

Global optimization of nonconvex bilevel problems: implementation and computational study of the Branch-and-Sandwich algorithm

Remigijus Paulavičius^a, Polyxeni-M. Kleniati^{a,b} and Claire S. Adjiman^a

^a*Department of Chemical Engineering, Centre for Process Systems Engineering, Imperial College London, London SW7 2AZ, UK*

^b*Current address: Process Systems Enterprise, 26-28 Hammersmith Grove, London W6 7HA, UK
c.adjiman@imperial.ac.uk*

Abstract

We describe BASBL, an implementation of the Branch-and-Sandwich deterministic global optimization algorithm, (Kleniati and Adjiman, J. Glob. Opt. 60, 425–458, 2014), for nonlinear bilevel problems, within the MINOTAUR toolkit. The algorithm is extended to include heuristics for branching and node selection. The computational performance of the bilevel solver is analyzed for different combinations of the heuristics by solving nonconvex bilevel test problems from the literature.

Keywords: Nonconvex bilevel optimization, Deterministic global optimization, Branch-and-bound, Algorithm implementation

1. Introduction

Mathematical programming, and optimization in general, have become essential tools in process systems engineering, and have found applications in synthesis, design, control, operations and planning (Biegler and Grossmann, 2004). Many of these optimization problems are bilevel problems in nature (Grossmann and Biegler, 2004) and can thus be framed as two-person, hierarchical optimization problems having a second optimization problem as part of the constraints. This includes, for instance, optimization problems in design or parameter estimation in which phase equilibrium is present in the constraints, or problems in which the optimum economic design is sought while minimizing energy consumption. Special cases of bilevel problems, such as problems of linear (quadratic) – linear (quadratic) type, have been studied extensively and many algorithms have been proposed in the literature (Dempe, 2002; Floudas and Gounaris, 2009). However, the general nonconvex form is very challenging and only recently the first deterministic method was proposed by Mitsos et al. (2008) and the approximation method introduced by Tsoukalas et al. (2009).

In Kleniati and Adjiman (2014a), a new deterministic global optimization algorithm, Branch-and-Sandwich (B&S), was proposed for general nonlinear optimistic bilevel programming problems (BPP) that satisfy a regularity condition in the inner problem:

$$\min_{\mathbf{x} \in X, \mathbf{y}} \{F(\mathbf{x}, \mathbf{y}) \text{ s.t. } G(\mathbf{x}, \mathbf{y}) \leq 0, \mathbf{y} \in \arg \min_{\mathbf{y} \in Y} \{f(\mathbf{x}, \mathbf{y}) \text{ s.t. } g(\mathbf{x}, \mathbf{y}) \leq 0\}\}, \quad (\text{BPP})$$

where the n -dimensional vector $\mathbf{x}$ denotes the outer (leader) variables and the m -dimensional vector $\mathbf{y}$ denotes the inner (follower) variables. The functions $F, f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$, $G : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^p$ and $g : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^q$ denote the outer/inner objective and outer/inner constraint functions and the host sets $X = [\mathbf{x}^L, \mathbf{x}^U] \subset \mathbb{R}^n$ and $Y = [\mathbf{y}^L, \mathbf{y}^U] \subset \mathbb{R}^m$ are compact. Neither a special class nor convexity of the functions involved is assumed. The optimistic (or co-operative, weak) formulation (Dempe, 2002) implies that if for a given $\mathbf{x}$ the inner (follower) program has multiple globally optimal solutions $\mathbf{y}$ to which the follower is indifferent, the outer (leader) optimizer can choose among them. Hence the outer minimization in (BPP) is with respect to the whole set of variables.

The B&S approach makes use of an equivalent formulation of (BPP) (Dempe, 2002):

$$\min_{\mathbf{x} \in X, \mathbf{y} \in Y} \{F(\mathbf{x}, \mathbf{y}) \text{ s.t. } G(\mathbf{x}, \mathbf{y}) \leq 0, g(\mathbf{x}, \mathbf{y}) \leq 0, f(\mathbf{x}, \mathbf{y}) \leq w(\mathbf{x})\}, \quad (\text{eBPP})$$

where $w(\mathbf{x})$ is the optimal value function (OVF) of the inner problem:

$$w(\mathbf{x}) = \min_{\mathbf{y}} \{f(\mathbf{x}, \mathbf{y}) \text{ s.t. } g(\mathbf{x}, \mathbf{y}) \leq 0, \mathbf{y} \in Y\}. \quad (\text{OVF})$$

Part of the B&S method is based on the observation that by reformulation (eBPP) and particularly by the constraint $f(\mathbf{x}, \mathbf{y}) \leq w(\mathbf{x})$, a restriction of problem (OVF), i.e., an upper bound on $w(\mathbf{x})$, yields a relaxation of (eBPP) and consequently, a relaxation of (BPP), and that, similarly, a relaxation of (OVF) gives a restriction of (BPP) (Mitsos et al., 2008). The convergence properties of the B&S algorithm and promising numerical results were reported in Kleniati and Adjiman (2014b). The main focus of the current paper is on the implementation of the B&S algorithm within the MINOTAUR toolkit (Mahajan et al., 2011) and on investigation of the computational performance with respect to branching and node selection strategies.

The paper is organized as follows. Basic concepts of B&S and notations are introduced in Section 2. Alternative strategies for branching and for the management of nodes follow in Section 3. Numerical results and an implementation of B&S in MINOTAUR are presented in Section 4. Conclusions are drawn in Section 5.

2. Basic concepts and notation

The main aspects of the B&S algorithm discussed in this paper are related to branching and node selection, therefore only concepts and definitions relevant to these are presented here. A more detailed description can be found in Kleniati and Adjiman (2014a). A node k in the Branch-and-Sandwich tree represents (sub)domain $X^{(k)} \times Y^{(k)} \subseteq X \times Y$. The root node ($k = 1$) corresponds to the whole domain $X \times Y$. Each node k has the following attributes (properties): inner lower ($\underline{f}^{(k)}$), inner upper ($\bar{f}^{(k)}$) and outer lower ($\underline{F}^{(k)}$) and upper ($\bar{F}^{(k)}$) bound values and the level (depth) ($l^{(k)}$) in the Branch-and-Sandwich search tree; for the root node $l^{(1)} = 0$. Based on the bounds computed at a node k , we may decide to fully fathom or “outer-fathom” the node (cf. Sect. 4.6 in Kleniati and Adjiman (2014a)). If the node is not fully fathomed, then we need to continue exploring it further via branching. The replacement of the parent node k by new child nodes in the Branch-and-Sandwich search tree is managed using appropriate lists of nodes. The list $\mathcal{L}$ corresponds to the classical list of active nodes, i.e., the nodes that require further branching. The list $\mathcal{L}_{\text{in}}$ is the list of *inner-open* (or *outer-fathomed*) nodes, i.e., nodes that should be explored further with respect to the inner problem only, but which are known not to contain the global solution of (BPP). In addition to lists $\mathcal{L}$ and $\mathcal{L}_{\text{in}}$, auxiliary (independent) lists $\mathcal{L}^p, p \in P$ are used. These lists consist of a collection of *sublists* containing nodes with overlapping X subdomains but non-overlapping Y subdomains:

$$\mathcal{L}^p = \{\mathcal{L}_1^p, \dots, \mathcal{L}_{s^p}^p\}.$$

Remark 1 A node k that has not yet been fully fathomed belongs either to list $\mathcal{L}$ or to list $\mathcal{L}_{\text{In}}$. It must also belong to an independent list $\mathcal{L}^p$ for some $p \in P$, i.e., $k \in (\mathcal{L} \cup \mathcal{L}_{\text{In}}) \cap \mathcal{L}^p$.

3. Node selection and branching heuristics

The branching scheme introduced in the original B&S algorithm allows branching on the outer and inner variables. To branch at the “most promising” nodes in $\mathcal{L} \cup \mathcal{L}_{\text{In}}$ the B&S algorithm uses information from the outer and inner levels and chooses at least one node: one from the list of open nodes $\mathcal{L}$ and one from the list of inner open nodes $\mathcal{L}_{\text{In}}$, if non empty. Definition 1 states four variants of the node selection operation:

Definition 1 (Node selection heuristics) *The two-step node selection procedure is:*

Step 1 find the independent list $\mathcal{L}^p, p \in P$, containing a node $k \in \mathcal{L}^p \cap \mathcal{L}$ with the lowest lower bound ($\underline{F}^{(k)}$) and, if several nodes with the smallest lower bound exist, the smallest level ($l^{(k)}$):

$$k = \arg \min_i \{l^{(i)} : i = \arg \min_{j \in \mathcal{L}} \{\underline{F}^{(j)}\}\}; \quad (\text{Best-}\underline{F}l)$$

alternatively, find $\mathcal{L}^p$ with a node having the smallest level ($l^{(k)}$) and, if several nodes with the smallest level exist, the lowest lower bound ($\underline{F}^{(k)}$):

$$k = \arg \min_i \{\underline{F}^{(i)} : i = \arg \min_{j \in \mathcal{L}} \{l^{(j)}\}\}; \quad (\text{Best-}l\underline{F})$$

Step 2 select a node $k \in \mathcal{L} \cap \mathcal{L}^p$ and a node $k_{\text{In}} \in \mathcal{L}_{\text{In}} \cap \mathcal{L}^p$, if non empty, with the smallest level ($l^{(k)}$) and, if several nodes have the same smallest level, the smallest inner lower bound ($\underline{f}^{(k)}$):

$$k = \arg \min_i \{\underline{f}^{(i)} : i = \arg \min_{j \in \mathcal{L} \cap \mathcal{L}^p} \{l^{(j)}\}\} \text{ and } k_{\text{In}} = \arg \min_i \{\underline{f}^{(i)} : i = \arg \min_{j \in \mathcal{L}_{\text{In}} \cap \mathcal{L}^p} \{l^{(j)}\}\}; \quad (\text{Best-}l\underline{f})$$

alternatively, select nodes k and k_{In} with the smallest level ($l^{(k)}$) and, if several nodes with the lowest level exist, the smallest inner upper bound ($\bar{f}^{(k)}$) value:

$$k = \arg \min_i \{\bar{f}^{(i)} : i = \arg \min_{j \in \mathcal{L} \cap \mathcal{L}^p} \{l^{(j)}\}\} \text{ and } k_{\text{In}} = \arg \min_i \{\bar{f}^{(i)} : i = \arg \min_{j \in \mathcal{L}_{\text{In}} \cap \mathcal{L}^p} \{l^{(j)}\}\}. \quad (\text{Best-}l\bar{f})$$

The original B&S node selection procedure described in Kleniati and Adjiman (2014a) is very similar to the Best- $\underline{F}l$ -Best- $l\underline{f}$ (in short, $\underline{F}l$ - $l\underline{f}$) heuristic. This heuristic implies that at Step 1 preference is given to the independent list containing the node(s) with the best (lowest) outer lower bound value $\underline{F}$ (i.e., the best candidate from the outer problem perspective) and at Step 2, preference is given to nodes at the “highest” level in the selected list (i.e., smallest l). If several nodes are at the same “highest” level, the node with the lowest inner lower bound $\underline{f}$ is chosen (i.e., the best candidate from the inner problem perspective). Moreover, in the current B&S implementation we resolve an ambiguity which arises at Step 1, when several nodes with the same lowest outer lower bound value exist. In this case, the node with the smallest l value is chosen. The $(l\bar{f})$ heuristic is an alternative strategy in which, at Step 2, the node with the lowest inner upper bound value $\bar{f}$ is chosen if several nodes with the same smallest l exist. In this way we select the best candidate based on the outer problem perspective, as tighter inner upper bound values lead to tighter outer lower bound values

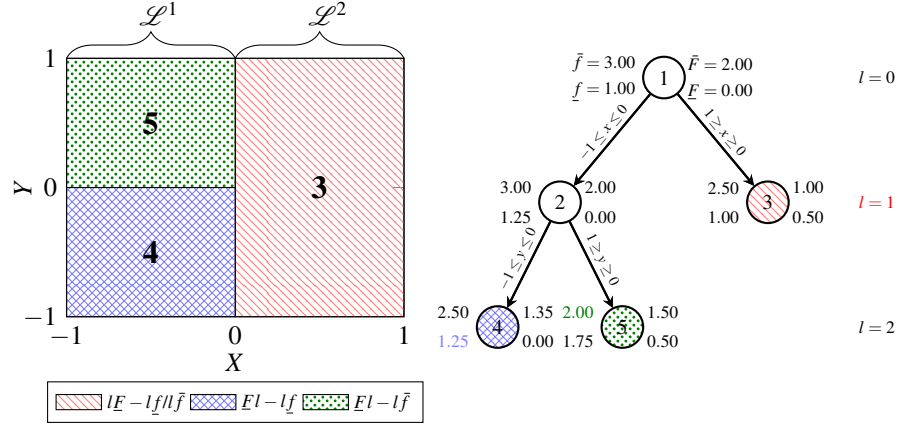


Figure 1: Illustration of selected nodes by using different node selection heuristics.

(cf. Sect. 4.5 in Kleniati and Adjiman (2014a)) and potentially faster outer-fathoming. The remaining two heuristics at Step 1 select the independent list with the best node based on l and $\underline{F}$ values.

For instance, consider the partitioning of a two-dimensional region $X \times Y = [-1, 1] \times [-1, 1]$ with the Branch-and-Sandwich tree shown in Fig. 1. Suppose that, at the current iteration, all nodes belong only to the (outer) active list, i.e., $\mathcal{L} = \{3, 4, 5\}$ and $\mathcal{L}_{\text{in}} = \emptyset$. There are two independent lists with one sublist each: $\mathcal{L}^1 = \mathcal{L}_1^1 = \{4, 5\}$ and $\mathcal{L}^2 = \mathcal{L}_1^2 = \{3\}$. At Step 1, using the (lF) node selection heuristic, the B&S algorithm selects list $\mathcal{L}^2$ with the node $k = 3$ having the smallest level $l^{(3)} = 1$. Node 3 is the sole candidate in $\mathcal{L}^2$, therefore at Step 2, B&S selects this node. However, when the $(\text{Best-}\underline{F}l)$ selection heuristic is used at Step 1, B&S selects list $\mathcal{L}^1$ as it contains node 4 with the smallest lower bound $\underline{F}^{(4)} = 0.00$. At Step 2, the two inner-level information based heuristics return different candidates, as $l^{(4)} = l^{(5)} = 2$ but $\underline{f}^{(4)} < \underline{f}^{(5)}$ and $\bar{f}^{(4)} > \bar{f}^{(5)}$.

Once a node k is chosen, a bisection on the midpoint of the variable with the largest normalized range is used. If several variable satisfy this requirement, priority is given to the variable with the lower/higher variable index (LVI/HVI).

4. Implementation and numerical results

In this section we describe an implementation of the Branch-And-Sandwich BiLevel algorithm – BASBL solver. BASBL is implemented in C++ within MINOTAUR (Mixed-Integer Nonlinear Optimization Toolkit – Algorithms, Underestimators, Relaxations) (Mahajan et al., 2011) which provides interfaces for classical branch-and-bound algorithms and the AMPL modelling language (Fourer et al., 2002). However, several modifications are needed because, in the Branch-and-Sandwich tree, additionally to the usual list of open nodes, we use the list of inner-open nodes together with independent lists and sublists. Moreover, every node in the Branch-and-Sandwich tree must hold lower and upper bounds for both the inner and outer problems. We have therefore extended the MINOTAUR tree and node management facilities and adapted them to fit the B&S requirements. Inputs to the BASBL solver are the outer and inner optimality tolerances ε_F and ε_f respectively, the node selection rule, the branching variable strategy and the maximum number of Branch-and-Sandwich iterations. Then if explicit bounds are known for all variables, the participating functions are

continuous and a constraint qualification holds for the inner problem for all $\mathbf{x}$ values, the BASBL solver returns the optimal value F^{UB} and the $(\mathbf{x}^{UB}, \mathbf{y}^{UB})$ point corresponding to a global ε -optimal solution (Kleniati and Adjiman, 2014a).

To test the current implementation, we used the 10 most challenging bilevel instances from the test set used in Kleniati and Adjiman (2014b) and Mitsos et al. (2008). These bilevel instances have either nonconvex inner problems or structure that caused slow convergence in earlier works. All bilevel optimization problems were written in the AMPL modelling language format (.mod) files and read through the AMPL interface from (.nl) files. To solve such problems globally, we have implemented an interface with GAMS, since available solvers in MINOTAUR can only ensure local optimality for NLP problems. All bounding problems were solved to global optimality with the Branch-And-Reduce Optimization Navigator (BARON) version 14.4.0 (Tawarmalani and Sahinidis, 2005) using GAMS version 24.4.1 (GAMS Development Corporation, 2014) through system calls. The optimality and feasibility tolerances for BARON were set to $\varepsilon_{NLP} = 10^{-6}$ for all problems. The outer and inner objective tolerances were set to $\varepsilon_F = 10^{-3}$ and $\varepsilon_f = 10^{-5}$ for all problem instances except for mb_1.1_06 and mb_1.1_11 where $\varepsilon_F = 0.1$ was used. All experiments were performed on a computer with 64-bit Intel(R) Xeon(R) CPU E5-2667 v2 @ 3.30GHz processor running Linux Ubuntu 14.04.3 LTS.

All 10 problems were solved with the BASBL solver using 8 combinations of algorithmic options and the resulting number of iterations (*Iter*), total number of inner and outer (sub)problems (#P) and CPU time (*t(s)*) in each case are reported in Table 1 for four of the combinations. This comparison does not include the (*IF*) node selection based variants, as it does not affect the performance of BASBL for this test set. The first column lists the problem label, as given in Mitsos and Barton (2006). In the next three columns, the results for the combination closest to the original B&S were used: the highest variable index (HVI) branching strategy and $\underline{Fl-lf}$ node selection heuristic. The other groups of three columns are based on changing one of the options at a time, as indicated by the headings in the top row.

Let us first observe that the mb_1.1_06 and mb_1.1_11 bilevel instances are the most challenging for the BASBL solver and caused the slowest converge. This is because their convergence is dependent on eliminating KKT inner suboptimal points that cause slow tightening of outer lower bound values and consequently slow outer-fathoming of certain nodes. Comparing the different branching variable selection rules (HVI vs LVI), we can observe that branching more on $\mathbf{x}$ -variables (LVI strategy) reduces significantly the number of subproblems (#P) and execution time for these two instances. This is possibly because the LVI bisection strategy leads to a higher number of independent lists but a lower number of nodes inside these lists, which means a smaller number of inner subproblems are needed for the outer upper bounding procedure (cf. Sect. 4.7 in Kleniati and Adjiman (2014a)). On the other hand for easier bilevel instances, the HVI strategy often gives a better performance. A possible reason is that a smaller number of lists and, a higher number of nodes inside them leads to a better (lower) best inner upper bound value (see Definition 7 in Kleniati and Adjiman (2014a)) and faster full-fathoming of the nodes (cf. Sect. 4.6 in Kleniati and Adjiman (2014a)). Finally, there is no advantage in choosing one node selection strategy over the other ($\underline{Fl-lf}$ vs $\underline{Fl-l\hat{f}}$) for this small test set, although reductions of around a few percent in CPU times on average and a maximum reduction of 12% can be observed.

5. Conclusions

In this paper, BASBL, a solver based on the Branch-and-Sandwich algorithm and embedded within the MINOTAUR toolkit, was discussed. Alternative choices for branching and node selection were examined. Numerical comparisons revealed that a notable performance improvement may be achieved by selecting

Table 1: The number of iterations, total number of solved subproblems and total CPU time in seconds based on different variants of the BASBL solver. See text for an explanation of the algorithmic options

Label	HVI- $\underline{Fl-lf}$			HVI- $\underline{Fl-l\tilde{f}}$			LVI- $\underline{Fl-lf}$			LVI- $\underline{Fl-l\tilde{f}}$		
	<i>Iter</i>	#P	<i>t(s)</i>	<i>Iter</i>	#P	<i>t(s)</i>	<i>Iter</i>	#P	<i>t(s)</i>	<i>Iter</i>	#P	<i>t(s)</i>
mb_1.1.05	5	47	2.35	5	47	2.46	8	72	4.09	8	72	4.07
mb_1.1.06	130	3627	655	131	3638	669	88	1827	248	89	1843	253
mb_1.1.06v	7	99	5.20	7	99	5.25	4	49	2.55	4	49	2.57
mb_1.1.07	7	69	3.27	8	76	3.73	11	106	5.43	11	97	5.21
mb_1.1.08	4	37	1.79	4	37	1.83	6	56	2.82	6	56	2.91
mb_1.1.09	7	82	3.84	7	82	3.91	4	39	1.82	4	39	1.85
mb_1.1.10	5	48	2.46	7	48	2.48	8	71	3.98	8	71	4.12
mb_1.1.11	36	567	43.7	34	520	38.2	26	344	22.9	26	344	23.3
mb_1.1.13	7	65	3.33	7	65	3.35	11	103	5.82	11	91	5.40
mb_1.1.14	3	31	1.88	3	31	1.89	2	19	1.23	2	19	1.22
Average	21.1	467.2	72.3	21.3	464.3	73.2	16.8	268.6	29.9	16.9	268.1	30.4

specific branching rules. However, the full potential of BASBL will be drawn after examining its performance on a larger suite of bilevel problems and by investigating different bounding approaches and alternative node management.

Acknowledgements We gratefully acknowledge funding from the Leverhulme Trust through the Philip Leverhulme Prize and from the EPSRC through a Leadership Fellowship [EP/J003840/1].

Data access statement Data underlying this article can be accessed on Zenodo at <http://dx.doi.org/10.5281/zenodo.44997>, and used under the Creative Commons Attribution licence.

References

- L. T. Biegler, I. E. Grossmann, 2004. Retrospective on optimization. *Computers & Chemical Engineering* 28 (8), 1169–1192.
- S. Dempe, 2002. *Foundations of Bilevel Programming*. Springer Science & Business Media, Boston.
- C. A. Floudas, C. E. Gounaris, 2009. A review of recent advances in global optimization. *Journal of Global Optimization* 45 (1), 3–38.
- R. Fourer, D. M. Gay, B. W. Kernighan, 2002. *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press.
- GAMS Development Corporation, 2014. *General Algebraic Modeling System (GAMS) Release 24.4.1*. Washington, DC, USA.
- I. E. Grossmann, L. T. Biegler, 2004. Part II. Future perspective on optimization. *Computers & Chemical Engineering* 28 (8), 1193–1218.
- P.-M. Kleniati, C. S. Adjiman, 2014a. Branch-and-Sandwich: a deterministic global optimization algorithm for optimistic bilevel programming problems. Part I: Theoretical development. *Journal of Global Optimization* 60 (3), 425–458.
- P.-M. Kleniati, C. S. Adjiman, 2014b. Branch-and-Sandwich: a deterministic global optimization algorithm for optimistic bilevel programming problems. Part II: Convergence analysis and numerical results. *Journal of Global Optimization* 60 (3), 459–481.
- A. Mahajan, S. Leyffer, J. Linderoth, J. Luedtke, T. Munson, 2011. MINOTAUR: a toolkit for solving mixed-integer nonlinear optimization. Wiki-page. <http://wiki.mcs.anl.gov/minotaur>.
- A. Mitsos, P. I. Barton, 2006. *A Test Set for Bilevel Programs*. Technical report. Massachusetts Institute of Technology.
- A. Mitsos, P. Lemonidis, P. I. Barton, 2008. Global solution of bilevel programs with a nonconvex inner program. *Journal of Global Optimization* 42 (4), 475–513.
- M. Tawarmalani, N. V. Sahinidis, 2005. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming* 103 (2), 225–249.
- A. Tsoukalas, B. Rustem, E. Pistikopoulos, 2009. A global optimization algorithm for generalized semi-infinite, continuous minimax with coupled constraints and bi-level problems. *Journal of Global Optimization* 44 (2), 235–250.