8-2014

Computational strategies for improved MINLP algorithms

Lijie Su
Northeastern University, China

Lixin Tang
Northeastern University, China

Ignacio E. Grossmann
Carnegie Mellon University, grossmann@cmu.edu

Published In
Computers and Chemical Engineering, 75, 40-48.

Computational Strategies for Improved MINLP Algorithms

Lijie Su[1]，Lixin Tang[1*]，Ignacio E. Grossmann[2]

[1] The Logistics Institute, Northeastern University, Shenyang 110819, Liaoning, P. R. China

[2] Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh 15213, PA, USA

Abstract： In order to improve the efficiency for solving MINLP problems, we present in this paper three computational strategies. These include multiple-generation cuts, hybrid methods and partial surrogate cuts for the Outer Approximation and Generalized Benders Decomposition. The properties and convergence of the strategies are analyzed. Five new MINLP algorithms are described based on the proposed strategies, and their implementation is discussed. Results of numerical experiments are reported for benchmark MINLP problems to demonstrate the efficiency of the proposed methods.

Key words: Mixed Integer Nonlinear Programming(MINLP), Outer Approximation(OA), Generalized Benders Decomposition(GBD), Cutting Plane.

1. Introduction

MINLP is the most general mathematical programming model including discrete and continuous variables, constraints with linear and nonlinear functions. Its applications are common in Process System Engineering(PSE), such as process synthesis, planning and scheduling, blending problems, and enterprise-wide optimization[1-15]. Most common methods for MINLP include Branch and Bound(B&B), Outer Approximation(OA), Generalized Benders Decomposition(GBD),

*Correspondence author. E-mail address: Lixintang@mail.neu.edu.cn (Lixin Tang)

Branch and Cut(B&C), Extended Cutting Plane Method(ECP). Reviews can be found in Grossmann, Belotti and Bonami[16-18]. There are several typical characteristics in process industries different from discrete manufacture industries, such as complex process networks, reaction mechanisms, fluid material flow that often involves nonlinear relationships among the flow variables. This has motivated the development of MINLP methods.

The paper is organized as follows. In section 2, we review the Outer-Approximation algorithm and Generalized Benders Decomposition. The advantages and disadvantages of these algorithms are summarized in order to introduce the motivation of our work. In section 3, three core improved strategies are presented in detail with theoretical analysis for their optimal properties and convergence. In section 4, five algorithms are described based on the strategies described in section 3. In section 5, we implement the proposed algorithms on benchmark MINLP test problems, and compare the performances with basic OA and GBD. Finally, we draw conclusions based on the performance of the proposed methods.

2. Background

MINLP models are usually classified into convex and nonconvex according to the convexity/nonconvexity of continuous model with relaxed discrete variables. According to separability of discrete and continuous variables, separable and nonseparable MINLP are also defined. This paper focuses on solving convex MINLP problems.

The general form of a convex MINLP model is as follows.

$$
\begin{aligned}
&\min_{x,y} \quad z = f(x,y) \\
&s.t. \quad g(\mathrm{x},\mathrm{y}) \le 0 \\
&\qquad Ax + Ey \le b \\
&\qquad \mathrm{x} \in \mathrm{X} = \left\{ x \middle| x \in R^n, \mathrm{C}\, x \le c, x^L \le x \le x^U \right\} \\
&\qquad y \in Y = \left\{ y \middle| y \in Z^m, Dy \le d \right\}
\end{aligned} \tag{1.1}
$$

where the nonlinear function $f : R^n \to R$ and vector functions $\mathrm{g} : R^n \to R^p$ are assumed to be continuous differentiable and convex on the compact polyhedral convex set X, and Y is a finite discrete/integer variable set. Usually the discrete variables appear in linear form, therefore the MINLP formulation is reduced to formulation (1.2).

$$
\begin{aligned}
&\min_{x,y} \quad z = f(x) + h^T y \\
&s.t. \quad g(\mathrm{x}) + \mathrm{H}\mathrm{y} \le 0 \\
&\qquad Ax + Ey \le b \\
&\qquad \mathrm{x} \in \mathrm{X} = \left\{ x \middle| x \in R^n, \mathrm{C}\, x \le c, x^L \le x \le x^U \right\} \\
&\qquad y \in Y = \left\{ y \middle| y \in Z^m, Dy \le d \right\}
\end{aligned} \tag{1.2}
$$

Major methods for solving MINLP include OA, GBD, B&B, B&C, ECP methods[16]. OA and GBD are two common methods for solving MINLP problems.

OA is a decomposition method that divides an MINLP into an NLP sub-problem with fixed integer variables, and an MILP master problem with first-order approximation of nonlinear functions. From the solution of the NLP subproblem, assuming it is feasible, we obtain an upper bound of the primal problem, while solving the MILP master problem, a lower bound of the primal problem is obtained. The NLP sub-problem and MILP master problem are solved in a cycle of iterations in order to successively update the lower bound and the upper bound of primal MINLP problem, until the relative gap of those bounds lies within a tolerance[19-23].

The theory of OA method is that it first decomposes discrete and continuous variables through a

cutting plane MILP approximation for the search in the discrete variable space, then determines the optimal continuous point for each fixed discrete variable, and finally constructs cutting planes with linear supports on those points from the NLP subproblem that approximate the nonlinear feasible region and objective function.

GBD is derived from Benders Decomposition method for solving MILP, which is also decomposition method[23, 24]. GBD is different from OA in the construction of the master problem. The master problem of GBD is a pseudo-integer programming problem, which is constructed by the y-space projection of Lagrange cuts based on dual information of the NLP subproblem. These cuts can be interpreted as a surrogate of the supporting hyperplanes in OA. As there are no continuous variables in the GBD master problem, the lower bound is weaker than the one of OA. Therefore, the total number of iterations for GBD is larger than in OA. However, GBD requires less CPU time for solving the pseudo-integer programming master problem than OA requires for solving its MILP master problem.

As for the other MINLP algorithms, B&B relies on a tree search in which NLP subproblems are solved at each node [25, 26]. If these NLP subproblems are expensive to solve, the computational requirements of B&B can be very high. The LP/NLP based B&B method is one that involves branch and cut search in which the branch and bound method for the MILP of OA is integrated with NLP subproblems which are normally solved at integer feasible nodes. The LP/NLP based B&B generally requires less computational effort than OA, but it is more difficult to implement[27, 28]. ECP involves the successive solution of MILP master problems generating supporting hyperplanes for the most violated constraints[29]. Since the ECP method converges the nonlinear part with the MILP subproblems, its convergence can be slow. Among recent developments in

the solution of convex MINLP problems, Bonami et al. has proposed a new algorithm based on Branch & Bound and polyhedral outer-approximation in order to solve large scale MINLP problems[30].

The MINLP models from real applications in PSE often include many discrete variables, and complex nonlinearities in the continuous variables. Many discrete variables implies a large combinatorial search space. Complex nonlinearities can lead to nonconvexities making the global optimum solution difficult to obtain. Furthermore, MINLP models can also lead to infeasible NLP subproblems, which in turn can lead to weak cutting planes. Additional specific limitations that arise in OA and GBD are described below.

Although OA is one of the common MINLP algorithms with good performance for many cases, there are some cases while it may have poor performance. For example, when solving the convex MINLP problem CLAY55M with DICOPT [31], the number of iterations and CPU time can be very large as shown in Figure 1, due to the fact that there are many infeasible subproblems. Furthermore, the MILP master problem in OA grows in size as iterations proceed, making the successive solution of the master problem increasingly expensive.

A major limitation of GBD is that the gap between the lower and the upper bounds is often large, which leads to a large number of iterations. An example is given in Figure 2 where solving the MINLP problem CSCHED1 [32], many iterations are required to increase the initial bounds which are very weak.

Also, GBD may fail to converge if there are many discrete variables [19]. If the discrete variables have few constraints in the MINLP, then the NLP subproblems are likely to be infeasible. Furthermore, the GBD infeasibility cuts tend to be weak, so there are almost no changes in the

lower bound. As specific examples, GBD fails to converge in test 4 from Duran& Grossmann[19]and CSCHED4 from minlplib because most of the subproblems are infeasible.

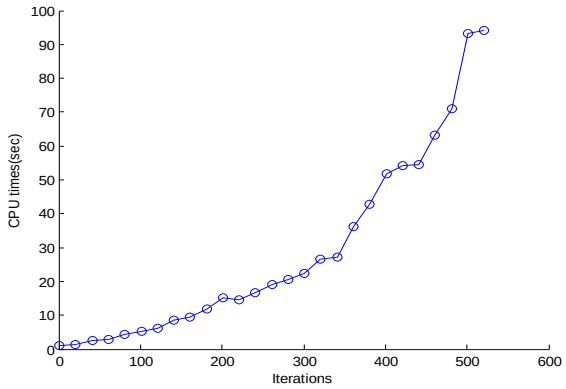

Fig. 1. CPU times of each iteration for CLAY55M with DICOPT

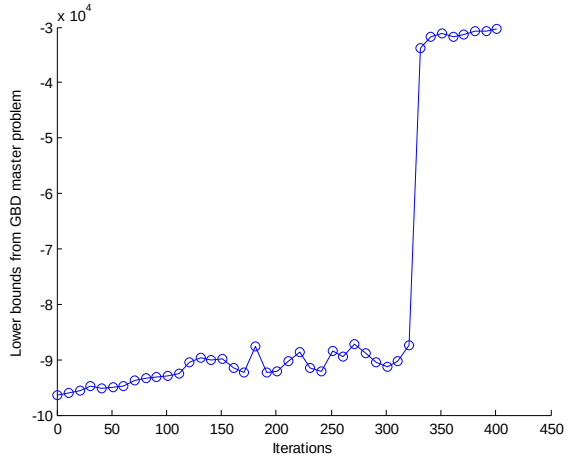

Fig. 2. Lower bounds of CSCHED1 with GBD

Our work is aimed at designing and implementing improved OA and GBD algorithms. Based on the theory of OA and GBD methods, three improved strategies are proposed, and their properties are analyzed, namely multi-generation cuts(MC), hybrid GBD(H-GBD) and partial surrogate cuts(PSC). Five MINLP algorithms are developed through combination of these improved strategies with basic OA and GBD, which are MC-OA, MC-GBD, H-GBD, PSC and MC-PSC. As will be shown, encouraging results are obtained with these methods.

3. Three Improved MINLP Solution Strategies

In order to improve the solution efficiency of OA and GBD methods, three strategies are described in this section, which are multi-generation cuts(MC), hybrid GBD(H-GBD) and partial surrogate cuts(PSC).

3.1. Multi-generation Cuts

When solving MINLP problems with OA or GBD, cuts are accumulated for constructing the master problem in order to predict the lower bounds of MINLP. The OA cuts at the kth iteration are given by the supporting hyperplanes in equations (3.1), which are the same for feasible or infeasible NLP subproblems.

$$\begin{aligned} &\alpha \geq f\left(x^k\right)+\nabla f_x\left(x^k\right)\cdot\left(x-x^k\right)+h^T y \\ &g\left(x^k\right)+\nabla g_x\left(x^k\right)\cdot\left(x-x^k\right)+Hy\leq 0 \end{aligned} \tag{3.1}$$

The GBD cuts at the kth iteration are given by equation (3.2), which involve feasible cuts in KFS, and infeasible cuts in KIS, where KFS is the set of feasible subproblems, and KIS the set of infeasible subproblems.

$$\begin{aligned} \alpha \geq\ & f\left(x^k\right)+h^T\cdot y+\left(\lambda^k\right)^T\cdot\left(g\left(x^k\right)+H\cdot y\right) \\ & +\left(\mu^k\right)^T\cdot\left(A\cdot x^k+E\cdot y-b\right) \qquad k\in KFS \\ \left(\lambda^k\right)^T\cdot & \left(g\left(x^k\right)+H\cdot y\right)+\left(\mu^k\right)^T\cdot\left(A\cdot x^k+E\cdot y-b\right)\leq 0 \quad k\in KIS \end{aligned} \tag{3.2}$$

The idea of MC is that instead of generating one single cut per iteration, that a number of cuts be generated at each iteration to improve the approximation of the MILP. This requires solving multiple NLP subproblems at each iteration. The diagram of multi-generation cuts is shown in Figure 3, and the specific steps in this strategy are as follows:

Step1. While solving the master MILP problem, obtain a set of multiple integer solutions,

including the optimal and suboptimal solutions.

Step2. Fix the integer variables from the S solutions obtained in the MILP master problem in step 1, and solve the S NLP sub-problems.

Step3. Multiple linear supporting cuts are derived from the S solutions in step 2 and added to the MILP master problem.

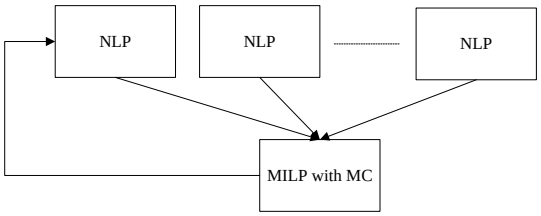

Fig. 3. The diagram of multi-generation cuts

Given S multiple solutions of NLP subproblems in kth iteration, $\left(x^{k,s}, y^{k,s}\right), s=1,2,\ldots,S$, the corresponding OA cuts are as follows.

$$\left.\begin{aligned} &\alpha \geq f\left(x^{k,s}\right)+\nabla f_x\left(x^{k,s}\right)\cdot\left(x-x^{k,s}\right)+h^T y \\ &g\left(x^{k,s}\right)+\nabla g_x\left(x^{k,s}\right)\cdot\left(x-x^{k,s}\right)+Hy\leq 0 \end{aligned}\right\} \quad s\in S \tag{3.3}$$

Similarly, the multiple cuts for GBD are given by equation(3.4).

$$\begin{aligned} \alpha \geq\ & f\left(x^{k,s}\right)+h^T\cdot y+\left(\lambda^{k,s}\right)^T\cdot\left(g\left(x^{k,s}\right)+H\cdot y\right) \\ & +\left(\mu^{k,s}\right)^T\cdot\left(A\cdot x^{k,s}+E\cdot y-b\right) \qquad k\in KFS, s\in S \\ & \left(\lambda^{k,s}\right)^T\cdot\left(g\left(x^{k,s}\right)+H\cdot y\right)+\left(\mu^{k,s}\right)^T\cdot\left(A\cdot x^{k,s}+E\cdot y-b\right)\leq 0 \quad k\in KIS, s\in S \end{aligned} \tag{3.4}$$

To generate multiple solutions for the MILP master problems, a two-phase implementation of branch and cut is used, in which the optimal solution is found in the first phase, and the second phase consists in exploring the branch tree to obtain multiple solutions[33]. By selecting appropriate parameters for the MILP solver, multiple solutions can be obtained according to the required quality or diversity. The criterion for selecting multiple solutions is based on the best

objective function values. Also, note that since the NLP subproblems are independent they can be solved in parallel.

In MC, the upper bound corresponds to the incumbent with the best solution of the NLP subproblems generated in one iteration if it is better than current upper bound. The lower bound corresponds to the solution of the MILP master problem, which increases monotonically.

The MC procedure strengthens the approximation of the master problem, with which improved lower bounds are obtained. Therefore, the total number of iterations of OA or GBD algorithms with MC is reduced, which also means solving fewer MILP problems. The main computational bottleneck for OA or GBD algorithms is often solving the MILP master problem. However, the total CPU computational time of the MC algorithm can be expected to decrease in many problems. The following theorem trivially holds for this strategy.

Theorem 1. The MC strategy does not change the optimal solution of OA or GBD algorithm for convex MINLP problems.

In order to optimize and balance the CPU times for solving the MILP master and the NLP subproblems, we can refine the MC strategy into an adaptive version of MC, where we set a larger number for S at the beginning, and reduce it progressively with the consideration that the master problem will be more time consuming as more cuts are added.

We can simply set S as a decreasing sequence of numbers. For example, we set S as {6,4,2}, and fixed the iterations steps as 10, which means the S is equals to 6 for the first 10 iterations, and changes to 4 from 11^{th} iteration until 20^{th} one, then it changes to 2 until the termination of method.

The other extension is to adjust the size of set S according to a maximum CPU time. In this way, when the CPU time exceeds a maximum value, we reduce the size of S.

3.2. Hybrid GBD

The relative advantages of the two MINLP methods, which are simpler master problem for GBD and tighter lower bound for OA, motivates the use of the two methods in different solution stages. In the improved hybrid strategy, GBD is used to solve the problem in the first stage to derive cuts that provide a rough approximation of the projected feasible region. OA is then used in the second stage with these cuts to complete the convergence of the solution procedure. In this hybrid scheme, the CPU time of the MILP master problem is smaller with GBD, and the larger when one switches to OA.

The following theorem trivially holds for the hybrid strategy.

Theorem 2. The hybrid strategy of OA and GBD does not change the optimal solution of OA or GBD algorithm for convex MINLP problems.

GBD cuts of infeasible NLP subproblems are weak because the discrete variables are only loosely constrained with each infeasibility cut. In contrast for OA, the cuts of infeasible NLPs are tighter than the GBD cuts since they provide an outer approximation in the full space. In order to improve the performance of H-GBD, in the first stage, we add OA cuts when the subproblem is infeasible, but add GBD cuts when the subproblem is feasible.

It is interesting to note that the hybrid scheme where OA is used in the first stage and GBD in the second stage, does not yield good results, because the GBD cuts are weaker, which is not favorable for final convergence.

3.3. Partial Surrogate Cuts

Quesada & Grossmann[27] proposed the concept of partial surrogate cuts, which are special types cuts for MINLP problems. The idea is to partition the continuous variables x into linear w and

nonlinear *v* variables, with which the MINLP becomes,

$$\begin{aligned} \min \quad & z = f(v) + d^T w + h^T y \\ s.t. \quad & g(v) + Dw + Hy \le 0 \\ & A_1 w + A_2 v + Ey \le b \\ & y \in Y, \begin{pmatrix} w \\ v \end{pmatrix} = x \in X \end{aligned} \tag{3.5}$$

where *(w, v)* are linear and nonlinear continuous variables, respectively, according to their occurrence in linear/nonlinear terms. The projection onto nonlinear terms is as follows in PSC,

$$\alpha \ge h^T y + d^T w + f(v^k) + (\lambda^k)^T \left[Hy + Dw + g(v^k) \right] - (\mu^k)^T \mathrm{A}_2(\mathrm{v} - v^k) \tag{3.6}$$

where v^k is the solution of the NLP subproblem. When the NLP subproblem is infeasible, the corresponding partial surrogate cut is,

$$(\lambda^k)^T \cdot \left[Hy + Dw + g(\mathrm{v}^k) \right] - (\mu^k)^T A_2 (v - \mathrm{v}^k) \le 0 \tag{3.7}$$

The MILP master problem of PSC is then as follows[27],

$$\begin{aligned} \min \quad & \alpha \\ s.t. \quad & \alpha \ge h^T y + d^T w + f(v^k) \\ & \quad + (\lambda^k)^T \left[Hy + Dw + g(v^k) \right] - (\mu^k)^T \mathrm{A}_2(\mathrm{v} - v^k) \ \mathrm{k} \in \mathrm{KFS} \\ & (\lambda^k)^T \cdot \left[Hy + Dw + g(\mathrm{v}^k) \right] - (\mu^k)^T \mathrm{A}_2 (v - \mathrm{v}^k) \le 0 \ \mathrm{k} \in \mathrm{KIS} \\ & \mathrm{A}_1 w + \mathrm{A}_2 v + Ey \le b \\ & y \in Y, \begin{pmatrix} w \\ v \end{pmatrix} = x \in X, \alpha \in R^1 \end{aligned} \tag{3.8}$$

Compared with the GBD cuts, the PSC keeps the continuous variables in the Lagrangian cut, and in all the linear constraints. As Quesada & Grossmann pointed out[27], the lower bounds from the master problem OA, PSC and GBD obey the following relationship,

$$LB_{GBD} \le LB_{PSC} \le LB_{OA} \tag{3.9}$$

The PSC method can be regarded as a combination of OA and GBD with dual cuts and linear constraints. Considering that there are often many linear constraints in MINLP models, the PSC

strategy can be expected to have good computational performance solving these types of MINLP problems.

Theorem 3. The decomposition method based on PSC converges to the optimal solution of convex MINLP.

Since the PSC cuts are rigorously derived from a combination of KKT conditions and OA cuts, these cuts do not cut off feasible solutions of convex MINLP models. Therefore, the decomposition method based on PSC converges to the optimal solution of convex MINLP.

We should note that the MC strategy can be applied to PSC. The extension of H-GBD can also be used in PSC. That is when the subproblem is infeasible, generating OA cuts instead of PSC infeasible cuts to accelerate convergence.

4. Improved MINLP algorithms

We describe five improved MINLP algorithms based on the proposed strategies described in Section 3, including MC, H-GBD, and PSC using the basic OA and GBD algorithms. Applying the strategy of MC to OA and GBD, the improved algorithms are defined as MC-OA and MC-GBD. The H-GBD is the third algorithm. The PSC method makes use of partial surrogate cuts in the MILP master problem. MC-PSC method is the combination multi-generation cuts with the decomposition method of PSC. The proposed framework of improved algorithms is illustrated in Figure 4. The termination condition is same for OA or GBD, when the upper or lower bounds lie within a specified tolerance.

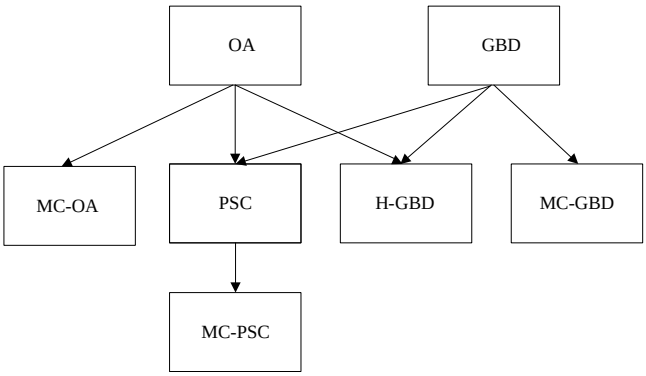

Fig. 4. The Framework of Improved MINLP Algorithms

5. Numerical Experiments

The improved algorithms were implemented on GAMS 24.2[34]. The computer operating system is Windows 7, and quad CPU 2.6 GHz. The solutions of the NLP problems are obtained using the CONOPT3 solver, and the solutions of the MILP problems are obtained using CPLEX12.6. The NLP problems are not solved in parallel. The test instances are 19 benchmark MINLP models from Trespalacios and Grossmann[31] and minlplib[32].

Since integer cuts are used for all methods, the termination criteria for all algorithms are either when a crossover occurs, or the relative gap between incumbent upper and lower bound is within a specified tolerance, which is set 0.001. For GBD, as well as for MC-GBD, PSC and MC-PSC we add OA cuts when subproblem is infeasible, and relevant cuts when feasible.

5.1. MC-OA

We set |S| as 2, 3 and 4 in MC. OA means the special case of MC-OA with |S| equal to 1. MC-OAN means OA with |S| equals to N. For example, MC-OA2 means the number of multi-generation solution is 2. The percentage of problem solved vs. CPU time by OA and MC-OA is shown in Figure 5, based on 19 test instances. As can be seen, the MC-OA methods perform much better than OA with increasing solution time. For the three MC-OA methods,

MC-OA3 performs better than MC-OA2 and MC-OA4, and MC-OA4 performs a little better than MC-OA2 in general.

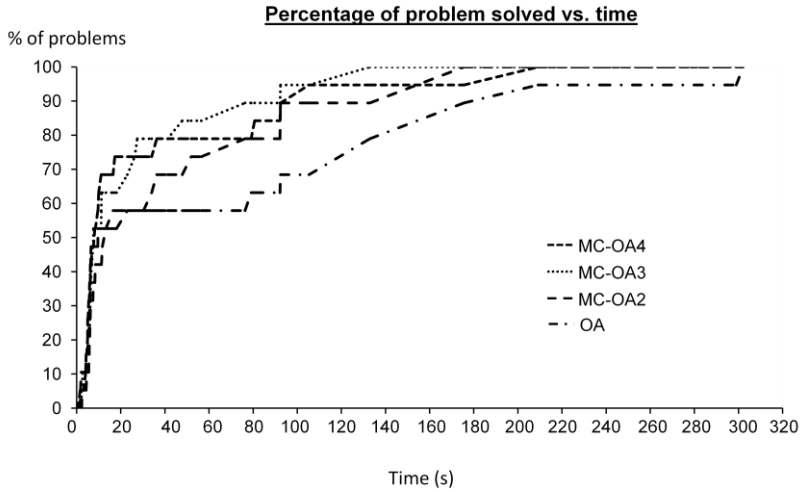

Fig. 5. Comparing CPU performance of MC-OA with basic OA

The number of iterations for 12 test instances is summarized in Table 1, a subset of the 19 test instances. Cases1-4 represents csched1a, 1, 2a, 2, and Cases5-12 are CLAY42-55.

Table 1. Computational Results for MC-OA: Statistics number of Iterations

Method	Case1	Case2	Case3	Case4	Case5	Case6	Case7	Case8	Case9	Case10	Case11	Case12
DICOPT	4	10	88	215	88	593	362	347	40	645	66	516
OA	4	7	111	92	9	10	20	18	11	10	18	26
MC-OA2	4	6	54	133	6	8	11	13	6	7	10	16
MC-OA3	3	7	22	85	4	7	8	12	7	6	7	15
MC-OA4	2	5	25	102	4	6	7	9	6	6	7	13
%Max Reduce	50	28	80	8	56	40	65	50	45	40	61	50
%Min Reduce	0	0	51	-45	33	20	45	28	36	30	44	38

Compared with DICOPT, there are significant reductions in the number of iterations for MC-OA, and even with OA. Compared with OA, the maximum percentage reduction of iterations is 80%,

and the smallest one is 8% in the best case. And in the worst case, the maximum percentage reduction of iterations is 51%, and the smallest one is -45%, which occurs only for one instance. The performance of DICOPT is worse than our implementation of OA for the case when there are many infeasible subproblems, although the linearizations of infeasible NLP subproblem is set in DICOPT. This difference is probably due to the fact that in DICOPT the augmented penalty with slacks is used in the objective function. It is clear that if parallel computation were used for the NLP problems, the performance of MC-OA can be further improved.

5.2. MC-GBD

For MC-GBD, we set the size of multiple solution |S| as 2, 3, 4, 5, 6, as more iterations are required for GBD than for OA. The percentage of problems solved vs. CPU time by GBD and MC-GBD is shown in Figure 6, based on the same 19 test instances. The performance of MC-GBD is better than GBD, but not as much as with MC-OA. That is expected as the number of GBD cuts is less than in the case of OA cuts.

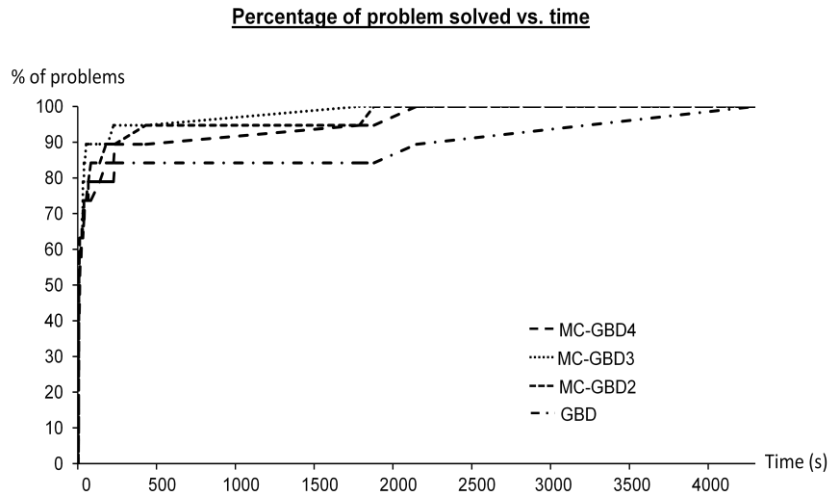

Fig. 6. Comparing CPU performance of MC-GBDs with basic GBD

The iterations for 12 test instances are summarized in Table 2, which are the same MINLP case as with MC-OA. Compared with basic GBD, the maximum percentage reduction of iterations is

91%, and the smallest one is 50% in the best case. In the worst case, the maximum percentage reduction of iterations is 77%, and the smallest one is -65%. For some complex or large-scale MINLP cases, |S| of MC-GBD could be set to larger values than for OA, considering that more iterations are required for GBD than for OA.

Table 2. Computational Results for MC-GBD: Statistics number of Iterations

Method	Case1	Case2	Case3	Case4	Case5	Case6	Case7	Case8	Case9	Case10	Case11	Case12
DICOPT	4	10	88	215	88	593	362	347	40	645	66	516
GBD	11	399	451	1079	12	13	19	20	8	10	18	28
MC-GBD2	8	93	160	1193	7	9	11	17	5	13	12	18
MC-GBD3	5	54	136	1775	6	8	10	13	7	8	9	13
MC-GBD4	2	37	102	364	5	6	9	9	7	10	7	12
MC-GBD5	2	41	182	310	5	6	8	10	6	5	5	11
MC-GBD6	2	38	92	798	5	5	7	7	4	7	5	11
% Max Reduce	82	91	80	64	58	62	63	65	50	50	72	61
% Min Reduce	27	77	60	-65	42	31	42	15	13	-30	33	36

5.3. H-GBD

The H-GBD methods are different when specifying the number of iterations for the GBD stage. In the experiments, we usually set the iterations of the GBD stage to less than 1/3 of the total iterations when solving with OA. For example, for case 3, since the total number of iterations with OA is 111, we set iterations of the GBD stage as 15, 20, 25 and 30 respectively for H-GBD1, 2, 3 and 4.

The comparison results of CPU times and number of iterations are shown in Figure 7 and Table 3.

Although the efficiency of H-GBD is not significant in iterations compared with OA, the solution CPU times of H-GBD are smaller than OA with increased problem size. For the same test instances, the performance of H-GBD is better than GBD, and nearly the same with OA. That means the performance of H-GBD is not worse than one of OA, and even better than the latter in some instances.

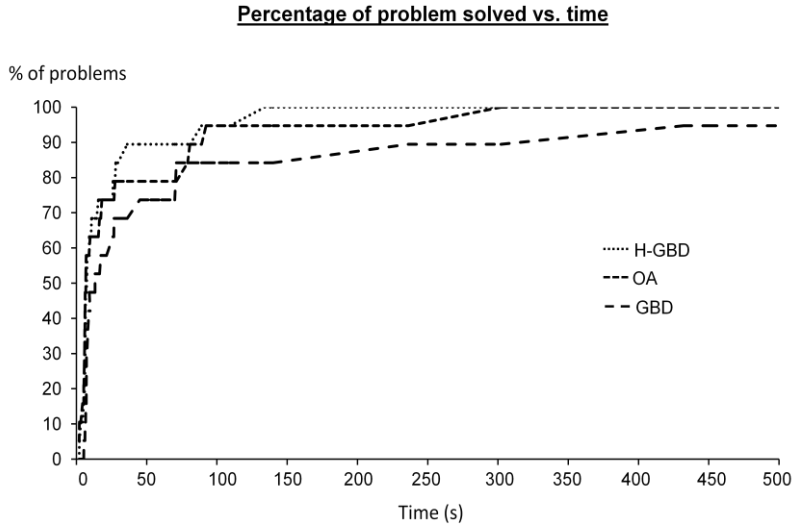

Fig. 7. Comparing CPU performance of H-GBD with OA and GBD

Table 3. Computational Results for H-GBD: Statistics number of Iterations

Method	Case1	Case2	Case3	Case4	Case5	Case6	Case7	Case8	Case9	Case10	Case11	Case12
OA	4	7	111	92	9	10	20	18	11	10	18	26
GBD	11	399	451	1079	12	13	19	20	8	10	18	28
H-GBD1	4	15	51	79	10	12	20	18	11	8	14	25
H-GBD2	5	16	73	101	10	12	20	19	15	11	14	26
H-GBD3	6	17	65	135	11	13	20	21	12	9	11	34
H-GBD4	7	18	48	155	15	13	20	24	17	13	19	33

5.4. PSC and MC-PSC

PSC and MC-PSC2 are used to solve 12 test instances, which is a subset of the larger former test problems. The iterations and CPU times are shown in Figure 8 and Table 4. As can be seen, the performance of PSC is much better than GBD. Compared to OA, PSC generally improves the computational performance. The performance of MC-PSC2 is almost as good as PSC.

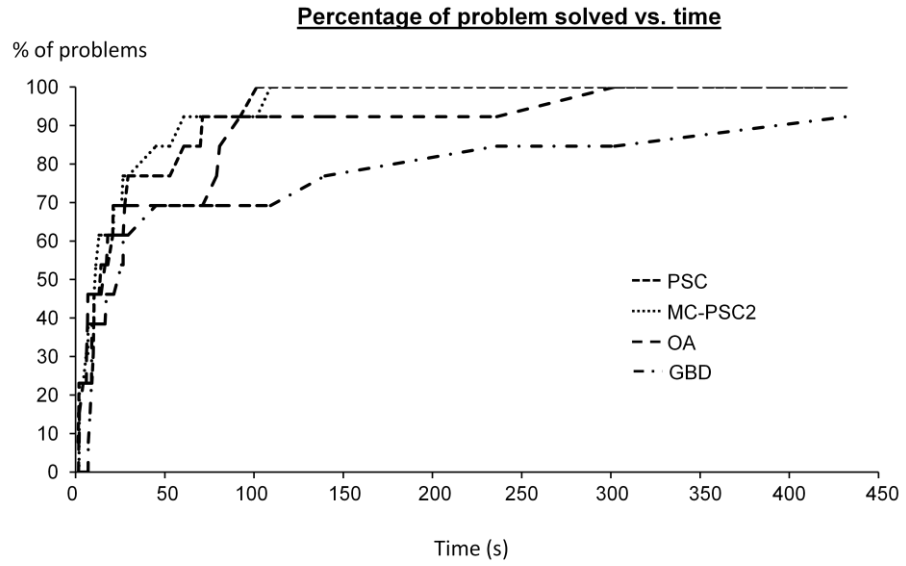

Fig. 8. Comparing CPU performance of PSC / MC-PSC2 with OA / GBD

Table 4. Computational Results for PSC and MC-PSC: Statistics number of Iterations

Method	Case1	Case2	Case3	Case4	Case5	Case6	Case7	Case8	Case9	Case10	Case11	Case12
GBD	11	399	451	1079	12	13	19	20	8	10	18	28
PSC	4	7	88	70	15	13	15	19	16	10	12	27
MC-PSC2	3	6	27	78	14	7	10	18	7	9	8	14
OA	4	7	111	92	9	10	20	18	11	10	18	26

6. Conclusion

In this paper, three MINLP solution strategies have been presented, multi-generation cuts, hybrid with GBD and partial surrogate cuts in order to improve the performance of the MINLP algorithms. MC reduces the total number of iterations, therefore, decreasing the CPU solution times of the master problem. H-GBD is used to control the problem size of master problem. PSC provides a novel approach for decomposition MINLP methods that are characterized by a large number of linear constraints. Five improved MINLP algorithms are described and implemented to increase solution efficiency: MC-OA, MC-GBD, H-GBD, PSC and MC-PSC. Through the testing of certain MINLP benchmark problems, the efficiency of these methods has been illustrated. MC-OA performs best on average among all the methods. H-GBD can be faster than OA on some instances. PSC has shown improved performance compared to OA. MC-GBD and MC-PSC are not as effective as MC-OA.

For general MINLP instances, MC-OA is generally the preferred choice. When the master problem of OA is large, PSC(MC-PSC), H-GBD, MC-GBD are suggested in the given order, since the master problems are smaller than OA. It should be noted that when there are many discrete variables and many infeasible subproblems, OA cuts should be added to the stage for generating partial surrogate or GBD infeasible cuts.

Acknowledgments

The authors acknowledge financial support from State Key Program of National Natural Science Foundation of China (Grant No. 71032004), the Fund for Innovative Research Groups of the

National Natural Science Foundation of China (Grant No. 71321001), and the Fund for the National Natural Science Foundation of China (Grant No. 61374203).

References

1. I. E. Grossmann, J. A. Caballero, H. Yeomans, Mathematical programming approaches for the synthesis of chemical process systems. Korean Journal of Chemical Engineering, 1999, 16, 407-426.
2. Z. Novak, Z. Kravanja, I. E. Grossmann, Simultaneous synthesis of distillation sequences in overall process schemes using an improved MINLP approach. Computers and Chemical Engineering, 1996, 20, 1425 – 1440.
3. K. C. Furman, Nikolaos V. Sahinidis, A critical review and annotated bibliography for heat exchanger network synthesis in the 20th Century. Industrial and Engineering Chemistry Research, 2002, 41, 2335-2370.
4. F. Trespalacios, I. E. Grossmann. Review of Mixed-Integer Nonlinear and Generalized Disjunctive Programming MethodsProgramming Models, Chemie Ingenieur Technik, 2014, 86, 7, 1 – 23.
5. C. A. Méndez, J. Cerdá, I. E. Grossmann, I. Harjunkoski, M. Fahl. State-of-the-art review of optimization methods for short-term scheduling of batch processes. Computers and Chemical Engineering, 2006, 30, 913-946.
6. X. Zhang, R. W. H. Sargent. The Optimal Operation of Mixed Production Facilities-A General Formulation and Some Solution Approaches for the Solution, Computers and Chemical Engineering, 1996, 20, 897-904.
7. L. Mockus, G. V. Reklaitis, Continuous Time Representation Approach to Batch and Continuous Process Scheduling 1. MINLP Formulation. Industrial and Engineering Chemistry Research, 1999, 38, 197-203.
8. N. V. Sahinidis, I. E. Grossmann, MINLP Model for Cyclic Multiproduct scheduling on Continuous Parallel Lines. Computers and Chemical Engineering, 1991, 15, 2, 85-103.

9. J. Pinto, I. E. Grossmann. Optimal Cyclic Scheduling of Multistage Continuous Multiproduct Plants. Computers and Chemical Engineering, 1994, 18, 797-816.

10. V. Jain, I. E. Grossmann. Cyclic Scheduling of continuous parallel-process units with decaying performance. AIChE Journal, 1998, 44,7, 1623-1636.

11. A. Alle, L. G. Papageorgiou, J. M. Pinto. A mathematical programming approach for cyclic production and cleaning scheduling of multistage continuous plants. Computers and Chemical Engineering. 2004, 28, 3-15.

12. C. A. Meyer, C. A. Floudas. Global Optimization of a Combinatorially Complex Generalized Pooling Problem. AIChE Journal, 2006, 52, 3, 1027-1037.

13. R. Karuppiah, K. C. Furman, I. E. Grossmann. Global optimization for scheduling refinery crude oil operations, Computers and Chemical Engineering, 2008, 32, 2745-2766.

14. V. D. Kosmidis, J. D. Perkins, E. N. Pistikopoulo. A mixed integer optimization formulation for the well scheduling problem on petroleum fields. Computers and Chemical Engineering, 2005, 29, 1523-1541.

15. I. E. Grossmann. Enterprise-wide optimization: A new frontier in process systems engineering. AIChE Journal, 2005,51, 1846–1857.

16. I. E. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. Optimization and Engineering, 2002, 3, 227-252.

17. P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan, Mixed-Integer Nonlinear Optimization, Acta Numerica, 2013, 22:1-131.

18. P. Bonami, M. Kilinç, J. Linderoth, Algorithms and software for convex mixed integer nonlinear programs. Mixed Integer Nonlinear Programming, Springer, New York, 2012, 1-39.

19. M. A. Duran, I. E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Mathematical Programming, 1986, 36, 307-339.

20. G. R. Kocis, I. E. Grossmann, Computational Experience with DICOPT solving MINLP Problems in Process Systems Engineering, Computers and Chemical Engineering, 1989, 13, 307-315.

21. J. Viswanathan, I. E. Grossmann, A combined Penalty Function and Outer Approximation Method for MINLP Optimization, Computers and Chemical Engineering, 1990, 14. 769-782.

22. R. Fletcher, S. Leyffer, Solving mixed integer nonlinear programs by outer approximation. Mathematical Programming, 1994, 66, 327-349.

23. C. A. Floudas. Nonlinear Mixed-Integer Optimization: Fundamentals and Applications, Oxford Univeristy Press, The United States, 1995.

24. A. M. Geoffrion. Generalized benders decomposition. Journal of Optimization Theory and Applications, 1972, 10, 4, 237-260.

25. O. K. Gupta, A. Ravindran, Branch and bound experiments in convex nonlinear integer programming, Management Science, 1985, 31, 1533-1546.

26. S. Leyffer, Integrating SQP and branch-and-bound for mixed integer nonlinear programming, Computers Optimization Application, 2001, 18, 295-309.

27. I. Quesada, I. E. Grossmann, An LP/NLP based branched and bound algorithm for convex MINLP optimization problems, Computers and Chemical Engineering, 1992, 16, 937-947.

28. R. A. Stubbs, S. Mehrotra, A branch-and-cut method for 0-1 mixed convex programming. Mathematic Programming, Ser. A, 1999, 86, 515-532.

29. T. Westerlund, H. Skrifvars, I. Harjunkoski, R. Porn. An extended cutting plane method for a class of non-convex MINLP problems, Computers chem. Engng, 1998, 22, 3, 357-365.

30. P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuejols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya and A. Wachter. An algorithmic framework for convex mixed integer nonlinear programs. Discrete Optimization, 2008, 5, 186-204.

31. F. Trespalacios, I. E. Grossmann, Algorithmic approach for improved mixed-integer reformulations of convex Generalized Disjunctive Programs, to appear in INFORMS Journal of Computing. 2014.

32. http://www.minlplib.com

33. E. Danna, M. Fenelon, R. Wunderling, Z. H. Gu. Generating multiple solutions for mixed integer programming models. Proceedings of IPCO 2007, Lecture Notes on Computer Science, 4513, 280-294.

34. A. Brooke, D. Kendrick, A. Meeraus, R. Raman. GAMS – A Users Guide, The Scientific Press, Marrickville, NSW 1998.