Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2010, Article ID 617398, 18 pages
doi:10.1155/2010/617398

Research Article

Reverse Bridge Theorem under Constraint Partition

Minghao Yin,[1,2] Tingting Zou,[1,2] and Wenxiang Gu[1,2]

[1] *College of Computer, Northeast Normal University, Changchun 130117, China*
[2] *Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China*

Correspondence should be addressed to Tingting Zou, zoutt354@nenu.edu.cn

Received 6 October 2009; Revised 7 January 2010; Accepted 7 May 2010

Academic Editor: Joaquim J. Júdice

Reverse bridge theorem (RBTH) has been proved to be both a necessary and sufficient condition for solving Nonlinear programming problems. In this paper, we first propose three algorithms for finding constraint minimum points of continuous, discrete, and mixed-integer nonlinear programming problems based on the reverse bridge theorem. Moreover, we prove that RBTH under constraint partition is also a necessary and sufficient condition for solving nonlinear programming problems. This property can help us to develop an algorithm using RBTH under constraints. Specifically, the algorithm first partitions mixed-integer nonlinear programming problems (MINLPs) by their constraints into some subproblems in similar forms, then solves each subproblem by using RBTH directly, and finally resolves those unsatisfied global constraints by choosing appropriate penalties. Finally, we prove the soundness and completeness of our algorithm. Experimental results also show that our algorithm is effective and sound.

1. Introduction

Nonlinear programming problems (NLPs) play an important role in both manufacturing systems and industrial processes and have been widely used in the fields of operations research, planning and scheduling, optimal control, engineering designs, and production management [1–7]. Due to its significance in both academic and engineering applications, different kinds of approaches have been proposed to solve NLPs and obtained some achievements. In [8], we propose a general approach, called reverse bridge theorem (RBTH), which can significantly reduce the complexity in solving NLPs. We also prove in [8] that RBTH is a necessary and sufficient condition for solving NLPs. Compared to other methods such as extended saddle-point condition (ESPC) [9, 10], RBTH has two obvious advantages. Firstly, the core inequality of RBTH is formed by only one subinequality and one subequality; thus RBTH is easier to handle. Secondly, RBTH does not need extra conditions for solving

NLPs and therefore can be used more widely. However, in [8] we do not provide any concrete algorithm to solve NLPs using RBTH. Consequently, in this paper, we first present three algorithms for solving discrete nonlinear programming problems (DNLPs), continuous nonlinear programming problems (CNLPs), and mixed-integer nonlinear programming problems (MINLPs), respectively. After that, we prove the soundness and completeness of these algorithms.

On the other hand, constraint partition has been proved to be an attractive approach for solving large-scale problems in NLPs recently [11–18]. Based on the regular constraint structure of a problem instance, we can cluster its constraints into multiple loosely coupled partitions. Accordingly, the original problem can be partitioned by its constraints into several subproblems, each of which is a relaxation of the problem and can be solved in exponentially less time than the original problem. In Section 4 of this paper, we first prove that RBTH under constraint partition is a necessary and sufficient condition for solving nonlinear programming problems. Then we further prove that this necessary and sufficient condition can be rewritten into $N + 2$ necessary conditions. This property can help us to develop a novel algorithm using RBTH under constraint partition. Specifically speaking, the algorithm firstly partitions a nonlinear programming problem into several relaxed subproblems, each of which is then solved by using RBTH. After that the algorithm resolves the unsatisfied global constraints by choosing appropriate penalties. Finally, we prove that this algorithm is sound and complete.

The paper is organized as follows. After this introduction, we recall some basic notions and related work in Section 2. Then in Section 3, we introduce how to solve nonlinear programming problems using RBTH. In Section 4, we show how to solve nonlinear programming problems using RBTH under constraint partition. In Section 5, simulations and comparisons based on some benchmarks are carried out, which show that our algorithm is both effective and efficient. In the last section we conclude this paper.

2. Basic Concepts and Related Work

In this section, firstly we recall some basic concepts that will be used in this paper. For details, we refer to [1–7].

2.1. Basic Concepts

Generally speaking, nonlinear programming problems can be divided into three categories, that is, continuous nonlinear programming problems, discrete nonlinear programming problems, and mixed-integer nonlinear programming problems. The general forms of these nonlinear programming problems are as follows:

Definition 2.1 (Continuous Nonlinear Programming). A continuous nonlinear programming problem (CNLP) is defined as

$$\begin{aligned}(P_c):\quad & \min_x f(x) \quad \text{where } x = (x_1, x_2, \ldots, x_v)^T \in \Re^v, \\ & \text{subject to} \quad h(x) = (h_1(x), h_2(x), \ldots, h_m(x))^T = 0, \\ & \qquad\qquad g(x) = \left(g_1(x), g_2(x), \ldots, g_r(x)\right)^T \leq 0,\end{aligned} \tag{2.1}$$

where $x \in \Re^v$ are continuous variables. The function f is assumed to be continuous and differentiable, and the constraint functions g and h can be discontinuous, nondifferentiable, and not in closed form.

Definition 2.2 (Discrete Nonlinear Programming). A discrete nonlinear programming problem (DNLP) is defined as

$$\begin{aligned}(P_d): \quad & \min_y f(y) \quad \text{where } y = (y_1, y_2, \ldots, y_w)^T \in \Re^w, \\ & \text{subject to} \quad h(y) = (h_1(y), h_2(y), \ldots, h_m(y))^T = 0, \\ & \qquad\qquad\quad g(y) = (g_1(y), g_2(y), \ldots, g_r(y))^T \leq 0,\end{aligned} \tag{2.2}$$

where $y \in \Re^w$ are discrete variables. The functions f, g, and h may be assumed discontinuous, nondifferentiable, or not even given in closed form.

Definition 2.3 (Mixed-Integer Nonlinear Programming). A mixed-integer nonlinear programming problem (MINLP) is defined as

$$\begin{aligned}(P_m): \quad & \min_{x,y} f(x,y) \\ & \text{subject to } h(x,y) = (h_1(x,y), h_2(x,y), \ldots, h_m(x,y))^T = 0, \\ & \qquad\qquad\quad g(x,y) = (g_1(x,y), g_2(x,y), \ldots, g_r(x,y))^T \leq 0,\end{aligned} \tag{2.3}$$

where $x \in \Re^v$ are continuous variables, and $y \in \Re^w$ are discrete variables. The function f is bounded below and is assumed to be continuous and differentiable with respect to x, and the constraint functions g and h are general functions that can be discontinuous, nondifferentiable, or not even given in closed form.

The aims of solving continuous, discrete, and mixed-integer nonlinear programming problems are, respectively, to find the constrained minima with respect to neighbourhood of a continuous point x, a discrete point y, and a mixed point (x, y).

Definition 2.4 (Constrained Minimum of CNLP). Point x^* is a constrained minimum of P_c (CMc), if x^* is feasible and $f(x^*) \leq f(x)$ for all feasible x in the neighbourhood of x^*, that is, $x \in N_c(x^*) = \{x' : \|x' - x^*\| \leq \xi \text{ and } \xi \to 0\}$.

Definition 2.5 (Constrained Minimum of DNLP). Point y^* is a constrained minimum of P_d (CMd), if y^* is feasible and $f(y^*) \leq f(y)$ for all feasible y in a neighbourhood of y^*.

Definition 2.6 (Constrained Minimum of MINLP). Point (x^*, y^*) is a constrained minimum of P_m (CMm), if (x^*, y^*) is feasible and $f(x^*, y^*) \leq f(x, y)$ for all feasible (x, y) in a neighbourhood of (x^*, y^*).

2.2. Related Work

In the following, we introduce some existing methods for solving nonlinear programming problems.

2.2.1. Necessary Karush-Kuhn-Tucker (KKT) Condition [18, 19]

KKT is mainly designed for solving continuous nonlinear programming problems. The penalty function of P_c is a Lagrangian function with Lagrange-multiplier vectors $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m)^T \in \Re^m$ and $\beta = (\beta_1, \beta_2, \ldots, \beta_r)^T \in \Re^r$, which is defined as

$$L(x, \alpha, \beta) = f(x) + \alpha^T h(x) + \beta^T g(x). \tag{2.4}$$

Theorem 2.7 (see [19]). *If point x^* is a CMc of P_c and a regular point (gradient vectors of equality constraints and active inequality constraints are linearly independent), then there exist unique $\alpha^* \in \Re^m$ and $\beta^* \in \Re^r$ such that*

$$\nabla_x L(x^*, \alpha^*, \beta^*) = 0, \tag{2.5}$$

where $\beta_j = 0$ for all $j \notin A(x^*) = \{i \mid g_i(x^*) = 0\}$ (the set of active constraints), and $\beta_j \geq 0$ otherwise.

The unique x, α, and β that satisfy equation (2.5) can be found by solving equation (2.5) as a system of nonlinear equations. Because KKT is just a necessary condition, there exist some points that are not CMc. Moreover, the approach is limited to solving CNLPs with continuous and differentiable functions.

2.2.2. Sufficient Saddle-Point (SP) Condition [6]

The concept of saddle point has been widely studied during the past decades.

Theorem 2.8 (see [6]). *For continuous and differentiable constraint functions, point x^* is CMc of P_c if there exist unique $\alpha^* \in \Re^m$ and $\beta^* \in \Re^r$ that satisfy the following saddle-point condition at x^*:*

$$L(x^*, \alpha, \beta) \leq L(x^*, \alpha^*, \beta^*) \leq L(x, \alpha^*, \beta^*), \tag{2.6}$$

for all $x \in N_c(x^*)$ and all $\alpha \in \Re^m$ and $\beta \in \Re^r$.

The existing saddle-point condition is only a sufficient but not necessary condition. This means that there exist some α^* and β^* that do not satisfy (2.6) for each CMc x^* of P_c.

2.2.3. The Necessary and Sufficient Reverse Bridge Theorem (RBTH) [8]

In [8], we propose RBTH as a method for finding a constrained minimum.

Definition 2.9 (Penalty Function for CRBTH). The penalty function of P_c with penalty-multiplier $\alpha \in \Re^m$ and $\beta \in \Re^r$ is defined as

$$L_c(x, \alpha, \beta) = f(x) + \alpha^T |h(x)| + \beta^T \max(0, g(x)), \tag{2.7}$$

where $|h(x)| = (|h_1(x)|,|h_2(x)|,\ldots,|h_m(x)|)$ and $\max(0,g(x)) = (\max(0,g_1(x)),\max(0,g_2(x)),\ldots,\max(0,g_r(x)))$.

Theorem 2.10 (see [8]). *Point x^* is CMc of P_c if and only if there exist finite $\alpha^* \geq 0$ and $\beta^* \geq 0$ such that for any $\alpha^{**} > \alpha^*$, $\beta^{**} > \beta^*$ the following condition is satisfied:*

$$L_c(x^*,\alpha,\beta) = L_c(x^*,\alpha^{**},\beta^{**}) \leq L_c(x,\alpha^{**},\beta^{**}) \tag{2.8}$$

for each x in the neighbourhood of x^* and all $\alpha \in \Re^m$ and $\beta \in \Re^r$.

Definition 2.11 (Penalty Function for DRBTH). The penalty function of P_d with penalty-multiplier $\alpha \in \Re^m$ and $\beta \in \Re^r$ is defined as

$$L_d(x,\alpha,\beta) = f(x) + \alpha^T|h(x)| + \beta^T \max(0,g(x)), \tag{2.9}$$

where $|h(x)| = (|h_1(x)|,|h_2(x)|,\ldots,|h_m(x)|)$ and $\max(0,g(x)) = (\max(0,g_1(x)),\max(0,g_2(x)),\ldots,\max(0,g_r(x)))$.

Theorem 2.12 (see [8]). *Point y^* is CMd of P_d if and only if there exist finite $\alpha^* \geq 0$ and $\beta^* \geq 0$ such that for any $\alpha^{**} > \alpha^*$, $\beta^{**} > \beta^*$ the following condition is satisfied:*

$$L_d(y^*,\alpha,\beta) = L_d(y^*,\alpha^{**},\beta^{**}) \leq L_d(y,\alpha^{**},\beta^{**}), \tag{2.10}$$

for each y in the neighbourhood of y^* and all $\alpha \in \Re^m$ and $\beta \in \Re^r$.

Definition 2.13 (Penalty Function for MRBTH). The penalty function of P_m with penalty-multiplier $\alpha \in \Re^m$ and $\beta \in \Re^r$ is defined as

$$L_m(x,y,\alpha,\beta) = f(x,y) + \alpha^T|h(x,y)| + \beta^T \max(0,g(x,y)), \tag{2.11}$$

where $|h(x,y)| = (|h_1(x,y)|,|h_2(x,y)|,\ldots,|h_m(x,y)|)$ and $\max(0,g(x,y)) = (\max(0,g_1(x,y)),\max(0,g_2(x,y)),\ldots,\max(0,g_r(x,y)))$.

Definition 2.14 (Mixed-Neighbourhood). A mixed-neighbourhood $N(x,y)$ in mixed space $\Re^v \times \Re^w$ is defined as

$$N(x,y) = \{(x',y') \mid \|x'-x\| \leq \xi, \|y'-y\| \leq \xi, \xi \longrightarrow 0,\}. \tag{2.12}$$

Theorem 2.15 (see [8]). *Point (x^*,y^*) is CMm of P_m if and only if there exist finite $\alpha^* \geq 0$ and $\beta^* \geq 0$ such that for any $\alpha^{**} > \alpha^*$, $\beta^{**} > \beta^*$ the following condition is satisfied:*

$$L_m(x^*,y^*,\alpha,\beta) = L_m(x^*,y^*,\alpha^{**},\beta^{**}) \leq L_m(x,y,\alpha^{**},\beta^{**}), \tag{2.13}$$

for all (x,y) in the neighbourhood of (x^*,y^*) and all $\alpha \in \Re^m$ and $\beta \in \Re^r$.

Procedure RBTH_CNLP $(P_c, x, \overline{\alpha}, \overline{\beta})$
$\alpha \to 0, \beta \to 0$
repeat
increase α_i by δ if $(h_i(x) \neq 0$ and $\alpha_i < \overline{\alpha}_i)$ for $i = 1, \ldots, m$;
increase β_j by δ if $(g_j(x) \not\leq 0$ and $\beta_j < \overline{\beta}_j)$ for $j = 1, \ldots, r$;
repeat
perform descent of $L_c(x, \alpha, \beta)$ with respect to x;
until a local minimum of $L_c(x, \alpha, \beta)$ is found;
until a CMc of P_c is found or $(\alpha_i > \overline{\alpha}_i$ for all $h_i(x) \neq 0$ and $\beta_j > \overline{\beta}_j$ for all $g_j(x) \not\leq 0)$.
return CMc if found;
end_procedure

Algorithm 1: The solving procedure for finding CMc of P_c.

3. Solving NLPs Using Reverse Bridge Theorem

We have proved that RBTH is a necessary and sufficient condition for constrained local optima under a range of penalties in [8]. However, in [8] we do not provide any concrete algorithm to solve NLPs using RBTH. Consequently, in this paper, we first present three algorithms for solving CNLPs, DNLPs, and MINLPs, respectively.

We first present an algorithm, called RBTH_CNLP, to find the constrained minimum of CNLPs, as is shown in Algorithm 1. According to Theorem 2.10, if x^* is a local minimum of a CNLP with respect to x, x^* must be a constrained minimum of the CNLP as well. Therefore, given an arbitrary CNLP P_c, to find its constrained minimum, we only need to find its local minimum. According to formula (2.8), let L_c be the penalty function of P_c; a reverse bridge point is the local minimum of L_c. Consequently, in order to find the constraint minimum of a CNLP, we only need to find its reverse bridge point. In this way, we design our algorithm RBTH_CNLP. The key idea of the algorithm is to increase α^{**} and β^{**} gradually and to minimize $L_c(x, \alpha^{**}, \beta^{**})$ simultaneously until $\alpha^{**} > \alpha^*$, $\beta^{**} > \beta^*$.

As we can see in Algorithm 1, we first initialize the values of α^{**} and β^{**}. Then the algorithm is executed to find a local minimum x^* of $L_c(x, \alpha^{**}, \beta^{**})$ according to formula (2.8). If point x^* is not a feasible solution of P_c, then the algorithm increases the penalties corresponding to the violated constraints. The process is repeated until we find that a CMc or α^{**} (resp., β^{**}) is larger than its maximum bound $\overline{\alpha}$ (resp., $\overline{\beta}$).

Algorithm 2 shows the algorithm RBTH_DNLP to solve discrete nonlinear programming problems. The idea of this algorithm is similar to algorithm RBTH_CNLP. Algorithm RBTH_DNLP first initializes the values of α^{**} and β^{**}, then gradually increases α^{**} and β^{**}, and minimizes $L_d(y, \alpha^{**}, \beta^{**})$ until $\alpha^{**} > \alpha^*$, $\beta^{**} > \beta^*$.

In order to solve MINLPs using RBTH, we need to define two neighbourhoods, that is, discrete neighbourhood and mixed neighbourhood.

Definition 3.1 (Discrete Neighbourhood). A discrete neighbourhood $N_d(y)$ of y in discrete space $\Re^w$ is a finite set of points $\{y' \in \Re^w\}$ in such a way that y' is reachable from y in one step, that $y' \in N_d(y) \Leftrightarrow y \in N_d(y')$, and that it is possible to reach every y'' from any y in one or more steps through neighbouring points.

Procedure RBTH_DNLP $(P_d, y, \overline{\alpha}, \overline{\beta})$
$\alpha \to 0, \beta \to 0$;
repeat
increase α_i by δ if $(h_i(y) \neq 0$ and $\alpha_i < \overline{\alpha}_i)$ for $i = 1, \ldots, m$;
increase β_j by δ if $(g_j(y) \not\leq 0$ and $\beta_j < \overline{\beta}_j)$ for $j = 1, \ldots, r$;
repeat
perform descent of $L_d(y, \alpha, \beta)$ with respect to y;
until a local minimum of $L_d(y, \alpha, \beta)$ is found;
until a CMd of P_d is found or ($\alpha_i > \overline{\alpha}_i$ for all $h_i(y) \neq 0$ and $\beta_j > \overline{\beta}_j$ for all $g_j(y) \not\leq 0$).
return CMd if found;
end_procedure

Algorithm 2: The solving procedure for finding CMd of P_d.

Definition 3.2 (Mixed Neighbourhood). A mixed neighbourhood $N_m(x, y)$ in mixed space $\Re^v \times \Re^w$ is defined as

$$N_m(x, y) = N_c(x)|_y + N_d(y)|_x = \{(x', y) \mid x' \in N_c(x)\} \bigcup \{(x, y') \mid y' \in N_d(y)\}, \tag{3.1}$$

where $N_c(x)$ is continuous neighbourhood of variable x, and $N_d(y)$ is discrete neighbourhood of variable y.

Corollary 3.3. *According to Definition 3.2 and Theorem 2.15, the RBTH in formula* (2.13) *can be rewritten into two following necessary conditions that, collectively, are sufficient:*

$$\begin{aligned} L_m(x^*, y^*, \alpha, \beta) = L_m(x^*, y^*, \alpha^{**}, \beta^{**}) \leq L_m(x^*, y, \alpha^{**}, \beta^{**}), \quad y \in N_d(y^*), \\ L_m(x^*, y^*, \alpha^{**}, \beta^{**}) \leq L_m(x, y^*, \alpha^{**}, \beta^{**}), \quad x \in N_c(x^*). \end{aligned} \tag{3.2}$$

Based on the corollary, we present an algorithm RBTH_MINLP, as is seen in Algorithm 3, to find the constrained minimum of MINLPs. According to Theorem 2.15 as we mentioned in Section 2, if the point (x^*, y^*) is a local minimum of an MINLP with respect to (x, y), this point must also be a constrained minimum of the MINLP. This means that given an arbitrary MINLP P_m, to find its constrained minimum, we only need to find its local minimum. On the other hand, according to formula (2.13), let L_m be the penalty function of P_m; a reverse bridge point is the local minimum of L_m. In this sense, in order to find a MINLP's constrained minimum, we only need to find the reverse bridge point.

In the procedure of RBTH_MINLP, we first initialize the values of α^{**} and β^{**}. In the first inner loop, we focus on finding the local minimum x^* of $L_m(x, y, \alpha, \beta)$ with respect to the continuous neighbourhoods of x; in the second inner loop, we are devoted to looking for the local minimum y^* by $L_m(x, y, \alpha, \beta)$ with respect to the discrete neighbourhoods of y. If the local minimum point (x^*, y^*) violates global constraints of P_m, we increase the penalties of violated constraints in the outer loop. The process is repeated until we find that a CMm of P_m or α^{**} (resp., β^{**}) is larger than $\overline{\alpha}$ (resp., $\overline{\beta}$).

Procedure RBTH_MINLP $(P_m, x, y, \overline{\alpha}, \overline{\beta})$
$\alpha \to 0, \beta \to 0;$
repeat
increase α_i by δ if $(h_i(x) \neq 0$ and $\alpha_i < \overline{\alpha}_i)$ for $i = 1, \ldots, m;$
increase β_j by δ if $(g_j(x) \not\leq 0$ and $\beta_j < \overline{\beta}_j)$ for $j = 1, \ldots, r;$
repeat
perform descent of $L_m(x, y, \alpha, \beta)$ with respect to x for given $y;$
until a local minimum of $L_m(x, y, \alpha, \beta)$ with respect to x is found;
repeat
perform descent of $L_m(x, y, \alpha, \beta)$ with respect to y for given $x;$
until a local minimum of $L_m(x, y, \alpha, \beta)$ with respect to y is found;
until a CMm of P_m is found or $(\alpha_i > \overline{\alpha}_i$ for all $h_i(x) \neq 0$ and $\beta_j > \overline{\beta}_j$ for all $g_j(x) \not\leq 0)$
return CMm if found;
end_procedure

Algorithm 3: The resolving procedure for finding CMm of P_m.

Obviously, the main idea of all the three algorithms RBTH_CNLP, RBTH_DNLP, and RBTH_MINLP is to find the reverse bridge points for CNLPs, DNLPs, and MINLPs, respectively. According to Theorems 2.10, 2.12 and 2.15 as we mentioned in Section 2, we know that these reverse bridge points are also constraint minima of NLPs. Because CRBTH, DRBTH, and MRBTH are all necessary and sufficient conditions for solving CNLPs, DNLPs, and MINLPs, respectively, as we proved in [8], the following theorem stands.

Theorem 3.4. *Given a CNLP (DNLP, MINLP, resp.), if there exists a solution, the algorithm RBTH_CNLP (RBTH_DNLP, RBTH_MINLP, resp.) can find the solution; if algorithm RBTH_CNLP (RBTH_DNLP, RBTH_MINLP, resp.) finds a solution, then it must be the solution of the CNLP.*

In this section, we have shown that RBTH is an effective method for solving NLPs. However, it is difficult and expensive to solve some large-scale NLPs by RBTH because of their huge search spaces and several constraints in different form. Thus, in the next section, we further propose an approach to solve large-scale NLPs using RBTH under constraint partition.

4. RBTH under Constraint Partition

In this section, we show how to solve a nonlinear programming problem using RBTH under constraint partition. Because CNLPs and DNLPs can be regarded as special cases of MINLPs, in this section we only focus on MINLPs. Constraint partitioning has led to a major breakthrough in solving nonlinear programming problems in operations research and engineering applications [11–18]. In this section, we first prove that RBTH under constraint partition is also a necessary and sufficient condition for solving MINLPs. Then we prove that this necessary and sufficient condition can be rewritten into several necessary conditions by providing a partitioned neighbourhood. After that, we present an algorithm for solving MINLPs and prove that this algorithm is sound and complete.

Table 1: Results of solving selected CNLP benchmarks from the CUTE library. All timing results are in seconds. Here nc and nv represent the number of constraints and the number of variables, respectively. "—" means that no feasible solutions were found in the time limit (600 seconds). "∗" means that solutions were obtained by submitting problems through commercial version of LANCELOT but no CPU times were available. Numbers in bold represent the best solutions among the three methods.

Test Problem			LANCELOT		SNOPT		CPRBTH	
ID	nc	nv	Sol.	Time	Sol.	Time	Sol.	Time
ALJAZZAF	3	1	**75.00**	0.46	**75.00**	0.01	**75.00**	0.05
ALLINITC	4	1	**30.44**	∗	30.49	0.01	30.49	0.10
ALSOTAME	2	1	0.082	0.57	**0.08**	0.01	**0.08**	0.09
AVION2	49	45	—	—	**9.47E7**	0.01	**9.47E7**	0.10
BATCH	46	73	—	—	**2.59E5**	0.01	**2.59E5**	0.11
BT11	5	3	0.825	0.62	**0.82**	0.01	**0.82**	0.07
BT12	5	3	**6.188**	0.47	6.19	0.01	6.19	0.07
BT6	5	2	**0.277**	0.56	0.28	0.01	0.28	0.07
CB2	3	3	1.952	0.60	**1.95**	0.01	**1.95**	0.07
CRESC4	6	8	—	—	**0.87**	0.01	**0.87**	0.01
CSFI1	5	4	−49.07	0.63	**−49.08**	0.01	**−49.08**	0.09
DIPIGRI	7	4	680.6	0.68	**680.63**	0.01	**680.63**	0.09
DIXCHLNG	10	5	0.0	1.12	**2.47E3**	0.01	**2.47E3**	0.05
DNIEPER	61	24	**1.87E4**	0.83	**1.87E4**	0.01	**1.87E4**	0.13
EXPFITA	5	22	1.13E−3	0.65	**0.00**	0.01	**0.00**	0.10
GAUSSELM	14	11	−2.25	0.55	**0.00**	104.90	**0.00**	0.12
HIMMELBI	100	12	−1735.6	1.23	**−1755.00**	0.01	**−1755.00**	0.13
HIMMELP2	2	1	−62.05	0.63	**−62.05**	0.01	**−62.05**	0.05
HIMMELP6	2	5	−59.01	0.69	**−59.01**	0.01	**−59.01**	0.05
HONG	4	1	22.57	0.50	**1.35**	0.01	**1.35**	0.05
HUBFIT	2	1	0.0169	0.46	**0.02**	0.01	**0.02**	0.04
LOADBAL	31	31	0.453	0.69	**0.45**	0.01	**0.45**	0.12
MADSEN	3	6	**0.616**	0.55	0.62	0.01	0.62	0.03
MARATOS	2	1	**−1.00**	0.40	**−1.00**	0.01	**−1.00**	0.02
MATRIX2	6	2	**0.00**	0.52	**0.00**	0.01	**0.00**	0.01
MISTAKE	9	13	**−1.00**	0.58	**−1.00**	0.01	**−1.00**	0.11
MWRIGHT	5	3	**24.97**	0.56	24.98	0.01	24.98	0.01
ODFITS	10	6	−2380	0.50	**−2380.03**	0.01	**−2380.03**	0.03
OPTCNTRL	32	20	**550.00**	0.51	**550.00**	0.01	**550.00**	0.03
OPTPRLOC	30	30	**−16.42**	4.02	**−16.42**	0.01	**−16.42**	0.11
OPTHREGB	27	6	**0.0**	0.76	**0.00**	0.01	**0.00**	0.06
PENTAGON	6	15	1.509E−4	0.56	**0.00**	0.01	0.00	0.09

4.1. RBTH for Partitioned Subproblems

Given an arbitrary mixed-integer nonlinear programming problem P_t, we can partition its constraints into $N + 1$ stages. Each stage t $(t = 0, 1, \ldots, N)$ includes u_t local variables, m_t local equality constraints, and r_t local inequality constraints. Here local constraints restrict the variables of each stage, and global constraints restrict all the variables of problems. By applying this partition, the variable vector $z \in Z$ of the problem P_t can be decomposed into $N + 1$ subvectors $z(0), z(1), \ldots, z(N)$, where $z(t) = (z_1(t), \ldots, z_{u_t}(t))^T$ is a vector of dynamic

state variables in mixed space and stage t. In this sense, the MINLP formulation P_t is as follows:

$$\begin{aligned}(P_t): \quad & \min_z J(z) \\ & \text{subject to} \quad h^{(t)}(z(t)) = 0, \quad g^{(t)}(z(t)) \le 0, \\ & \qquad\qquad H(z) = 0, \quad G(z) \le 0,\end{aligned} \tag{4.1}$$

where J is assumed to be continuous and differentiable with respect to z, $h^{(t)} = (h_1^{(t)}, \ldots, h_{m_t}^{(t)})^T$ and $g^{(t)} = (g_1^{(t)}, \ldots, g_{r_t}^{(t)})^T$ are vectors of local-constraint functions that involve $z(t)$ and time in stage t, and $H = (H_1, \ldots, H_p)^T$ and $G = (G_1, \ldots, G_q)^T$ are vectors of global-constraint functions that involve state variables and time in two or more stages.

A solution z of P_t can be regarded as the assignments of all the variables in z. The goal of solving P_t is then to find a constraint minimum with respect to all the feasible solutions in its mixed neighbourhood. It is clear that the partition of each stage needs to be further decomposed into discrete and continuous parts; however, we do not consider such situation for the purpose of simplification. In the following, we define the penalty function of P_t and then propose the partitioned necessary and sufficient RBTH condition on CMm of P_t.

Definition 4.1 (Penalty Function). The penalty function for P_t and the corresponding penalty function in stage t are defined as follows:

$$\begin{aligned}L_m(z, \alpha, \beta, \gamma, \eta) &= J(z) + \sum_{t=0}^{N} \left\{ \alpha(t)^T \left| h^{(t)}(z(t)) \right| + \beta(t)^T \max\left(0, g^{(t)}(z(t))\right) \right\} \\ &\quad + \gamma^T |H(z)| + \eta^T \max(0, G(z)), \\ \Gamma_m(z, \alpha(t), \beta(t), \gamma, \eta) &= J(z) + \alpha(t)^T \left| h^{(t)}(z(t)) \right| + \beta(t)^T \max\left(0, g^{(t)}(z(t))\right) \\ &\quad + \gamma^T |H(z)| + \eta^T \max(0, G(z)),\end{aligned} \tag{4.2}$$

where $\alpha(t) = (\alpha_1(t), \ldots, \alpha_{m_t}(t))^T \in \Re^{m_t}$ and $\beta(t) = (\beta_1(t), \ldots, \beta_{r_t}(t))^T \in \Re^{r_t}$ are the penalty vectors for the local constraints in stage t, $t = 0, 1, \ldots, N$; $\gamma = (\gamma_1, \ldots, \gamma_p) \in \Re^p$ and $\eta = (\eta_1, \ldots, \eta_q) \in \Re^q$ are the penalty vectors for the global constraints.

Theorem 4.2. *Solution z^* is a CMm of P_t with respect to its mixed neighbourhood if and only if there exist finite $\alpha^* \ge 0$, $\beta^* \ge 0$, $\gamma^* \ge 0$, and $\eta^* \ge 0$ for any $\alpha^{**} > \alpha^*$, $\beta^{**} > \beta^*$, $\gamma^{**} > \gamma^*$, and $\eta^{**} > \eta^*$ such that the following RBTH condition is satisfied:*

$$L_m(z^*, \alpha, \beta, \gamma, \eta) = L_m(z^*, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}) \le L_m(z, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}), \tag{4.3}$$

for all $\alpha \in \Re^{\sum_{i=0}^{N} m_i}$, $\beta \in \Re^{\sum_{i=0}^{N} r_i}$, $\gamma \in \Re^p$, $\eta \in \Re^q$, and $z \in N_m(z^*)$.

Proof. The proof consists of two parts.

"$\Rightarrow$" parts: given a constraint minimum z^*, we need to prove that there exist finite $\alpha^* \ge 0$, $\beta^* \ge 0$, $\gamma^* \ge 0$, and $\eta^* \ge 0$ to satisfy formula (4.3).

Equality Part:

z^* is a feasible solution; so it satisfies all local constraints and global constraints. Therefore, we obtain

$$\begin{aligned} L_m(z^*, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}) &= J(z^*) + \sum_{t=0}^{N} \left\{ \alpha(t)^{**T} \left| h^{(t)}(z^*(t)) \right| + \beta(t)^{**T} \max\left(0, g^{(t)}(z^*(t))\right) \right\} \\ &\quad + \gamma^{**T} |H(z^*)| + \eta^{**T} \max(0, G(z^*)) = J(z^*), \\ L_m(z^*, \alpha, \beta, \gamma, \eta) &= J(z^*) + \sum_{t=0}^{N} \left\{ \alpha(t)^{T} \left| h^{(t)}(z^*(t)) \right| + \beta(t)^{T} \max\left(0, g^{(t)}(z^*(t))\right) \right\} \\ &\quad + \gamma^{T} |H(z^*)| + \eta^{T} \max(0, G(z^*)) = J(z^*). \end{aligned} \tag{4.4}$$

So, $L_m(z^*, \alpha, \beta, \gamma, \eta) = L_m(z^*, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**})$.

Inequality Part:

It is as follows.

(1) z^* is the unique minimum of $J(\cdot)$. For any z, $|H(z)| \geq 0$, $\max(0, G(z)) \geq 0$, $|h^{(t)}(z(t))| \geq 0$, and $\max(0, g^{(t)}(z(t))) \geq 0$. Therefore the following equation is right regardless of the choice of the penalties:

$$L_m(z, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}) \geq J(z) \geq J(z^*) = L_m(z^*, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}). \tag{4.5}$$

(2) z^* is not the unique minimum of $J(\cdot)$ or is not a minimum of $J(\cdot)$. Assume that there exists some z', which satisfies $J(z') \leq J(z^*)$.

(i) $J(z') = J(z^*)$, $H(z') = 0$, $G(z') \leq 0$, $h^{(t)}(z'(t)) = 0$, and $g^{(t)}(z'(t)) \leq 0$. This means that z' is another CMm of P_t. Therefore, regardless of the choice of penalties, the inequality of formula (4.3) is satisfied.

(ii) $J(z') = J(z^*)$, and z' violates some constraints. We suppose that it violates a global equality constraint function $H_i(\cdot)$ (the case with a global inequality constraint or a local constraint is similar), and thus $\gamma^* \geq 0$ is enough.

(iii) $J(z') \leq J(z^*)$, and z' is a feasible solution. This is impossible because z^* is not CMm of P_t in this situation.

(iv) $J(z') \leq J(z^*)$, and z' violates some constraints. We suppose that it violates a global equality constraint function $H_i(\cdot)$ (the case with a global inequality constraint or a local constraint is similar), so $|H_i(z')| \neq 0$. Therefore, let $\gamma^* = (J(z^*) - J(z'))/|H_i(z')|$. Then we have $L_m(z^*, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}) = J(z^*)$,

$$\begin{aligned} L_m(z', \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}) &= J(z') + \sum_{t=0}^{N} \left\{ \alpha(t)^{**T} \left| h^{(t)}(z'(t)) \right| \right. \\ &\qquad \left. + \beta(t)^{**T} \max\left(0, g^{(t)}(z'(t))\right) \right\} \\ &\quad + \gamma^{**T} \left| H(z') \right| + \eta^{**T} \max(0, G(z')) \\ &\geq J(z') + \gamma_i^{**} \left| H_i(z') \right| \\ &> J(z') + \frac{J(z^*) - J(z')}{|H_i(z')|} \left| H_i(z') \right| = J(z^*). \end{aligned} \tag{4.6}$$

Thus $L_m(z^*, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}) \leq L_m(z, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**})$.

"$\Leftarrow$" parts: assume that formula (4.3) is satisfied; we need to prove that z^* is a CMm of P_t. Because $L_m(z^*, \alpha, \beta, \gamma, \eta) = L_m(z^*, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**})$, we have

$$\begin{aligned} &J(z^*) + \sum_{t=0}^{N} \left\{ \alpha(t)^T \left| h^{(t)}(z^*(t)) \right| + \beta(t)^T \max\left(0, g^{(t)}(z^*(t))\right) \right\} + \gamma^T |H(z^*)| + \eta^T \max(0, G(z^*)) \\ &= J(z^*) + \sum_{t=0}^{N} \left\{ \alpha(t)^{**T} \left| h^{(t)}(z^*(t)) \right| + \beta(t)^{**T} \max\left(0, g^{(t)}(z^*(t))\right) \right\} \\ &\quad + \gamma^{**T} |H(z^*)| + \eta^{**T} \max(0, G(z^*)), \end{aligned} \tag{4.7}$$

for any $\alpha(t)$, $\beta(t)$,γ, η and $t = 0, 1, \ldots, N$. Thus we can obtain $|H(z^*)| = 0$, $\max(0, G(z^*)) = 0$, $|h^{(t)}(z^*(t))| = 0$ and $\max(0, g^{(t)}(z^*(t))) = 0$. Therefore z^* is a feasible solution.

In the following, we prove that z^* is minimum for all feasible solutions. Assume that z' is another feasible solution; we have

$$\begin{aligned} L_m(z^*, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}) &= J(z^*) + \sum_{t=0}^{N} \left\{ \alpha(t)^{**T} \left| h^{(t)}(z^*(t)) \right| + \beta(t)^{**T} \max\left(0, g^{(t)}(z^*(t))\right) \right\} \\ &\quad + \gamma^{**T} |H(z^*)| + \eta^{**T} \max(0, G(z^*)) = J(z^*), \\ L_m(z', \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}) &= J(z') + \sum_{t=0}^{N} \left\{ \alpha(t)^{**T} \left| h^{(t)}(z'(t)) \right| + \beta(t)^{**T} \max\left(0, g^{(t)}(z'(t))\right) \right\} \\ &\quad + \gamma^{**T} \left| H(z') \right| + \eta^{**T} \max(0, G(z')) = J(z'). \end{aligned} \tag{4.8}$$

Because $L_m(z^*, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}) \leq L_m(z, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**})$, we get $J(z^*) \leq J(z')$. Therefore, z^* is a constrained minimum of P_t. □

In order to partition RBTH into several independent necessary conditions efficiently, we define the mixed neighbourhood of solution z^* as follows.

Definition 4.3 (Partitioned Mixed Neighbourhood). The partitioned mixed neighbourhood of $z \in Z$, denoted by $N_m(z)$, is defined as

$$N_m(z) = \bigcup_{i=0}^{N} N_p^{(t)}(z) = \bigcup_{i=0}^{N} \{z' \mid z'(t) \in N_m(z(t)), z'(i \mid i \neq t) = z(i)\}, \tag{4.9}$$

where $N_m(z(t))$ is the mixed-space neighbourhood of variable vector $z(t)$ in stage t.

Based on this definition, we can further partition the condition described in formula (4.3) into multiple conditions.

Theorem 4.4. *Given $N_m(z)$, the RBTH in formula* (4.3) *can be rewritten into $N + 2$ following necessary conditions that, collectively, are sufficient:*

$$\Gamma_m(z^*, \alpha(t), \beta(t), \gamma^{**}, \eta^{**}) = \Gamma_m(z^*, \alpha(t)^{**}, \beta(t)^{**}, \gamma^{**}, \eta^{**}) \leq \Gamma_m(z, \alpha(t)^{**}, \beta(t)^{**}, \gamma^{**}, \eta^{**}) \tag{4.10}$$

$$L_m(z^*, \alpha(t)^{**}, \beta(t)^{**}, \gamma, \eta) = L_m(z^*, \alpha(t)^{**}, \beta(t)^{**}, \gamma^{**}, \eta^{**}) \tag{4.11}$$

for all $z \in N_m^{(t)}(z^*)$, $\alpha(t) \in \Re^{m_t}$, $\beta(t) \in \Re^{r_t}$, $\gamma \in \Re^p$, $\eta \in \Re^q$, and $t = 0, 1, \ldots, N$.

Proof. We prove that formula (4.3) is equivalent to the combined formula (4.10) and formula (4.11).

"⇒" parts: given a z^* satisfying formula (4.3), we need to prove that it also satisfies formula (4.10) and formula (4.11). For all $t = 0, 1, \ldots, N$, any point $z \in N_m^{(t)}(z^*)$ is also a point in $N_m(z^*)$; therefore $J(z^*) \leq J(z)$, $H(z^*) = 0$, $\max(0, G(z^*)) = 0$, $h^{(t)}(z^*(t)) = 0$, and $\max(0, g^{(t)}(z^*(t))) = 0$:

$$\begin{aligned}
\Gamma_m(z, \alpha(t)^{**}, \beta(t)^{**}, \gamma^{**}, \eta^{**}) &= J(z) + \alpha(t)^{**T} \left| h^{(t)}(z(t)) \right| + \beta(t)^{**T} \max\left(0, g^{(t)}(z(t))\right) \\
&\quad + \gamma^{**T} |H(z)| + \eta^{**T} \max(0, G(z)) \\
&\geq J(z), \\
\Gamma_m(z^*, \alpha(t)^{**}, \beta(t)^{**}, \gamma^{**}, \eta^{**}) &= J(z^*) + \alpha(t)^{**T} \left| h^{(t)}(z^*(t)) \right| + \beta(t)^{**T} \max\left(0, g^{(t)}(z^*(t))\right) \\
&\quad + \gamma^{**T} |H(z^*)| + \eta^{**T} \max(0, G(z^*)) \\
&= J(z^*).
\end{aligned} \tag{4.12}$$

Therefore, $\Gamma_m(z^*, \alpha(t), \beta(t), \gamma^{**}, \eta^{**}) \leq \Gamma_m(z, \alpha(t)^{**}, \beta(t)^{**}, \gamma^{**}, \eta^{**})$.

Then we can know that the equality in formula (4.10) and the equality in formula (4.11) hold in case of z^* satisfying all the constraints.

"⇐" parts: we prove this part by contradiction. Assume that z^* satisfies formula (4.10) and formula (4.11) but does not satisfy formula (4.3). The equality in formula (4.3) cannot be violated because the equality in formula (4.10) and the equality in formula (4.11) imply that all local and global constraints are satisfied. Therefore, the inequality in formula (4.3) is

unsatisfied. In this case, there exists some $N_m(z^*)$ and a unique t' where $z \in N_m^{(t)}(z^*)$ such that

$$L_m(z^*, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}) \not\leq L_m(z, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}). \tag{4.13}$$

But

$$\begin{aligned} L_m(z, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}) &= J(z) + \sum_{t=0}^{N} \left\{ \alpha(t)^{**T} \left| h^{(t)}(z(t)) \right| + \beta(t)^{**T} \max\left(0, g^{(t)}(z(t))\right) \right\} \\ &\quad + \gamma^{**T} |H(z)| + \eta^{**T} \max(0, G(z)), \\ L_m(z^*, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}) &= J(z^*) + \sum_{t=0}^{N} \left\{ \alpha(t)^{**T} \left| h^{(t)}(z^*(t)) \right| + \beta(t)^{**T} \max\left(0, g^{(t)}\ (z^*(t))\right) \right\} \\ &\quad + \gamma^{**T} |H(z^*)| + \eta^{**T} \max(0, G(z^*)) \\ &= J(z^*). \end{aligned} \tag{4.14}$$

Therefore

$$\begin{aligned} J(z^*) \not\leq J(z) &+ \alpha(t')^{**T} \left| h^{(t)}(z(t')) \right| + \beta(t')^{**T} \max\left(0, g^{(t)}\ (z(t'))\right) \\ &+ \gamma^{**T} |H(z)| + \eta^{**T} \max(0, G(z)). \end{aligned} \tag{4.15}$$

This implies that

$$\Gamma_m(z^*, \alpha(t')^{**}, \beta(t')^{**}, \gamma^{**}, \eta^{**}) \not\leq \Gamma_m(z, \alpha(t')^{**}, \beta(t')^{**}, \gamma^{**}, \eta^{**}) \tag{4.16}$$

holds for $t = t'$, which contradicts our assumption.

Therefore, any z^* that satisfies formula (4.10) and formula (4.11) must also satisfy (4.3).

Theorem 4.4 proves that the original RBTH mentioned in Theorem 2.15 can be partitioned into $N+1$ necessary conditions as formula (4.10) and a global necessary condition as formula (4.11). Consequently, local reverse bridge points, which satisfy formula (4.10) in stage t, are local minimum of the original RBTH in stage t. The above reverse bridge points are essentially the solutions of solving the following MINLP $P_t^{(t)}$, in which we add the global constraints to the original objective function $J(z)$:

$$\begin{aligned} \left(P_t^{(t)}\right): \quad & \min_{z(t)} J^{(t)}(z) = J(z) + \gamma^T |H(z)| + \eta^T \max(0, G(z)) \\ & \text{subject to} \quad h^{(t)}(z(t)) = 0, \quad g^{(t)}(z(t)) \leq 0. \end{aligned} \tag{4.17}$$

□

Procedure RBTH_partition_resolve_mixed$(P_t, z, \overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\eta})$
  $\gamma \to 0$, $\eta \to 0$;
  repeat
    increase γ_i by δ if $(H_i(z) \neq 0$ and $\gamma_i < \overline{\gamma}_i)$ for $i = 1, \ldots, p$;
    increase η_j by δ if $(G_j(z) \not\leq 0$ and $\eta_j < \overline{\eta}_j)$ for $j = 1, \ldots, q$;
    for $t = 0$ **to** N
      call RBTH_MINLP $(P_t^{(t)}, z, \overline{\alpha}, \overline{\beta})$ to solve $P_t^{(t)}$.
    end for;
  until a CMm of P_t is found or $(\gamma_i > \overline{\gamma}_i$ for all $H_i(z) \neq 0$ and $\eta_j > \overline{\eta}_j$ for all $G_j(z) \not\leq 0)$.
  return CMm of P_t if found;
end_procedure

Algorithm 4: The partitioning and resolving procedure for finding CMm of P_t.

In brief, solving the original problem can be reduced to solve multiple MINLPs defined by $P_t^{(t)}$ in formula (4.17) and to increase the penalties of violated global constraints defined in formula (4.11). Therefore, by partitioning the original problem and solving each subproblem, Theorem 4.4 leads to a significant reduction on computational complexity.

4.2. The Partitioning and Resolving Procedure

Algorithm 4 shows the partitioning and resolving procedure for finding the reverse bridge points satisfying the conditions in Theorem 4.4. As is shown in the algorithm, γ^* and η^* are initialized firstly; the inner loop is then carried out to find a reverse bridge point of MINLP $P_t^{(t)}$ in each stage t, which can be implemented by the procedure RBTH_MINLP, as we introduced in Section 3. After all the subproblems are solved, the penalties corresponding to the violated global constraints are increased in the outer loop. The process is repeated until a CMm of P_t is found or γ^{**} and η^{**} are larger than their maximum bounds $\overline{\gamma}$ and $\overline{\eta}$.

According to Theorems 4.2 and 4.4, we know that solving the original problem can be reduced to solve multiple MINLPs. For every MINLP in stage t of original problem, algorithm RBTH_partition_resolve_mixed calls RBTH_MINLP to solve it. Moreover, Theorem 3.4 has proved that algorithm RBTH_MINLP is sound and complete for solving MINLPs. Therefore, the following theorem stands.

Theorem 4.5. *Given a MINLP, if there exists a solution, the algorithm RBTH_partition_resolve_mixed can find the solution; if the algorithm finds a solution, then it must be the solution of the MINLP.*

5. Numerical Simulation Results and Comparisons

All of our algorithms are coded in C, and in our simulation, numerical experiments are performed on a PC with Pentium 3.0 GHz Processor and 1.0 GB memory. We first compare our algorithm CPRBTH (RBTH under constraint partition) to two of the best CNLP solvers, Lancelot [20] and SNOPT [21]. To test the performance of the proposed algorithms, computational simulations are carried out with some well-studied benchmark problems taken from the CUTE library [22]. These problems are all minimization problems. Some of these problems were constructed by researchers to test optimization algorithms, while others

Table 2: Results of solving selected MINLP benchmarks from the MacMINLP library. Here nc and nv represent the number of constraints and the number of variables, respectively. "—" means that no feasible solutions were found in the time limit (3600 seconds). Numbers in bold represent the best solutions among the three methods.

Test Problem			MINLP-BB		BARON		CPOPT		CPRBTH	
ID	nc	nv	Sol.	Time	Sol.	Time	Sol.	Time	Sol.	Time
C-RELOAD-q-49	1430	3733	—		—	—	**−1.13**	66.32	**−1.13**	**59.25**
C-RELOAD-q-104	3338	13936	—	—	—	—	**−1.14**	298.87	**−1.14**	**110.22**
Ex12.6.3	57	92	**19.6**	23	**19.6**	423.1	**19.6**	**13.37**	**19.6**	**13.37**
Ex12.6.5	76	130	15.1	4	**10.3**	845.5	10.6	3.19	10.4	**2.1**
PUMP	34	24	—	—	131124	977	**130788**	**81.25**	130789	96.33
SPACE-960-i	6497	5537	—	—	—	—	**7.65E6**	179.12	**7.65E6**	**97.22**
SPACE-960-ir	3617	2657	—	—	—	—	**7.64E6**	132.96	**7.64E6**	**96.34**
SPACE-960	8417	15137	—	—	—	—	**7.84E6**	**1137.13**	**7.84E6**	1311.13
SPACE-960-r	5537	12257	—	—	—	—	**5.13E6**	91.56	**5.13E6**	**87.34**
STOCKCYCLE	97	480	—	—	436341	n/a	119948.7	6.12	**119948.6**	**5.47**
TRIMLON4	24	24	12.2	10	**8.3**	11.0	**8.3**	**1.66**	**8.3**	2.77
TRIMLON6	36	48	18.8	19	**15.6**	1092.9	**15.6**	13.25	**15.6**	**7.92**
TRIMLON12	72	168	—	—	—	—	**95.5**	**298.11**	**95.5**	345.50
TRIMLOSS4	64	105	10.8	99	—	—	**10.6**	8.87	**10.6**	**7.88**
TRIMLOSS5	90	161	12.6	190	—	—	**10.7**	**75.31**	**10.6**	78.87
TRIMLOSS6	120	215	—	—	—	—	**22.1**	**68.19**	**22.1**	**68.19**
TRIMLOSS7	154	345	—	—	—	—	**26.7**	58.14	26.8	**37.22**
TRIMLOSS12	384	800	—	—	—	—	138.8	278.33	**138.1**	199.10

were from real applications, such as computer production planning in operations research. For each instance, the algorithm is independently executed 15 times for comparison. The experimental results are shown in Table 1. The first three columns show the problems IDs, the number of constraints (nc), and the number of variables (nv). The last six columns show the solutions (Sol.) and CPU times we obtain by using LANCELOT, SNOPT and CRBTH. Both the CPU time and the solutions are the average of the measure of the 15 executions of the algorithms. Numerical results indicate that the algorithm usually performs quite well in terms of CPU time and quality of solution found.

We then compare our algorithm with three famous MINLP solvers, MINLP_BB [23], BARON [24], and CPOPT [9]. MINLP_BB performs a branch and bound algorithm with a sequential-quadratic-programming solver for solving continuous problems. BARON is a MINLP solver implementing the branch and reduce algorithm. CPOPT is an MINLP solver implementing the extended saddle point condition under constraint partition algorithm. To test the performance of the proposed algorithms, computational simulations are carried out with some well-studied benchmark problems taken from the MacMINLP library [25]. The first three columns show the problems IDs, the number of constraints (nc), and the number of variables (nv). The last eight columns show the solutions (Sol.) and CPU times we obtain by MINLP_BB, BARON, CPOPT, and CPRBTH. Both the CPU time and the solutions are the average of the measure of the 15 executions of the algorithm. Because the results of MINLP_BB and BARON in [13] were obtained by submitting jobs to the NEOS server and BARON's site, respectively, we accept the results of [13]. The other two solvers were run on a PC with Pentium 3.0 GHz Processor and 1.0 GB memory. The experimental results are shown

in Table 2. Compared to CPOPT, the solutions of CPRBTH are at least competitive, and the running cost of CPRBTH is relatively lower.

6. Conclusion

RBTH is a necessary and sufficient condition for constrained local optima under a range of penalties. In this paper, we first propose three algorithms to solve NLPs using RBTH and then prove that these algorithms are both sound and complete. Additionally, we combine RBTH with constraint partition to solve large-scale MINLPs. Specifically, we decompose the constraints of MINLPs into some easier subproblems that are significant relaxations of the original problem, each of which can be solved by using RBTH directly, and then resolve those violated global constraints across the subproblems via RBTH. In the final part, we present an algorithm for implementing this search procedure and also prove that the algorithm is sound and complete for solving MINLPs under constraint partition. Experimental results also show that our algorithm is both sound and complete.

Acknowledgments

This project was granted by the National Natural Science Foundation of China under Grant nos. 60473042, 60573067, 60803102, and 60773097. The authors are grateful to the anonymous reviewers for providing their detailed, thoughtful, and helpful comments on improving the work presented here.

References

[1] A. M. Geoffrion, "Lagrangean relaxation for integer programming," *Mathematical Programming Study*, no. 2, pp. 82–114, 1974.

[2] J. F. Shapiro, "Generalized Lagrange multipliers in integer programming," *Operations Research*, vol. 19, pp. 68–76, 1971.

[3] T. Wang, *Global Optimization of Constrained Nonlinear Programming*, Ph.D. thesis, Department of Computer Science, University of Illinois, 2000.

[4] B. W. Wah, Y. Chen, and T. Wang, "Simulated annealing with asymptotic convergence for nonlinear constrained optimization," *Journal of Global Optimization*, vol. 39, no. 1, pp. 1–37, 2007.

[5] B. W. Wah and Z. Wu, "The theory of discrete Lagrange multipliers for nonlinear discrete optimization," in *Proceedings of the Principles and Practice of Constraint Programming*, pp. 28–42, Springer, Berlin, Germany, 1999.

[6] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programming," *Operations Research*, vol. 8, pp. 101–111, 1960.

[7] M. A. Duran and I. E. Grossmann, "An outer-approximation algorithm for a class of mixed-integer nonlinear programs," *Mathematical Programming*, vol. 36, no. 3, pp. 307–339, 1986.

[8] W.-X. Gu and B. Li, "Effective method for constrained minimum—reverse bridge theorem," *Computers & Mathematics with Applications*, vol. 56, no. 10, pp. 2629–2637, 2008.

[9] Y. X. Chen, *Solving nonlinear constrained optimization problems through constraint partitioning*, Ph.D. thesis, University of Illinois at Urbana-Champaign, 2005.

[10] B. W. Wah and Y. Chen, "Constraint partitioning in penalty formulations for solving temporal planning problems," *Artificial Intelligence*, vol. 170, no. 3, pp. 187–231, 2006.

[11] B. W. Wah, Y. Chen, and A. Wan, "Constrained global optimization by constraint partitioning and simulated annealing," in *Proceedings of the International Conference on Tools with Artificial Intelligence (ICTAI '06)*, pp. 265–272, 2006.

[12] C. W. Hsu, B. W. Wah, R. Huang, and Y. X. Chen, "Constraint partitioning for solving planning problems with trajectory constraints and goal preferences," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI '07)*, pp. 1924–1929, 2007.

[13] B. W. Wah and Y. Chen, "Solving large-scale nonlinear programming problems by constraint partitioning," in *Proceedings of the Principles and Practice of Constraint Programming (CP '05)*, pp. 697–711, 2005.

[14] B. W. Wah and Y. Chen, "Partitioning of temporal planning problems in mixed space using the theory of extended saddle points," in *Proceedings of the International Conference on Tools with Artificial Intelligence (ICTAI '03)*, pp. 266–273, 2003.

[15] B. W. Wah and Y. X. Chen, "Subgoal partitioning and global search for solving temporal planning problems in mixed space," in *Proceedings of the International Conference on Tools with Artificial Intelligence (ICTAI '04)*, pp. 767–790, 2004.

[16] Y. Chen, B. W. Wah, and C.-W. Hsu, "Temporal planning using subgoal partitioning and resolution in SGPlan," *Journal of Artificial Intelligence Research*, vol. 26, pp. 323–369, 2006.

[17] Y. X. Chen, C. W. Hsu, and B. W. Wah, "SGPlan: subgoal partitioning and resolution in planning," in *Proceedings of the 4th International Planning Competition (IPC '04), International Conference on Automated Planning and Scheduling (ICAPS '04)*, pp. 30–33, 2004.

[18] S. Lee and B. Wah, "Finding good starting points for solving structured and unstructured nonlinear constrained optimization problems," in *Proceedings of the 20th International Conference on Tools with Artificial Intelligence (ICTAI '08)*, pp. 469–476, 2008.

[19] A. R. Conn, N. Gould, and Ph. L. Toint, "Numerical experiments with the LANCELOT package (Release A) for large-scale nonlinear optimization," *Mathematical Programming*, vol. 73, no. 1, pp. 73–110, 1996.

[20] A. R. Conn, N. Gould, and Ph. L. Toint, "Numerical experiments with the LANCELOT package (Release A) for large-scale nonlinear optimization," *Mathematical Programming*, vol. 73, no. 1, pp. 73–110, 1996.

[21] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: an SQP algorithm for large-scale constrained optimization," *SIAM Journal on Optimization*, vol. 12, no. 4, pp. 979–1006, 2002.

[22] I. Bongartz, A. R. Conn, N. Gould, and P. L. Toint, "CUTE: constrained and unconstrained testing environment," *ACM Transactions on Mathematical Software*, vol. 21, no. 1, pp. 123–160, 1995.

[23] S. Leyffer, "Mixed integer nonlinear programming solver," 2002, http://www.mcs.anl.gov/~leyffer/Solvers.html.

[24] N. V. Sahinidis, "BARON: a general purpose global optimization software package," *Journal of Global Optimization*, vol. 8, no. 2, pp. 201–205, 1996.

[25] S. Leyffer, "MacMINLP: AMPL collection of MINLP problems," 2003, http://wiki.mcs.anl.gov/leyffer/index.php/MacMINLP.