7200 sec for instances not solved to optimality.

The new LBBD results are somewhat better than the old ones in [19], presumably due to faster MIP and CP components. More importantly, LBBD remains significantly faster than the most recent CPLEX solver, particularly when there are 3 or 4 facilities. Decomposition is obviously more helpful when there are more facilities, because more decoupling is possible in the subproblem.

The "c" instances used in [13, 19] put LBBD at a disadvantage in two ways: they use relatively few facilities, and the facilities differ greatly (by a factor of m) in speed. The master problem assigns most jobs to the fastest facilities, resulting in harder scheduling problems for CP. We therefore ran MIP and LBBD on problem set "e," also used in [13, 19]. These instances have more facilities, with speeds differing by no more than a factor of 1.5. The results appear in Table 3. The LBBD advantage is even greater on these instances, easily solving

Table 2. Computational results for planning and scheduling "c" instances, showing the number of problem instances solved (out of 5) and computation time. The results are obtained for (a) the CPLEX MIP solver, (b) logic-based Benders decomposition implemented with simple nogood cuts, and (c) LBBD implemented with the strengthened cuts used in previous studies.

Size		MIP (CPLEX)			LBBD: Weak cuts			LBBD: Strong cuts		
m	n	Solved	Sec[1]	Sec[2]	Solved	Sec[1]	Sec[2]	Solved	Sec[1]	Sec[2]
2	10	5	0.1	0.1	5	0.1	0.1	5	0.1	0.1
	12	5	0.2	0.2	5	0.1	0.1	5	0.0	0.0
	14	5	0.1	0.1	5	0.1	0.1	5	0.0	0.0
	16	5	8.9	28	5	0.2	0.2	5	0.3	0.3
	18	5	65	388	5	0.5	0.5	5	0.6	0.7
	20	4	221+	1902	5	1.9	2.0	5	6.4	8.0
	22	3	1055+	3849+	5	38	617	5	67	955
	24	2	1000+	4351+	4	110+	1495+	5	267	1948
	26	1	6055+	6365+	5	94	327	5	299	1948
	28	1	1230+	4396+	5	509	1004	5	606	1133
	30	0	-	-	2	1916+	5391+	5	336	5401
	32	1	3932+	5828+	2	703+	4325+	2	704+	4325+
3	10	5	0.0	0.0	5	0.1	0.1	5	0.1	0.1
	12	5	0.1	0.1	5	0.4	0.5	5	0.1	0.1
	14	5	0.3	0.3	5	0.3	0.3	5	0.2	0.2
	16	5	8.6	13	5	2.5	2.7	5	0.8	0.8
	18	5	204	548	5	6.2	7.8	5	1.4	1.4
	20	4	326+	1712+	5	1.1	1.2	5	0.5	0.5
	22	3	1458+	3679+	5	6.2	7.5	5	2.4	2.6
	24	2	1941+	4438+	5	8.8	15	5	5.1	5.7
	26	0	-	-	5	119	191	5	62	98
	28	2	4035+	5254+	5	89	271	5	85	209
	30	0	-	-	4	612+	2356+	5	498	1856
	32	0	-	-	2	3091+	4666+	2	3350+	4751+
4	10	5	0.0	0.0	5	0.0	0.0	5	0.0	0.0
	12	5	0.1	0.1	5	0.1	0.1	5	0.1	0.1
	14	5	0.3	0.3	5	0.9	1.0	5	0.3	0.3
	16	5	1.0	1.0	5	0.4	0.4	5	0.1	0.1
	18	5	15	36	5	1.6	1.7	5	0.4	0.4
	20	5	109	522	5	1.1	1.1	5	0.3	0.3
	22	5	206	811	5	4.5	8.2	5	1.0	1.1
	24	1	5918+	6300+	5	16	23	5	6.8	9.1
	26	0	-	-	5	16	19	5	7.0	7.4
	28	1	3184+	5783+	5	19	36	5	11	11
	30	0	-	-	5	89	430	5	34	61
	32	0	-	-	5	286	679	5	206	478

[1]Shifted geometric mean [2]Average
+Computation terminated after 7200 sec for instances not solved to optimality.

all 20 of the instances with 35 jobs or more, while MIP solved only 5 of them within a two-hour time limit.

The results show that strengthened cuts can make a significant difference when there are 3 or more facilities, but not enough to explain most of the discrepancy described above in the LBBD results. This suggests that the discrepancy is primarily due to SCIP's relatively slow solution of CP subproblems.

4 Conclusions

We conclude that despite improvements in MIP software, logic-based Benders decomposition results in significantly faster solution of the problem class studied

Table 3. Computational results for planning and scheduling "e" instances.

Size		MIP (CPLEX)			LBBD: Weak cuts			LBBD: Strong cuts		
m	n	Solved	Sec[1]	Sec[2]	Solved	Sec[1]	Sec[2]	Solved	Sec[1]	Sec[2]
2	10	5	0.1	0.1	5	0.1	0.1	5	0.1	0.1
2	12	5	0.3	0.3	5	0.3	0.3	5	0.1	0.1
3	15	5	0.9	0.9	5	0.4	0.4	5	0.2	0.2
4	20	5	20	46	5	6.2	14	5	1.6	1.9
5	25	5	25	73	5	1.0	1.0	5	0.7	0.7
6	30	5	113	543	5	1.3	1.3	5	0.4	0.4
7	35	2	2956+	5122+	5	25	36	5	2.5	2.7
8	40	2	909+	4357+	4	130+	1527+	5	18	80
9	45	0	-	-	5	512	1050	5	29	35
10	50	1	6968+	6983+	5	22	45	5	5.0	5.4

[1]Shifted geometric mean [2]Average
+Computation terminated after 7200 sec for instances not solved to optimality.

here, generally with speedups of at least 100 to 1000 on larger instances. It is also superior to the other MIP/CP hybrid methods reported in [11] and shown in Table 1. The advantage of LBBD increases as the number of facilities increases, because it is a decomposition method that is naturally more effective when the problem decomposes into smaller subproblems.

Heinz and Beck [11] remark that even when MIP is slower than LBBD at proving optimality, it is more effective at finding good feasible solutions. This because LBBD does not find a feasible solution until it terminates with a proof of optimality.

In response, we first point out that this is a technical peculiarity of the minimum cost problem solved here, in which the Benders subproblem is a feasibility problem. It is not the case for other objective functions—such as makespan, number of late jobs, and total tardiness—for which the subproblem is an optimization problem that finds feasible solutions throughout the solution process [16, 17, 19].

More importantly, LBBD is proposed as an exact method, rather than a heuristic method for finding feasible solutions. The tests show that LBBD clearly excels at finding optimal solutions. If the goal is to find good feasible solutions, one can use primal heuristics. In fact, this is precisely the approach taken by the MIP solvers tested by [11]. LBBD can likewise find good feasible solutions if it is permitted to use a primal heuristic. However, in the tests reported in [11], only the MIP and MIP/CP hybrid methods are allowed to use primal heuristics.

Heinz and Beck also remark that unlike MIP, the decomposition approach has difficulty with constraints that couple the scheduling subproblems, such as precedence constraints. Decomposition in fact requires that the subproblems decouple and would presumably be less effective if the subproblems were linked. However, this is balanced by a weakness of the MIP model, which uses time-indexed variables. As the time horizon becomes longer, the number of integer variables increases, and the MIP problem becomes disproportionately harder. This is demonstrated in [7] even for a single-facility problem, in which case a LBBD method that decomposes the time horizon into short segments is far superior to MIP.

The more general issue is whether it is reasonable to pursue such special-purpose methods as LBBD when MIP software constantly improves. LBBD seems worthy of pursuit, for four reasons. One reason is that LBBD technology developed before 2007 is still far superior to the latest MIP solvers, at least on the problem class studied here. A second reason is that LBBD can also improve. Logic-based Benders cuts are more effective when they exploit information from the inference dual of the subproblem [21, 20]. This information is not available from the CP solver in pre-2007 implementations. An implementation that fully integrates the master and subproblem may be signficantly more effective.

A third reason is that LBBD is best conceived as an *enhancement* of MIP rather than as a *competitor* of MIP. After all, LBBD relies on MIP to solve the master problem, and conceivably the subproblems as well. MIP technology will doubtless continue to improve, but these improvements can be incorporated into LBBD. The relevant issue is whether the effectiveness of MIP can be increased *still more* by decomposing the problem and applying MIP to some or all of its components. So far, the answer appears to be yes.

Finally, LBBD need not be a "special-purpose" method, but can be part of an integrated approach to optimization. For example, LBBD has been implemented within the general-purpose solver SIMPL [1], which obtained the results in [28] mentioned earlier. SIMPL treats both the Benders method and MIP as special cases of a restrict-infer-and-relax algorithm. Rather than rely solely on engineering improvements in MIP solvers, it seems best to invest similar effort in integrated solvers, such as SCIP and SIMPL.

References

1. I. Aron, J. N. Hooker, and T. H. Yunes. SIMPL: A system for integrating optimization techniques. In J. C. Régin and M. Rueher, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2004)*, volume 3011 of *Lecture Notes in Computer Science*, pages 21–36. Springer, 2004.
2. J. C. Beck and Wen-Yang Ku. Personal communication, 14 September 2012.
3. J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.
4. L. Benini, D. Bertozzi, A. Guerri, and M. Milano. Allocation and scheduling for MPSoCs via decomposition and no-good generation. In *Principles and Practice of Constraint Programming (CP 2005)*, volume 3709 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2005.
5. H. Cambazard, P.-E. Hladik, A.-M. Déplanche, N. Jussien, and Y. Trinquet. Decomposition and learning for a hard real time task allocation problem. In M. Wallace, editor, *Principles and Practice of Constraint Programming (CP 2004)*, volume 3258 of *Lecture Notes in Computer Science*, pages 153–167. Springer, 2004.
6. Y. Chu and Q. Xia. Generating Benders cuts for a class of integer programming problems. In J. C. Régin and M. Rueher, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2004)*, volume 3011 of *Lecture Notes in Computer Science*, pages 127–141. Springer, 2004.

7. E. Coban and J. N. Hooker. Single-facility scheduling over long time horizons by logic-based benders decomposition. In A. Lodi, M. Milano, and P. Toth, editors, *Proceedings of the International Workshop on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming for Combintaorial Optimization Problems (CPAIOR 2010)*, pages 87–91. Springer, 2010.
8. A. I. Corréa, A. Langevin, and L. M. Rousseau. Dispatching and conflict-free routing of automated guided vehicles: A hybrid approach combining constraint programming and mixed integer programming. In J. C. Régin and M. Rueher, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2004)*, volume 3011 of *Lecture Notes in Computer Science*, pages 370–378. Springer, 2004.
9. I. Harjunkoski and I. E. Grossmann. A decomposition approach for the scheduling of a steel plant production. *Computers and Chemical Engineering*, 25:1647–1660, 2001.
10. I. Harjunkoski and I. E. Grossmann. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Computers and Chemical Engineering*, 26:1533–1552, 2002.
11. S. Heinz and J. C. Beck. Reconsidering mixed integer programming and MIP-based hybrids for scheduling. In N. Jussien and T. Petit, editors, *Proceedings of the International Workshop on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming for Combintaorial Optimization Problems (CPAIOR 2012)*, volume 7298 of *Lecture Notes in Computer Science*, pages 211–227. Springer, 2012.
12. J. N. Hooker. *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. Wiley, New York, 2000.
13. J. N. Hooker. A hybrid method for planning and scheduling. In M. Wallace, editor, *Principles and Practice of Constraint Programming (CP 2004)*, volume 3258 of *Lecture Notes in Computer Science*, pages 305–316. Springer, 2004.
14. J. N. Hooker. Planning and scheduling problem instances (with documentation), website. web.tepper.cmu.edu/jnh/instances.htm, 2004.
15. J. N. Hooker. A hybrid method for planning and scheduling. *Constraints*, 10:385–401, 2005.
16. J. N. Hooker. Planning and scheduling to minimize tardiness. In *Principles and Practice of Constraint Programming (CP 2005)*, volume 3709 of *Lecture Notes in Computer Science*, pages 314–327. Springer, 2005.
17. J. N. Hooker. An integrated method for planning and scheduling to minimize tardiness. *Constraints*, 11:139–157, 2006.
18. J. N. Hooker. *Integrated Methods for Optimization*. Springer, 2007.
19. J. N. Hooker. Planning and scheduling by logic-based Benders decomposition. *Operations Research*, 55:588–602, 2007.
20. J. N. Hooker. *Integrated Methods for Optimization, 2nd ed.* Springer, 2012.
21. J. N. Hooker and G. Ottosson. Logic-based Benders decomposition. *Mathematical Programming*, 96:33–60, 2003.
22. J. N. Hooker and H. Yan. Logic circuit verification by Benders decomposition. In V. Saraswat and P. Van Hentenryck, editors, *Principles and Practice of Constraint Programming: The Newport Papers*, pages 267–288, Cambridge, MA, 1995. MIT Press.
23. V. Jain and I. E. Grossmann. Algorithms for hybrid MILP/CP models for a class of optimization problems. *INFORMS Journal on Computing*, 13:258–276, 2001.

24. C. T. Maravelias and I. E. Grossmann. Using MILP and CP for the scheduling of batch chemical processes. In J. C. Régin and M. Rueher, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2004)*, volume 3011 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2004.
25. D. Terekhov, J. C. Beck, and K. N. Brown. Solving a stochastic queueing design and control problem with constraint programming. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI 2005)*, pages 261–266, 2005.
26. E. Thorsteinsson. Branch and check: A hybrid framework integrating mixed integer programming and constraint logic programming. In T. Walsh, editor, *Principles and Practice of Constraint Programming (CP 2001)*, volume 2239 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 2001.
27. C. Timpe. Solving planning and scheduling problems with combined integer and constraint programming. *OR Spectrum*, 24:431–448, 2002.
28. T. H. Yunes, I. Aron, and J. N. Hooker. An integrated solver for optimization problems. *Operations Research*, 58:342–356, 2010.