Global Optimization and Constraint Satisfaction: The Branch-and-Reduce Approach

Nikolaos V. Sahinidis

University of Illinois, Department of Chemical and Biomolecular Engineering
Urbana, IL 61801, USA
nikos@uiuc.edu
http:/archimedes.scs.uiuc.edu

Abstract. In the early 1990s, we proposed the integration of constraint programming and optimization techniques within the branch-and-bound framework for the global optimization of nonconvex nonlinear and mixed-integer nonlinear programs. This approach, referred to as *branch-and-reduce*, was subsequently supplemented with a variety of branching and bounding schemes. In this paper, we review the theory and algorithms behind branch-and-reduce, its implementation in the BARON software, and some recent successful applications.

1 Introduction

The integration of constraint programming and mathematical programming techniques has generated quite some excitement in the operations research and computer science communities in recent years (cf. [16]). Combining these techniques has been found necessary for the solution of many hard combinatorial optimization problems. In the context of nonlinear programs (NLPs) and mixed-integer nonlinear programs (MINLPs), Ryoo and Sahinidis [30, 31] proposed the first variant of the branch-and-reduce algorithm. This algorithm relied on constraints, interval arithmetic, and duality to draw inferences regarding ranges of integer and continuous variables in an effort to expedite the traditional branch-and-bound algorithm for the global optimization of NLPs and MINLPs. Subsequently, this approach was supplemented with branching schemes that lead to finite search trees while branching in continuous spaces [37, 2], and a number of convexification techniques for the construction of relaxations that enjoy tightness along with robustness and computational efficiency [41, 42, 32, 43]. This methodology has been implemented in the computational system BARON [34] and used in a variety of applications, including chemical process design and operation [30, 20], chip layout and design [10], design of just-in-time manufacturing systems [15], optimization under uncertainty [19, 2], pooling and blending problems [1, 44], and molecular design [35, 36].

In Section 2 of this paper, we state the general mixed-integer nonlinear program addressed by this line of research and review algorithms for its solution. Section 3 describes the theoretical and algorithmic components of the branch-and-reduce approach. The implementation is discussed in Section 4, followed by selective computational results in Section 5. Conclusions are drawn in Section 6.

© Springer-Verlag Berlin Heidelberg 2003

2 Mixed-Integer Nonlinear Programming

We address the problem of finding a globally optimal solution of the following mixed-integer nonlinear program:

$$
\text{(P)} \qquad \begin{array}{ll} \min & f(x) \\ \text{s.t.} & g(x) \le 0 \\ & x_i \in \mathbb{R}, \quad i = 1, \ldots, n_{\mathrm{d}} \\ & x_i \in \mathbb{Z}, \quad i = n_{\mathrm{d}} + 1, \ldots, n \end{array}
$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ and $g : \mathbb{R}^n \mapsto \mathbb{R}^m$.

As special cases of P, one recognizes the classical nonlinear and mixed-integer linear programs. Since both of these problem classes are NP-hard [25, 27], it follows that P is also NP-hard. Yet, P is of interest in a large number of applications, including chemical process synthesis, supply chain management and operation, and molecular design [26, 5, 44].

Initial attempts at the solution of P dealt mostly with problems that are convex when integrality restrictions are dropped [14, 11, 6, 7]. A few recent works have indicated that the application of deterministic branch-and-bound algorithms to the global optimization of general classes of MINLPs is promising [30, 39, 13, 38, 44]. Initially developed in the context of combinatorial optimization problems [18, 9], branch-and-bound was later extended to the more general multi-extremal problem P [12, 17]. To solve P, branch-and-bound computes lower and upper bounds on the optimal objective function value over successively refined partitions of the search space. Partition elements are generated and placed on a list of open partition elements. Elements from this list are selected for further processing and further partitioning, and are deleted when their lower bounds are no lower than the best known upper bound for the problem.

Contrary to the pure integer case, finite termination of this algorithm with an exact global optimum of P is, in general, not guaranteed when branching occurs in continuous variable spaces. The algorithm is convergent under fairly general assumptions on the problem functions [17] and, hence, finitely terminating when an ϵ-optimal solution is acceptable with $\epsilon > 0$. Another challenge associated with solving P with branch-and-bound is the construction of lower bounds. While it is straightforward to obtain a relaxation of a mixed-integer linear program by dropping integrality conditions, the development of lower bounds for P requires the (partial) convexification of functions of continuous variables. A further potential difficulty in solving P stems from the presence of nonlinearities. While LP technology has yielded robust and efficient software codes capable of solving large-scale LPs, only much smaller convex NLPs can be solved reliably. Finally, from the practical point of view, acceptance of MINLP algorithms by practitioners requires integration of these algorithms with modeling languages, such as AMPL and GAMS, and necessitates the development of new language concepts that go well beyond the realm of traditional LP and MILP. The next section addresses some of these challenges.

3 Theoretical and Algorithmic Elements of the Branch-and-Reduce Approach

3.1 Factorable Programming Relaxations

In this subsection, we consider the case of factorable functions f and g, *i.e.*, functions that are recursive sums and products of univariate functions. This class of functions suffices to describe most application areas of interest [24]. Examples of factorable functions include $f(x,y)=xy$, $f(x,y)=x/y$, $f(x,y,z,w)=\sqrt{\exp(xy+z\ln w)z^3}$, $f(x,y,z,w)=\left(x^2y^{0.3}z\right)/w^2+\exp\left(x^2w/y\right)-xy$, $f(x)=\sum_{i=1}^{n}\ln_i(x_i)$, and $f(x)=\sum_{i=1}^{T}\prod_{j=1}^{p_i}\left(c_{ij}^0+c_{ij}x\right)$ with $x\in\mathbb{R}^n$, $c_{ij}\in\mathbb{R}^n$ and $c_{ij}^0\in\mathbb{R}$ $(i=1,\ldots,T;j=1,\ldots,p_i)$.

In his seminar work, McCormick [23] developed bounding techniques for factorable programs. These techniques currently play a central role in many branch-and-bound implementations for problem P. The main observation is that a factorable NLP can be converted to an equivalent separable NLP after the recursive introduction of new variables and constraints [22]. The separable NLP can be relaxed through suitable under- and overestimators of the univariate functions involved. For example, the function $f(x,y,z,w)=\sqrt{\exp(xy+z\ln w)z^3}$, can be decomposed into an almost separable formulation as follows:

$$\begin{array}{ll} x_1 = xy & x_5 = \exp(x_4) \\ x_2 = \ln(w) & x_6 = z^3 \\ x_3 = zx_2 & x_7 = x_5x_6 \\ x_4 = x_1 + x_3 & f = \sqrt{x_7} \end{array}$$

It is straightforward to outer-approximate the univariate functions $\ln$, $\exp$, and $\sqrt{\ }$ over bounded intervals. Bilinear terms can be outer-approximated through their convex and concave envelopes over a rectangle [23, 3]:

$$x_1 \geq \mathrm{convenv}_{[x^L,x^U]\times[y^L,y^U]} = \max\{y^Lx+x^Ly-y^Lx^L, y^Ux+x^Uy-y^Ux^U\}$$
$$x_1 \leq \mathrm{concenv}_{[x^L,x^U]\times[y^L,y^U]} = \min\{y^Lx+x^Uy-y^Lx^U, y^Ux+x^Ly-y^Ux^L\}$$

Several variants of the factorable approach exist. For example, additional variables need not be explicitly introduced. In addition, the decomposition need not proceed until the problem becomes entirely separable. For the example above, non-separabilities in terms of bilinearities were retained. In general, it suffices to proceed only to the extent that the resultant problem can be outer-approximated by a convex feasible set.

3.2 Convexification via Convex Extensions

While factorable programming techniques lead to a completely automatable procedure for the construction of convex lower bounding problems for nonconvex NLPs and MINLPs, these bounding problems often exhibit a large relaxation gap. From the point of view of relaxation quality, it is always advantageous to convexify the original problem functions and constraints to the extent possible. The theory of convex extensions of Tawarmalani and Sahinidis [43] provides a systematic methodology for constructing the closed-form expression of convex envelopes of multidimensional, lower semi-continuous (l.s.c.) functions. This theory provides the capability to construct the convex and concave envelopes of continuous functions. In the sequel, $\bar{\mathbb{R}}$ denotes $\mathbb{R} \cup \{+\infty\}$.

Definition 1 ([43]). *Let C be a convex set and $X \subseteq C$. A convex extension of a function $\phi : X \mapsto \bar{\mathbb{R}}$ over C is any convex function $\eta : C \mapsto \bar{\mathbb{R}}$ such that $\eta(x) = \phi(x)$ for all $x \in X$.*

In other words, an extension is a convex function that agrees with a given (nonconvex) function at each of a predetermined set of points in the domain of definition of both functions. Depending on the original function and this set of points, a convex extension may or may not exist. Furthermore, when a convex extension exists, it need not be unique. In [43], we provide necessary and sufficient conditions for the constructibility of convex extensions.

Key to the development of convex and concave envelopes is the observation that these envelopes are often generated by finitely many sets of points. For instance, in the case of a concave univariate function over an interval, knowledge of the function values at the two endpoints suffices to completely characterize the convex envelope of the function over the interval. This envelope is nothing else but the tightest convex extension of the concave function restricted to the two endpoints. In general, we will refer to this restricted set of points as the *generating set*. One needs to be able to identify the tightest convex extension over this set in order to construct the convex envelope. Working with the convex hull of the epigraph of the convex envelope, allows the latter construction to be easily achieved through disjunctive programming techniques [28], thus leading to the convex envelope in closed-form. This discussion suggests the following two-step procedure for the construction of the convex envelope of a given function:

1. Identify the generating set.
2. Use disjunctive programming to construct the envelope in closed-form.

The main question then becomes how to identify the generating set. We restrict the discussion to functions with compact domains. Let $f(x)$ be the convex envelope of $\phi(x)$ over C, and let F be the epigraph of f. We will use $\text{vert}(F)$ to denote the vertex set of F. Then, the convex envelope of ϕ over C is completely specified by the following set:

$$G_C^{\text{epi}}(\phi) = \big\{x \mid (x, f(x)) \in \text{vert}(F)\big\}.$$

This set is the generating set of the epigraph of function ϕ. The following result characterizes points that do not belong to the generating set:

Theorem 1 ([43]). *Let $\phi(x)$ be a l.s.c. function on a compact convex set C. Consider a point $x_0 \in C$. Then, $x_0 \notin G_C^{\text{epi}}(\phi)$ if and only if there exists a convex subset X of C such that $x_0 \in X$ and $x_0 \notin G_X^{\text{epi}}(\phi)$. In particular, if for an*

ϵ-neighbourhood $N_\epsilon \subset C$ of x_0 it can be shown that $x_0 \notin G^{\text{epi}}_{N_\epsilon}(\phi)$, then $x_0 \notin G^{\text{epi}}_C(\phi)$.

We now illustrate the above process in the context of the multilinear function $\phi(x) = \sum_t a_t \prod_{i=1}^{p_t} x_i$, where $-\infty < l_i \leq x_i \leq u_i < \infty$ for $i = 1, \dots, n$. When all but one of the n variables are fixed, one is left with a line segment over which ϕ is linear. Hence, only the two endpoints of the line segment may belong to the generating set of the convex as well as concave envelope of ϕ. Applying this argument recursively to all variables and using Theorem 1, it follows [43] that the generating set of the convex as well as concave envelope of ϕ is nothing else but the set of extreme points of the hyperrectangle. Let these points be denoted by p_k, $k = 1, \dots, 2^n$, and let ϕ_k be the corresponding values of ϕ at these points. The polyhedral description of the convex outer-approximator of ϕ follows trivially from polyhedral representation theorems (cf. Theorem 4.8, p. 96 in [27]):

$$\begin{aligned} &x = \textstyle\sum_{k=1}^{2^n} \lambda_k p_k, \quad \phi = \sum_{k=1}^{2^n} \lambda_k \phi_k, \quad \sum_{k=1}^{2^n} \lambda_k = 1 \\ &\lambda_k \geq 0, \quad k = 1, \dots, 2^n \end{aligned}$$

The above approach can be readily applied to obtain the convex and concave envelopes of functions of the following forms:

- $\phi(x, y) = M(x_1, x_2, \dots, x_n)/(y_1^{a_1}, y_2^{a_2}, \dots, y_m^{a_m})$ over a hyperrectangle, where M is a multilinear expression, $y_1, \dots, y_m \neq 0$, and $a_1, \dots, a_m \geq 0$. For example, consider $(x_1x_2 + x_3x_4)/(y_1y_2y_3)$.
- $\phi(x, y) = f(x) \sum_{i=1}^n \sum_{j=-p}^k a_{ij} y_i^j$ over a hyperrectangle, where f is concave, $a_{ij} \geq 0$ for $i = 1, \dots, n; j = -p, \dots, k$, and $y_i > 0$. For example, consider $x/y + 3x + 4xy + 2xy^2$.

Additional examples and generalizations of this methodology can be found in [42, 43]. In [44], the theory of convex extensions was used to show that a particular relaxation of the pooling problem dominates earlier ones in the literature. An additional application of this theory is described next.

3.3 Convexification via Product Disaggregation

Throughout this subsection, we consider the following function:

$$\phi(x; y_1, \dots, y_n) = a_0 + \sum_{k=1}^n a_k y_k + x b_0 + x \sum_{k=1}^n b_k y_k$$

where a_k and b_k ($k = 0, \dots, n$) are given constants, $x \in [x^L, x^U]$, and $y_k \in [y_k^L, y_k^U]$. The convex extensions theory can be used to prove the following result:

Theorem 2 ([40]). *Let $H = [x^L, x^U] \times \prod_{k=1}^n [y_k^L, y_k^U]$. Then:*

$$\operatorname{convenv}_H \phi = a_0 + \textstyle\sum_{k=1}^n a_k y_k + x b_0 + \sum_{k=1}^n \operatorname{convenv}_{[y_k^L, y_k^U] \times [x^L \times x^U]}(b_k y_k x).$$

The standard way to bound ϕ is to substitute w for $\sum_{k=1}^n b_k y_k$ and then relax xw using the bilinear envelopes of Subsection 3.1. The required bounds on w follow by maximizing/minimizing $\sum_{k=1}^n b_k y_k$ over $\prod_{k=1}^n [y_k^L, y_k^U]$. While commonplace, such a construction does not yield the convex envelope of $\phi(x, y)$. Consider, for example, $x(2y_1 + 3y_2)$ over $[0, 1]^3$. At $x = 0.5$, $y_1 = 1$, and $y_2 = 0$, the convex envelope equals $x(2y_1 + 3y_2)$ with a value of 1 whereas the standard lower bounding procedure gives a value of 0. Just like the standard lower bounding procedure, the convex envelope of Theorem 2 also makes use of the bilinear convex envelope formula of Subsection 3.1. However, the bilinear envelopes are invoked only after the product is distributed over the summation.

Distribution of the product over the summation results in disaggregating xw into $\sum_{k=1}^n b_k z_k$, where $z_k = xy_k$. This procedure was termed *product disaggregation* in [40] as it is reminiscent of *variable disaggregation*, a procedure that provides tight linear relaxations of mixed-integer linear programs [27]. In [40], we provide a number of applications of product disaggregation, including fractional programs and certain optimization problems that arise from the discretization of dynamic systems.

Theorem 3 ([40]). *Let $f(x; y_1, \dots, y_n)$ be the convex envelope of $\phi(x; y_1, \dots, y_n)$ over $[x^L, x^U] \times \prod_{k=1}^n [y_k^L, y_k^U]$ and let $f_r(x; y_1, \dots, y_n)$ be the convex envelope of $\phi(x; y_1, \dots, y_n)$ over $[x^L, x_r^U] \times \prod_{k=1}^n [y_k^L, y_k^U]$, where $x_r^U < x^U$. Let $K^+ = \{k \mid b_k > 0\}$ and $K^- = \{k \mid b_k < 0\}$. Then, $f_r(x^0; y_1^0, \dots, y_n^0) > f(x^0; y_1^0, \dots, y_n^0)$ if and only if $(x^0; y_1^0, \dots, y_n^0) \in S$ where S is given by:*

$$S = \bigcup_{k \in K^+} \left\{(x, y) \mid (y_k^U - y_k^L)x + (x_r^U - x^L)y_k - x_r^U y_k^U + x^L y_k^L > 0, y_k < y_k^U\right\} \cup \bigcup_{k \in K^-} \left\{(x, y) \mid (y_k^U - y_k^L)x + (x^L - x_r^U)y_k + x_r^U y_k^L - x^L y_k^U > 0, y_k > y_k^L\right\}.$$

A similar result is presented in [40] when the lower bound on x is improved. These results highlight the importance of reducing bounds on x as much as possible when one is interested in deriving convex outer-approximators.

3.4 Polyhedral Outer-Approximation

The current state-of-the-art in linear programming permits the reliable solution of very large-scale LPs in reasonable computational times. On the contrary, nonlinear programs are harder to solve. As a result, it is often advantageous to use polyhedral instead of other convex relaxations in branch-and-bound, even when the latter relaxations are tighter. For this reason, Tawarmalani and Sahinidis [41] developed a polyhedral outer-approximation scheme that generates an entirely linear programming relaxation. The starting point of this approach is a convex nonlinear relaxation obtained by factorable programming and/or convex extensions techniques. Subsequently, a sandwich algorithm is used to provide a polyhedral outer-approximation of the nonlinear functions.

The sandwich algorithm is a template of outer-approximation schemes [8, 29]. At a given iteration, this algorithm begins with a number of points at which tangential outer-approximations of the convex function have been constructed. Then, at every iterative step, the algorithm identifies the interval with the maximum outer-approximation error and subdivides it at a suitably chosen point.

Let $\mathcal{G}(f)$ denote the set of points (x, y) such that $y = f(x)$. Let $\phi^o(x)$ be the outer-approximation of $\phi(x)$, and consider the projective error measure:

$$\epsilon_p = \sup_{p^{\phi^o} \in \mathcal{G}(\phi^o)} \inf_{p^{\phi} \in \mathcal{G}(\phi)} \left\{ \|p^{\phi} - p^{\phi^o}\| \right\}.$$

Given a number of outer-approximators, the strategy developed in [41] constructs the next outer-approximator as the line supporting ϕ at a point where the maximum projective error occurs. This scheme converges quadratically:

Theorem 4 ([41]). *Consider the univariate function $\phi : [x^L, x^U] \mapsto \mathbb{R}$. Let $R = \big(x^L, \phi(x^L)\big)$ and $S = \big(x^U, \phi(x^U)\big)$. Assume that the tangents at R and S intersect at O. Let θ be $\pi - \angle ROS$, and $L = |RO| + |OS|$. Let $k = L\theta/\epsilon_p$, where ϵ_p is the maximum allowable projective error. Then, the algorithm needs at most $\lceil \sqrt{k} - 2 \rceil$ supporting lines to achieve the required accuracy.*

3.5 Branch-and-*Reduce*

The previous subsections have illustrated that the quality of the relaxations thus obtained is a strong function of the bounds of variables that participate in nonlinear relationships. Thus, tighter variable bounds imply tighter relaxations and can be expected to lead to faster convergence of branch-and-bound. Our approach to global optimization places a strong emphasis on the derivation of tight bounds for all problem variables. In each node of the search tree, constraint programming techniques are utilized in a *preprocessing* step to reduce ranges of problem variables before a relaxation is constructed. Once the relaxation is solved, a *postprocessing* step utilizes the solution of the relaxed problem in an attempt to further reduce ranges of variables before branching occurs. Precisely because so much emphasis is placed on the reduction of ranges of problem variables, we refer to the overall algorithm as a branch-and-*reduce* approach. The main reduction strategies used in our framework are outlined next.

3.6 Drawing Inferences from Constraints: Feasibility-Based Range Reduction

Feasibility-based tightening, or feasibility-based range reduction, is a process that relies on the problem constraints to cut-off infeasible portions of the solution space. Assume, for example, that the following constraints are part of the problem to be solved at a given node:

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, ..., m.$$

To tighten variable bounds based on these linear constraints, one could simply solve the $2n$ LPs:

$$\left\{ \min \pm x_k \text{ s.t. } \sum_{j=1}^{n} a_{ij} x_j \leq b_i, i = 1, \ldots, m \right\}, k = 1, \ldots, n, \tag{1}$$

which would provide tightening that is optimal, albeit computationally expensive. An alternative approach is based on the observation that one of the constraints

$$\begin{cases} x_h \leq \frac{1}{a_{ih}} \left(b_i - \sum\limits_{j \neq h} \min \left\{ a_{ij} x_j^U, a_{ij} x_j^L \right\} \right), & a_{ih} > 0 \\ x_h \geq \frac{1}{a_{ih}} \left(b_i - \sum\limits_{j \neq h} \min \left\{ a_{ij} x_j^U, a_{ij} x_j^L \right\} \right), & a_{ih} < 0 \end{cases} \tag{2}$$

is also valid for each pair (i, h) that satisfies $a_{ih} \neq 0$. The constraints in (2) function as "poor man's linear programs," particularly when they are applied iteratively, looping over the set of variables several times. Constraints (2) have been used extensively in the constraint programming literature [16] as well as in the mixed-integer linear programming literature [4]. There are well-known pathological situations in which these constraints do not provide optimal or even any tightening. Shectman and Sahinidis [37] experimented with these constraints in the context of concave minimization over polytopes demonstrating that these constraints often provide optimal tightening, particularly when the full LPs (1) are solved once at the root node of the search tree. Our current approach is to solve, at every node of the search tree, the full LPs (1) for a few judiciously selected variables and aggressively apply the approximate strategy (2) to all variables.

The above approach can be extended to the case of nonlinear constraints, giving rise to "poor man's nonlinear programs," an approach particularly easy to implement in the context of factorable and separable nonlinear programs.

3.7 Drawing Inferences from Optimal Solutions: Optimality-Based Range Reduction

Optimality-based range reduction recognizes that dual solutions of the relaxation solved at any node of the search tree provide information about the shape of the value function of the relaxed problem. This information can be used to construct an underestimator of the value function of the relaxed problem that, in turn, underestimates the value function of the nonconvex problem. Requiring the so-constructed underestimator to take values below that of the current best known solution, leads to inferences regarding inferior parts of the search space.

In their simplest form, optimality-based inferences can be drawn about variable ranges as shown by Ryoo and Sahinidis [30]. Assume that the simple range constraint $x \leq x^U$ is active at a relaxed problem solution with a corresponding

Lagrange multiplier of $\lambda > 0$. Let L and U denote the objective function values of the relaxed problem's solution and the incumbent, respectively. Clearly, then, $L - \lambda(x - x^U)$ provides a first-order underestimator of the relaxation value function. Requiring this underestimator to take better values than U, gives the simplest optimality-based range reduction cut:

$$x \geq x^U - \frac{U - L}{\lambda}.$$

Similar inferences can be drawn about variables that, at the relaxed problem solution, go to their lower bounds. If a variable is at neither of its bounds in the relaxed problem solution, we can *probe* its bounds by temporarily fixing this variable at (or close to) its lower (upper) bound, constructing the linear underestimator of the value function, and contracting the range of the variable using the dual solution thus obtained. This process requires the solution of additional relaxations and, in certain cases, is less advantageous than solving the full range contraction LPs or NLPs discussed in the previous section.

In [30, 31], the above process of range contraction is extended to arbitrary constraints of the type $g(x) \leq 0$, or even to sets of constraints that may or may not be active at the relaxed problem solution. One is then able to infer valid inequalities, some of which may be nonconvex. Although they may exclude solutions that are feasible to P, these inequalities do not exclude any solutions of P with objective function values better than U.

3.8 A Unified Framework for Constraint Inferencing

The range reduction schemes of the two previous subsections involve the solution of some optimization problem. In the case of feasibility-based reduction, an optimization problem such as (1) is solved either approximately or exactly. In the case of optimality-based reduction, one solves a relaxation. Observe that these optimization problems are very closely related. In the simple case of a linearly constrained global optimization problem, the feasible space of these optimization problems is identical. Thus, any feasible dual solution of the relaxation is also dual feasible to the range reduction problem (1) and *vice versa*. Hence, solutions obtained for one problem can be used for range reduction in the other problem. In Chapter 6 of [44], Tawarmalani and Sahinidis build on this observation to provide a unified range reduction theory that subsumes the feasibility-based and optimality-based range reduction schemes of the two previous subsections as well as a variety of such schemes from the literature [21, 4, 30, 31, 46].

3.9 Node Selection and Branching

In contrast to branching on 0−1 variables, branching on continuous variables may not lead to finite partitioning. Consequently, the lower and upper bounding sequences generated by the algorithm are, in general, only convergent in the latter case. To ensure convergence, we use a *bound-improving* node selection rule by selecting a partition element with the current lowest bound every finitely many iterations. In addition, by periodically bisecting the longest edge amongst all nonlinear variables, we ensure an *exhaustive* partitioning scheme, *i.e.*, one that guarantees that partition elements converge to points or other sets over which P is easily solvable. This strategy guarantees convergence of the overall algorithm [17]. We have shown that, when applied to concave minimization over polytopes, this algorithm is finite as long as one branches on the incumbent solution when possible. For two-stage stochastic programs with integer variables in the second stage, we have shown that branching in the space of the "tender variables" renders this algorithm finite for this problem class as well [2].

We rely exclusively on rectangular partitioning. A variable is chosen and its range partitioned at the relaxation solution—unless bisection is required for convergence or branching on the incumbent is possible. Note, however, that when factorable relaxations are used, the problem is reformulated in a higher dimensional space. Branching on the reformulation variables may then result to partitions that are not rectangular in the original problem space. Finally, we note that the process for selecting the branching variable accounts for all deviations of outer-approximators from original nonlinear problem functions for which a variable is responsible for. Further, we account for the current relaxation problem solution and potential for its improvement as detailed in Chapter 6 of [44] in order to compute branching priorities. The variables with the largest branching priorities are considered candidates for probing.

3.10 Finding the K Best or All Feasible Solutions

Consider an optimization problem with k integer variables, x_i, $i = 1, \ldots, k$. For simplicity, we assume $0 \leq x_i \leq x_i^U$, $i = 1, \ldots, k$. It is common practice to identify multiple solutions of such a problem in one of the two following ways:

– Given a solution x^*, introduce the following *nonlinear* cut:

$$\sum_i (x_i^* - x_i)^2 \geq 1.$$

– Reformulate the problem by introducing binary variables:

$$x_i = \sum_{j=1}^{\lfloor log_2(x_i^U) \rfloor} 2^{j-1} y_{ij}, \quad i = 1, \ldots, k$$

The solution y^* can be excluded by the well-known *linear* integer cut:

$$\sum_{(i,j)\in\mathcal{B}^*} y_{ij} - \sum_{(i,j)\in\mathcal{N}^*} y_{ij} \leq |\mathcal{B}^*| - 1$$

where $\mathcal{B}^* = \{(i,j)|y_{ij}^* = 1\}$ and $\mathcal{N}^* = \{(i,j)|y_{ij}^* = 0\}$.

Several solutions of the problem can then be obtained by solving a series of models in which integer cuts are successively introduced to exclude the previous models' optimal solutions from further consideration. However, such an approach requires the search of a number of branch-and-bound trees.

Instead of using the integer cuts above, we modify the standard node fathoming step of the algorithm. Instead of deleting all inferior nodes when a feasible solution is found, we delete only the current node when it becomes infeasible or a point. Nodes where feasible solutions are identified are branched further until they become points in the search space or infeasible. All feasible solutions can be identified through a single application of branch-and-reduce.

Once the fathoming step of the algorithm has been modified as above, the optimality-based range reduction and probing techniques of Subsection 3.7 must also be modified. These techniques require an appropriate upper bound for optimality-based fathoming. Instead of using the upper bounds provided by feasible solutions identified during the search, we make use of a bound obtained by solving a linear relaxation of the problem.

Note that it is straightforward to modify this algorithm to provide only the K best solutions and that this scheme will work well in continuous spaces provided that the sought-after solutions are isolated (separated by a certain distance).

4 The BARON Computational System

The Branch-And-Reduce Optimization Navigator (BARON) implements the algorithms described above by combining branch-and-bound with constraint propagation and duality techniques for reducing ranges of variables in the course of the algorithm. From the very beginning, BARON was developed as a *user-configurable system* that could be easily modified by users to allow experimentation with different lower bounding, branching, and other algorithmic options.

The first version of BARON was merely 1800 lines of code written in the GAMS modeling language in 1991-93. The software evolved into 10,000 lines of FORTRAN 77 in 1994-95. Currently, BARON is a mix of about 42,000 lines in FORTRAN 90 and 24,000 lines in C. It still serves as a system that facilitates experimentation with novel branch-and-bound algorithms for global optimization. In addition, it provides a modeling language and a completely automated way for solving NLPs and MINLPs to global optimality. The latter option is also offered under the GAMS modeling system as illustrated in Chapter 11 of [44]. Other components of the system include an automatic function evaluator and differentiator, sparse matrix utilities, data manager, and links to solvers for the solution of LP, NLP, and SDP subproblems. While no complete rounding error control is currently attempted, an IEEE exception handler has been fully developed for objective and constraint function calculations for local search and lower bounding purposes.

5 Computations

Extensive computations with the proposed algorithm on over 500 problems are reported in [44]. Table 1 presents computational results with selected problems on a 332 MHz RS/6000 Model 43P with 128MB RAM and a LINPACK score of 59.9. For all problems solved, we report the total CPU seconds taken to solve the problem (T_{tot}), the total number of nodes in the branch-and-reduce tree (N_{tot}), and the maximum number of nodes that had to be stored in memory during the search (N_{mem}). Computations were carried out with an absolute termination tolerance (difference between upper and lower bounds) of 10^{-6}.

The problems of Table 1 include pooling problems (adhya1 to adhya4), a nonlinear fixed-charge problem (e27), a reliability problem (e29), a mechanical fixture design problem (e31), a heat exchanger network synthesis problem (e35), a pressure design problem (e38), a truss design problem (e40), a problem from the design of just-in-time manufacturing systems (jit), a particulary challenging molecular design problem (primary), and two difficult problems from discretization of dynamical systems (tiny, catmix).

Finally, Figure 1 shows results for the robot problem [45], a set of 8 quadratic equations in 8 unknowns. We utilize BARON's `numsol` option to identify solutions within an isolation tolerance of 10^{-4}. For `numsol` $= 1$, the algorithm requires 10 nodes to obtain the solution. For `numsol` $= 16$, 334 nodes are required, *i.e.*, approximately 21 nodes per solution found. For `numsol` ≥ 17, only 16 solutions are obtained thus proving that this problem has exactly 16 solutions.

Table 1. Selected computational results for problems from [44]

Problem	Obj.	m	n	n_{d}	T_{tot}	N_{tot}	N_{mem}
e27	2.00	2	1	1	0.02	3	2
e40	30.41	7	4	3	0.15	24	6
e29	-0.94	6	2	2	0.18	47	11
e38	7197.73	3	4	2	0.38	5	2
jit	173,983	33	26	4	0.67	63	10
adhya1	-549.80	52	13	0	1.20	5	1
adhya4	-877.65	67	18	0	1.35	1	1
adhya2	-549.80	69	13	0	1.75	11	1
adhya3	-561.05	78	20	0	1.95	5	1
e36	-246.00	2	2	1	2.59	768	72
e31	-2.00	135	112	24	3.75	351	56
e32	-1.43	18	35	19	13.7	906	146
e35	64868.10	39	32	7	16.4	465	57
primary	-1.2880	164	82	58	375	15930	1054
tiny	1.00594	96	71	16	1110	3728	244
catalyst	-0.01637	32	33	0	3540	3477	480

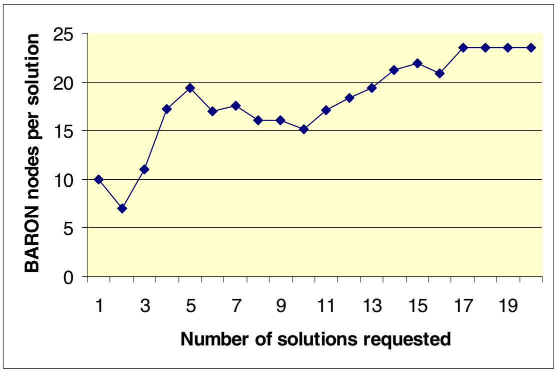

Fig. 1. BARON nodes per solution for the robot problem for different `numsol` values

6 Conclusions

The problems in Table 1 have been sorted in order of increasing time. Looking at this table, one observes that size alone is not a good indicator of problem difficulty. Nor is the number of integer variables. From the results of this table, it is clear that problems with up to a few hundred variables and constraints are solvable with the general-purpose BARON system. In [20, 37, 33, 44], we report computational results on problems with up to a few thousand variables with specialized implementations of branch-and-reduce. The problems of Table 1 are coming from a very wide variety of applications, demonstrating the broad applicability of the algorithms described in this paper.

Acknowledgements

Partial financial support from the National Science Foundation under awards ECS 00-98770, DMI 01-15166, and CTS 01-24751 is gratefully acknowledged.

References

[1] N. Adhya, M. Tawarmalani, and N. V. Sahinidis. A Lagrangian approach to the pooling problem. Industrial & Engineering Chemistry, 38:1956-1972, 1999. 1

[2] S. Ahmed, M. Tawarmalani, and N. V. Sahinidis. A finite branch and bound algorithm for two-stage stochastic integer programs. Mathematical Programming. Submitted, 2000. 1, 10

[3] F. A. Al-Khayyal and J. E. Falk. Jointly constrained biconvex programming. Mathematics of Operations Research, 8:273-286, 1983. 3

[4] D. E. Andersen and K. D. Andersen. Presolving in linear programming. Mathematical Programming, 71:221-245, 1995. 8, 9

[5] L. T. Biegler, I. E. Grossmann, and A. W. Westerberg. Systematic Methods of Chemical Process Design. Prentice Hall, Upper Saddle River, New Jersey, 1997. 2

[6] B. Borchers and J. E. Mitchell. An improved branch and bound for mixed integer nonlinear programs. Comput. Oper. Res., 21:359-367, 1994. 2

[7] B. Borchers and J. E. Mitchell. A computational comparison of branch and bound and outer approximation algorithms for 0-1 mixed integer nonlinear programs. Comput. Oper. Res., 24:699-701, 1997. 2

[8] R. E. Burkard, H. Hamacher, and G. Rote. Sandwich approximation of univariate convex functions with an application to separable convex programming. Naval Research Logistics, 38:911-924, 1992. 6

[9] R. J. Dakin. A tree search algorithm for mixed integer programming problems. Computer Journal, 8:250-255, 1965. 2

[10] M. C. Dorneich and N. V. Sahinidis. Global optimization algorithms for chip layout and compaction. Engineering Optimization, 25:131-154, 1995. 1

[11] M. A. Duran and I. E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Mathematical Programming, 36:307-339, 1986. 2

[12] J. E. Falk and R. M. Soland. An algorithm for separable nonconvex programming problems. Management Science, 15:550-569, 1969. 2

[13] C. A. Floudas. Deterministic Global Optimization: Theory, Algorithms and Applications. Kluwer Academic Publishers, Dordrecht, 1999. 2

[14] O. K. Gupta and A. Ravindran. Branch and bound experiments in convex nonlinear integer programming. Management Science, 31:1533-1546, 1985. 2

[15] R. A. Gutierrez and N. V. Sahinidis. A branch-and-bound approach for machine selection in just-in-time manufacturing systems. International Journal of Production Research, 34:797-818, 1996. 1

[16] J. Hooker. Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction. John Wiley & Sons, New York, NY, 2000. 1, 8

[17] R. Horst and H. Tuy. Global Optimization: Deterministic Approaches. Springer Verlag, Berlin, Third edition, 1996. 2, 10

[18] A. H. Land and A. G. Doig. An automatic method for solving discrete programming problems. Econometrica, 28:497-520, 1960. 2

[19] M. L. Liu and N. V. Sahinidis. Process planning in a fuzzy environment. European J. Operational Research, 100:142-169, 1997. 1

[20] M. L. Liu, N. V. Sahinidis, and J. P. Shectman. Planning of chemical process networks via global concave minimization. In I. E. Grossmann (ed.), Global Optimization in Engineering Design, Kluwer Academic Publishers, Boston, MA, pages 195-230, 1996. 1, 13

[21] O. L. Mangasarian and L. McLinden. Simple bounds for solutions of monotone complementarity problems and convex programs. Mathematical Programming, 32:32-40, 1985. 9

[22] G. P. McCormick. Converting general nonlinear programming problems to separable nonlinear programming problems. Technical Report T-267, The George Washington University, Washington D. C., 1972. 3

[23] G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I-Convex underestimating problems. Mathematical Programming, 10:147-175, 1976. 3

[24] G. P. McCormick. Nonlinear Programming: Theory, Algorithms and Applications. John Wiley & Sons, 1983. 3

[25] K. G. Murty and S. N. Kabadi. Some NP-complete problems in quadratic and nonlinear programming. Mathematical Programming, 39:117-129, 1987. 2

[26] S. V. Nabar and L. Schrage. Formulating and solving business problems as nonlinear integer programs. Technical report, Graduate School of Business, University of Chicago, 1992. 2

[27] G. L. Nemhauser and L. A. Wolsey. Integer and Combinatorial Optimization. Wiley Interscience, Series in Discrete Mathematics and Optimization, 1988. 2, 5, 6

[28] R. T. Rockafellar. Convex Analysis. Princeton Mathematical Series. Princeton University Press, 1970. 4

[29] G. Rote. The convergence rate of the sandwich algorithm for approximating convex functions. Computing, 48:337-361, 1992. 6

[30] H. S. Ryoo and N. V. Sahinidis. Global optimization of nonconvex NLPs and MINLPs with applications in process design. Computers & Chemical Engineering, 19:551-566, 1995. 1, 2, 8, 9

[31] H. S. Ryoo and N. V. Sahinidis. A branch-and-reduce approach to global optimization. Journal of Global Optimization, 8:107-139, 1996. 1, 9

[32] H. S. Ryoo and N. V. Sahinidis. Analysis of bounds for multilinear functions. Journal Global Optimization, 19:403-424, 2001. 1

[33] H. S. Ryoo and N. V. Sahinidis. Global optimization of multiplicative programs. Journal of Global Optimization. Accepted, 2002. 13

[34] N. V. Sahinidis. BARON: A general purpose global optimization software package. Journal of Global Optimization, 8:201-205, 1996. 1

[35] N. V. Sahinidis and M. Tawarmalani. Applications of global optimization to process and molecular design. Computers & Chemical Engineering, 24:2157-2169, 2000. 1

[36] N. V. Sahinidis, M. Tawarmalani, and M. Yu. Design of alternative refrigerants via global optimization. AIChE J. Accepted, 2003. 1

[37] J. P. Shectman and N. V. Sahinidis. A finite algorithm for global minimization of separable concave programs. Journal of Global Optimization, 12:1-36, 1998. 1, 8, 13

[38] H. D. Sherali and H. Wang. Global optimization of nonconvex factorable programming problems. Mathematical Programming, 89:459-478, 2001. 2

[39] E. M. B. Smith and C. C. Pantelides. Global optimisation of general process models. In I. E. Grossmann (ed.), Global Optimization in Engineering Design, Kluwer Academic Publishers, Boston, MA, pages 355-386, 1996. 2

[40] M. Tawarmalani, S. Ahmed, and N. V. Sahinidis. Product disaggregation and relaxations of mixed-integer rational programs. Optimization and Engineering, 3:281-303, 2002. 5, 6

[41] M. Tawarmalani and N. V. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. Mathematical Programming. Submitted, 1999. 1, 6, 7

[42] M. Tawarmalani and N. V. Sahinidis. Semidefinite relaxations of fractional programs via novel techniques for constructing convex envelopes of nonlinear functions. Journal of Global Optimization, 20:137-158, 2001. 1, 5

[43] M. Tawarmalani and N. V. Sahinidis. Convex extensions and convex envelopes of l.s.c. functions. Mathematical Programming, 93:247-263, 2002. 1, 4, 5

[44] M. Tawarmalani and N. V. Sahinidis. Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms,

Software, and Applications, volume 65 of Nonconvex Optimization and Its Applications. Kluwer Academic Publishers, Dordrecht, 2002. 1, 2, 5, 9, 10, 11, 12, 13

[45] L.-W. Tsai and A. P. Morgan. Solving the kinematics of the most general sixand five-degree-of-freedom manipulators by continuation methods. ASME J. of Mechanisms, Transmissions and Automation in Design, 107:48-57, 1985. 12

[46] J. M. Zamora and I. E. Grossmann. A branch and contract algorithm for problems with concave univariate, bilinear and linear fractional terms. Journal of Global Optimization, 14:217-249, 1999. 9