Optimization Methods and Software
Vol. 00, No. 00, Month 200x, 1–22

REVIEW

Interval Computations, Rigor and Non-Rigor in Deterministic Continuous Global Optimization

Ralph Baker Kearfott, Department of Mathematics, University of Louisiana, U.L. Box 4-1010, Lafayette, Louisiana, 70504-1010, USA (rbk@louisiana.edu).

(*Received 00 Month 200x; in final form 00 Month 200x*)

Deterministic branch and bound methods for the solution of general nonlinear programs have become increasingly popular during the last decade or two, with increasing computer speed, algorithmic improvements, and multiprocessors. There are presently several commercial packages. Although such packages are based on exhaustive search, not all of them rigorously take account of roundoff error, singularities, and other possibilities. Popular non-rigorous packages have much in common with mathematically rigorous packages, including overall structure and basic techniques. Nonetheless, it is not trivial to make non-rigorous packages rigorous. We

(1) Define different kinds of answers that global optimization software can claim to provide.
(2) Explain where rigor might be needed and where it is not practical.
(3) Briefly review salient techniques common to deterministic branch and bound methods.
(4) Illustrate pitfalls in non-rigorous branch and bound methods.
(5) Outline some of the techniques to make non-rigorous software rigorous, and provide guidance for research into and implementation of these techniques.
(6) Provide some theoretical backing, with examples, for convergence of common relaxation techniques.

Keywords: deterministic global optimization, mathematical rigor, relaxations, branch and bound, interval computations

1. Introduction

Global optimization as a field has advanced considerably during the past several decades, and, in particular, during the past ten years. This is evidenced by an exponential increase in the number of publications, commercial success of general software packages, and, especially during the past ten years, success at resolving practical problems that other methods could not. The developments have been both theoretical (analysis of optimality conditions, classes of problems, and complexity analysis) and practical (algorithmic development, solution of important problems).

From the 1950's to the 1970's, due to the speed of computing equipment and need for better understanding of algorithms, global optimization was largely impractical. When an optimum value, subject to constraints or not, was required, globality was out of reach except in certain cases, such as when convexity could be proven. During this era, efficient algorithms for finding a local optimum (such as are described in [7] or [16]) were developed.

Serious attempts to find global, as opposed to local, optimizers and optima began in the 1970's. Important benchmarks in this endeavor are [8] and [9] However, until relatively recently, the most common techniques in global optimization have been of a statistical nature, such as the tunneling method [35], or, more commonly genetic algorithms such as described in [2, 17], or simulated annealing introduced in [31, 32] and as described in [15, 46]. Such algorithms are not guaranteed to find

ISSN: 1055-6788 print/ISSN 1029-4937 online

DOI: 10.1080/03081080xxxxxxxxx
http://www.informaworld.com

global optima, but, under certain assumptions, will do so with a high probability. Although there is no guarantee that the answer will be near an actual optimizer or optimum, such statistically-based algorithms can be tuned to complete in a reasonable amount of time, and will supply parameter values corresponding to a feasible point with a relatively low objective value.

In general software for global optimization, alternatives to statistical techniques that are guaranteed to complete with a good approximation to a global optimum (often termed *deterministic* global optimization methods) have been impractical for all but the simplest problems until the past decade. Such alternatives involve very careful *relaxation* of the problem, a systematic search of the domain (sometimes termed *complete search*), or a combination of the two. However, during the past decade, software based on these techniques has become more common. A comparison of complete search techniques for global optimization appears in [40]. (A reference that reviews techniques for relaxations is [14]).

Much software based on complete search (branch and bound) algorithms is theoretically guaranteed to find the global optimum and, depending on the algorithm, all optimizing points, in the absence of roundoff error and error in certain approximations. Nonetheless, there are cases in practice where such algorithms complete with an answer that is not near a global optimizing point, or in which such software completes without finding all optimizing points or with any indication that this is so. However, it is possible to design branch and bound software in which all sources of error are taken into account, so that, when such software delivers bounds on the optimum and bounds on the global optimizers, that constitutes a mathematical proof that the optimum and all such optimizers lie within those bounds. Unfortunately, such software, with exceptions, has not been competitive with similar branch and bound software that does not attempt to be mathematically rigorous.

In summary, we have the following overall hierarchy of algorithms and software for global optimizations. We list these in increasing order of difficulty and in decreasing scope of application.

(1) Procedures to find an approximate local optimizing point
(2) Procedures that use statistical methods or heuristics to find points that are possibly globally optimal.
(3) Procedures that use structure, search, or both, to systematically find global optima, assuming converging iterations have converged and the computer arithmetic is exact.
(4) Procedures that provide mathematically rigorous bounds on the global optimum and optimizing points.

In [40] and elsewhere, the salient non-rigorous global optimization software (and BARON [45] in particular) apparently could solve more problems more efficiently than software designed to be mathematically rigorous (and, in particular, our `GlobSol` [26, 30] software). Even so, the basic structure and most, if not all, of the underlying techniques in the non-rigorous branch and bound methods are similar or identical to those used in mathematically rigorous branch and bound methods. This review analyzes these techniques and contrasts rigorous and non-rigorous branch and bound methods for constrained global optimization problems. We review how these techniques can be made rigorous, and present guidelines for future development and study.

In §2, we introduce notation and discuss different practical situations in which different kinds of answers (e.g. local or global) are admissible. In §3, we outline techniques in branch and bound methods, contrasting these techniques when used in non-rigorous versus rigorous branch and bound methods. In §4, we present consid-

erations in implementing these techniques in a rigorous way. In §5, we do a general analysis of convergence of solutions of relaxations to solutions of the original problem, and we give examples of how to apply this theory. We draw conclusions and outline next steps in §6.

2. Types of Answers: What is practical?

The performance of an algorithm depends on the type of problem it is trying to solve, and different optimization software solves, essentially, different kinds of problems. Furthermore, if software claims to be mathematically rigorous, the problem for which it claims to give mathematically rigorous bounds should be clearly defined. Finally, the minimum requirements of a solution differ from application to application. The purpose of this section is to clarify these concepts.

We begin with our notation for the most general problem instance.

2.1. *The General Global Optimization Problem*

All problems we will consider will be of the form

$$\boxed{\begin{array}{l}\text{minimize } \varphi(x)\\ \text{subject to } c_i(x) = 0, i = 1, \ldots, m_1,\\ \qquad\qquad g_i(x) \le 0, i = 1, \ldots, m_2,\\ \text{where}\\ \varphi : \mathbf{D}_\varphi \subseteq \mathbb{R}^n \to \mathbb{R},\ c_i : \mathbf{D}_{c_i} \subseteq \mathbb{R}^n \to \mathbb{R},\\ \text{and } g_i : \mathbf{D}_{g_i} \subseteq \mathbb{R}^n \to \mathbb{R}.\end{array}} \tag{1}$$

Often, bounds on the search region are given by

$$\boldsymbol{x} = ([\underline{x}_1, \overline{x}_1], \ldots [\underline{x}_n, \overline{x}_n])^T;$$

the region defined by such bounds $\boldsymbol{x}$ is called a *box*. Depending on the algorithm, the bounds $\boldsymbol{x}$ can be considered either as hard *bound constraints* (that is, as inequality constraints g_i of a special form), or as artificially imposed limits to tell the branch and bound algorithm where to look.

2.2. *Differing Requirements*

The following scenarios, although somewhat simplified here, occur in practice.

(1) φ is the cost of running a (nominally) \$50,000,000 per month plant.
In this scenario, the plant manager would like the smallest possible operating cost, but would be happy with a 5% lower cost than before.

(2) φ represents the potential energy of a particular conformation of a molecule.
In this scenario, the globally lowest value for φ gives the most information, but local minima give some information, and finding the global minimum may not be practical.

(3) A mathematician has reduced a proof to showing that $\varphi \ge 1$ everywhere
In this scenario, the global minimum must not only be found, but also must be rigorously proven to be so. However, only a mathematically rigorous lower bound on the global optimum is needed, whereas bounds on all global optimizers are not.

(4) A portfolio manager has estimated the expected rate of return and a risk measure for various stocks, and has a particular amount to invest among the stocks. The portfolio manager would like to allocate the investment to minimize the total risk, subject to a lower bound on the total rate of return.

If the global optimizer is not unique, the portfolio manager would like to know that, since he may have other criteria for choosing stocks that weren't included in the original model. In particular, a useful description of any parametrized set of solutions would be helpful.

In scenario (1), a few iterations of a steepest descent method, or of a derivative-free method such as described in [7] or [16] will suffice. This is fortunate, since evaluation of the objective in such a problem might be the result of a lengthy "black box" simulation, output from a set of computer programs that solve large systems of partial differential equations. However, additional developments may lead to the ability to compute parameters (i.e. approximate optimizers) that are more nearly globally optimal, leading to even more cost savings.

In scenario (2), the set of global optimizers (rather than just the optimum value) is of primary interest. Furthermore, the entire set (not just one or two optimizers) is of interest, not only just one or two points, and local optimizers whose objective value is small (but perhaps not globally optimal) are of interest[1]. It would be nice to have a mathematical proof that all such local minimizers have been found, but this, so far, has been beyond the capabilities of branch and bound and relaxations, with current models. See perhaps [13] for overviews of modeling and optimization of this problem.

Scenario (4), in which not only all global minimizers are required, but in which the problem may exhibit singularities (e.g. non-isolated sets of global minimizers) is in principle the most difficult. This is particularly so if mathematical guarantees are required. This type of problem benefits greatly from special techniques to handle the singularities, yet the subset of this set of problems that can be treated with mathematical rigor is limited. In contrast, scenario (3) is significantly easier in practice for mathematically rigorous methods[2]. A type of problem intermediate in difficulty between (3) and (4) is one in which all global minimizers are desired, but in which there are no singularities, that is, in which the global minimizers are isolated and associated derivative matrices are non-singular.

Of course, the actual difficulty of a problem arising from a particular application depends on the structure of the problem, as much as on the kind of answer that is needed. Also, applications of global optimization are wide-ranging, and it is not possible to list here the numerous variations of these scenarios and algorithms. Our intent is to delineate some principles determining the practicality of our hierarchy of algorithms.

2.3. *Epsilon-approximate Solutions*

Another type of answer that is sometimes easier than other types of answers for automatically verified software is a guarantees that a point in the parameter space corresponds to an approximate solution. For example, solution to a decision problem derived from a relaxation of the original problem may be sufficient, where a

[1]In fact, certain diseases, such as mad cow disease, are linked to molecular conformations corresponding to potential energy minimizers other than the usual one. Furthermore, in actual populations of molecules, various conformations corresponding to various local minima have been observed.

[2]This is borne out in experiments. One reason is that the search may stop once tight bounds on the optimum are known, whereas the search must subdivide the entire region if bounds on all optimizers are needed.

relaxation is defined as follows.

DEFINITION 2.1 *A* relaxation *of an optimization problem of the form* (1) *is an optimization problem whose feasible set contains the feasible set of the original problem and whose objective function is less than or equal to the objective function of the original problem.*

Thus, the global optimum of a relaxation is no greater than the global optimum of the original problem.

Relaxations are typically obtained by increasing the size of the feasible set e.g. by replacing the g_i by smaller ones or by replacing the objective function by a smaller one. If we replace the constraints to make the feasible region larger, we speak of *relaxing the constraints*.

Consider the following decision problem obtained by relaxing the constraints of our general problem (1), and seeking any point satisfying the relaxed constraints and whose objective is within a tolerance of the optimum of the original problem.

$$\boxed{\begin{array}{l}\text{Find points } x \text{ such that}\\ \varphi(x) - \underline{\varphi} < \epsilon_\varphi,\\ \text{where } \underline{\varphi} \text{ is the solution to (1), and such that}\\ |c_i(x)| \le \epsilon_c, \quad i = 1, \ldots, m_1, \text{ and}\\ g_i(x) \le \epsilon_g, \quad i = 1, \ldots, m_2.\end{array}} \tag{2}$$

Finding a set of bounds $\boldsymbol{x}$ such that every point within the bounds is proven to satisfy (2) is usually easier than finding a set of bounds $\hat{\boldsymbol{x}}$ such that a point in $\hat{\boldsymbol{x}}$ that satisfies (1) is proven to exist.

A complementary problem, important for eliminating regions in branch and bound algorithms, is

$$\boxed{\begin{array}{l}\text{Given a box } \boldsymbol{x}, \text{ show that at least one of the following is true for all } x \in \boldsymbol{x}.\\ \bullet\ \varphi(x) - \underline{\varphi} > \underline{\epsilon}_\varphi, \text{ where } \underline{\varphi} \text{ is the solution to (2), or}\\ \bullet\ |c_i(x)| > \underline{\epsilon}_c \text{ for at least one } i \text{ between 1 and } m_1, \text{ or}\\ \bullet\ g_i(x) > \underline{\epsilon}_g \text{ for at least one } i \text{ between 1 and } m_2.\end{array}} \tag{3}$$

Generally, in a branch and bound algorithm, boxes satisfying (2) can be constructed. Adjacent boxes can then be eliminated using (3), where $\underline{\epsilon}_\varphi$, $\underline{\epsilon}_c$, and $\underline{\epsilon}_g$ are chosen significantly smaller than ϵ_φ, ϵ_c, and ϵ_g, respectively. Such algorithms can be significantly more practical than algorithms that complete with a list of boxes such that all solutions x to (1) are proven to be in one of the boxes. This is particularly true for singular problems; for examples, see [43] and [42].

2.4. *Impact on Current Software*

The type of solution provided by current software packages is not necessarily well documented. In [43], we present a simple example that contains a parametrized line of global optimizers, but for which different packages give different approximate

optimizers, without any indication of singularities. Comparisons would be easier if design goals (with regard to types of solutions) were clearly and prominently stated.

3. A Toolbox of Techniques

In this section, we examine components of deterministic global optimization algorithms with an eye towards how practical it is to make them mathematically rigorous.

3.1. *Decomposition and Relaxations*

A popular and flexible type of relaxation is a linear relaxation.

3.1.1. Linear Relaxations

Suppose we replace φ by a linear function ℓ_φ such that $\ell_\varphi(x) \leq \varphi(x)$ for $x \in \boldsymbol{x}$, and we replace each constraint $g_i \leq 0$ by one or more linear constraints $\ell_{g_i} \leq 0$ with $\ell_{g_i}(x) \leq g_i(x)$ for $x \in \boldsymbol{x}$ and i between 1 and m_2. If we similarly replace each $c_i = 0$ by $\ell_{c_i} \leq 0$ and $\ell_{-c_i} \leq 0$, we will have replaced the problem (1) by a linear program, or *linear relaxation*. In fact, if φ and the g_i are convex and the c_i are linear[1], problem (1) can be approximated arbitrarily closely in this way. The result, when the original problem (1) represents a convex program, is that the corresponding linear relaxation (2) can be formed and solved in polynomial time, if it were exact, its solution $\underline{\varphi}_\ell$ would be a lower bound on the global optimum φ^* of (1), $\underline{\varphi}_\ell \leq \varphi^*$ and $\underline{\varphi}_\ell$ is as close to φ^* as desired. The process of forming the linear relaxation and finding $\underline{\varphi}_\ell$ can efficiently be made rigorous, so $\underline{\varphi}_\ell < \varphi^*$ is mathematically guaranteed; see §4.1 in this paper. However, relating an optimizing point x^*_ℓ of the linear relaxation to an optimizing point x^* of the original problem is trickier; see §5 of this paper.

Descriptions of how linear relaxations are formed occur in various works. We have written up an analysis of one type of process by which linear relaxations can be formed and have provided additional references in [27].

3.1.2. Other Relaxations

There are various ways of forming linear relaxations. For instance, in the process we analyzed in [27], evaluation of the objective and constraints is decomposed into individual operations (e.g. "+", "−", "×", and "÷"), slack variables and corresponding constraints are added corresponding to each such operation, and linear underestimators are supplied for each such operation. This type of linear relaxation is easy to automate[2], but can lead to very large linear programs, and may not be the most efficient way. An alternative is to identify larger sub-expressions of the objective and constraints that are appropriate to approximate linearly.

Along these lines, the approximation of a subexpression does not even need to be linear, but can be convex. The Floudas et al team has put forward noteworthy developments along these lines, as explained in [14]. The decomposition is based on decomposition of a function f, where f can correspond to an objective φ or a constraint $g_i \leq 0$, $c_i \leq 0$ or $-c_i \leq 0$, as follows.[3]

---

[1] that is, for a *convex program*

[2] e.g. with operator overloading, as opposed to development of a special parsing program

[3] Prof. Floudas has presented this composition explicitly in talks such as [11].

$$
\begin{aligned}
f(x) = \quad & \underset{\substack{\uparrow \\ \text{linear}}}{f_L(x)} + \underset{\substack{\uparrow \\ \text{convex}}}{f_C(x)} + \sum_{k=1}^{n_b} a_k \underset{\substack{\uparrow \\ \text{bilinear}}}{x_{i_{1,k,b}} x_{i_{2,k,b}}} \\
& + \sum_{k=1}^{n_t} \underset{\substack{\uparrow \\ \text{trilinear}}}{b_k x_{i_{1,k,t}} x_{i_{2,k,t}} x_{i_{3,k,t}}} + \sum_{k=1}^{n_f} \underset{\substack{\uparrow \\ \text{fractional}}}{c_k \frac{x_{i_{1,k,f}}}{x_{i_{2,k,f}}}} + \sum_{k=1}^{n_{ft}} \underset{\substack{\uparrow \\ \text{fractional} \\ \text{trilinear}}}{d_k \frac{x_{i_{1,k,ft}}}{x_{i_{2,k,ft}}}} \\
& + \sum_{k=1}^{n_s} \underset{\substack{\uparrow \\ \text{signomial}}}{f_{k,s}} + \sum_{k=1}^{n_{uc}} \underset{\substack{\uparrow \\ \text{univariate} \\ \text{concave}}}{f_{k,uc}} + \underset{\substack{\uparrow \\ \text{general} \\ \text{nonconvex}}}{f_{GNC}(x)},
\end{aligned}
\tag{4}
$$

where n_b, n_t, n_f, n_{ft}, n_s, and n_{uc} are the number of bilinear, trilinear, fractional, fractional trilinear, signomial[1], and univariate concave terms, respectively. The above decomposition may be motivated partly from the fact that terms of the represented forms are found in the various models considered by the Floudas team; see [14]. Relaxations do not need to be given for the linear and convex terms, while various techniques are used to construct relaxations to the bilinear, trilinear, fractional, fractional trilinear, signomial, and univariate concave terms. The bivariate and trivariate terms (bilinear, trilinear, fractional, and fractional trilinear) may be estimated with techniques predating the work of Floudas et al, but are unified in [36, Theorem 1], and explained fully in [14] and later work. Relaxations of general univariate concave functions over an interval may be obtained as linear interpolations of their values at the end points of the interval. Floudas develops the "αBB" method for convex relations of general non-convex terms f_{gnc}, see [14, Chapter 12]. In the αBB method, a parameter α times a quadratic term is added in such a way that the resulting term dominates all of the negative eigenvalues of the Hessian matrix over $\boldsymbol{x}$, while the negative eigenvalues are bounded using an interval evaluation of the Hessian matrix and Gerschgorin's theorem and other techniques; see [14, §12.4].

Recently, Floudas has reported solving a very large previously unsolved practical problem by arbitrary close approximation with these decomposition and relaxation techniques alone, without the need for branch and bound [12].

3.2. *Constraint Propagation*

Constraint propagation is based on a simple idea: Solve a relation in many variables for one of the variables, then plug in bounds on the other constraints to compute new bounds on the original constraint. We took this point of view, giving examples, in [22]. Actually, this simple idea is the underlying computation in the entire field of constraint propagation, also known as *constraint solving*. The computation underlies a programming paradigm, various books are devoted to constraint solving, as well as uniquely suited programming languages and periodic conferences, and there are numerous industrial applications; see [6], or perhaps [4, 41, 48]. An early deterministic global optimization package constructed upon the philosophy of

[1]Signomial terms are terms of the form $\prod_{i=1}^{N} t_i^{\alpha_i}$, where the α_i are arbitrary real numbers.

constraint logic programming is [51]. The technique is presently incorporated into leading commercial deterministic global optimization software, such as BARON [45].

For future discussion, we give a simple example. Consider

$$\begin{aligned} &\text{minimize } \varphi(x) = x_1^2 - x_2^2 \\ &\text{subject to } x_1^2 + x_2^2 = 1, \\ &\qquad\qquad x_1 + x_2 \le 0. \end{aligned}$$

Suppose we have already found the feasible point $\hat{x} = (0, -1)$ with $\varphi(\hat{x}) = -1$, so -1 is an upper bound on the optimum, and suppose we are searching in the box $([-1, 1], [-1, 1])$. Using the upper bound $\overline{\varphi}$ gives

$$x_1^2 - x_2^2 \le -1.$$

Solving this for x_1 gives

$$x_1 \le \sqrt{[-1,1]^2 - 1} = \sqrt{[0,1] - 1} = \sqrt{[-1,0]}$$

and

$$x_1 \ge \sqrt{[-1,1]^2 - 1} = \sqrt{[0,1] - 1} = \sqrt{[-1,0]}.$$

Here, it is appropriate to interpret $\sqrt{[-1,0]} = 0$, so we obtain $x_1 = 0$. We now solve for x_2 in $x_1^2 + x_2^2 = 1$ and plug in $x_1 = 0$ to get

$$x_2 = 1 \quad \text{or} \quad x_2 = -1,$$

Plugging $x_1 = 0$, $x_2 = 1$ into $x_1 + x_2 \le 0$ gives a contradiction, leading to the unique point $x = (0, -1)$ in $([-1, 1], [-1, 1])$ that can be a global optimizer.

It is clear from this example that propagation of continuous constraints depends on bounding the range of expressions over interval vectors, whether or not the software is designed to be mathematically rigorous.

3.3. *Interval Newton Methods*

Interval Newton methods and Krawczyk-like methods, usually found just in software designed to be mathematically rigorous, are a prominent feature throughout the literature on interval computations. For example, introductions can be found in our review and didactic works, such as [1, §8.4] or [37, Chapter 8], and there are many other excellent references. The general framework of interval Newton methods is as follows. Suppose $F : \boldsymbol{x} \subset \mathbb{R}^n \to \mathbb{R}^n$ where $\boldsymbol{x}$ is an interval n-vector, suppose $\dot{x} \in \boldsymbol{x}$ is fixed, and suppose that $\boldsymbol{A}$ is an interval matrix such that

$$\boxed{\begin{array}{l}\boldsymbol{A} \text{ contains all matrices } A \text{ such that} \\ F(x) - F(\dot{x}) = A(x - \dot{x}) \text{ for every } x \in \boldsymbol{x}.\end{array}} \tag{5}$$

(For example, a componentwise interval extension of the Jacobian matrix of F over $\boldsymbol{x}$ will do.) Then a *multivariate interval Newton operator* F is any mapping $\boldsymbol{N}(F, \boldsymbol{x}, \dot{x})$ from the set of ordered pairs $(\boldsymbol{x}, \dot{x})$ of interval n-vectors $\boldsymbol{x}$ and point

n-vectors $\dot{x}$ to the set of interval n-vectors, such that

$$\tilde{\boldsymbol{x}} \leftarrow \boldsymbol{N}(F, \boldsymbol{x}, \dot{x}) = \dot{x} + \boldsymbol{v}, \tag{6}$$

where $\boldsymbol{v} \in \mathbb{IR}^n$ (where $\mathbb{IR}^n$ is the set of all n-dimensional interval vectors) is any box that bounds the solution set to the linear interval system

$$\boldsymbol{A}v = -F(\dot{x}). \tag{7}$$

There are well-established ways of bounding solutions to (7), with a rich literature on interval linear algebra.

Attractive properties of interval Newton methods are as follows.

(1) Any solution $x^* \in \boldsymbol{x}$, $F(x^*) = 0$ is also in $\boldsymbol{N}(F, \boldsymbol{x}, \dot{x})$.
(2) If $\boldsymbol{N}(F, \boldsymbol{x}, \dot{x}) \cap \boldsymbol{x} = \emptyset$, then there are no solutions to $F(x) = 0$ within $\boldsymbol{x}$.
(3) If the interval matrix $\boldsymbol{A}$ obeys (5) for *every* $\dot{x} \in \boldsymbol{x}$ (such as if $\boldsymbol{A}$ is a componentwise interval extension of the Jacobian matrix of F over $\boldsymbol{x}$), and if $\boldsymbol{N}(F, \boldsymbol{x}, \dot{x})$ is contained in the interior of $\boldsymbol{x}$, then there is a unique solution of $F(x) = 0$ within $\boldsymbol{x}$.

(One reference for the theory underlying interval Newton methods is [38].)

Interval Newton methods are local methods, good for finding tight rigorous bounds on solutions to linear and nonlinear systems of equations, given a good approximation to such solutions. They are of some, but more limited use, over larger regions in reducing their size and in proving existence and non-existence. In particular, the condition in useful property (3) of interval Newton methods is most likely to hold if the point $\dot{x}$ is sufficiently close to an actual solution x^* of $F(x^*) = 0$, if $\dot{x}$ is near the center of $\boldsymbol{x}$, if the widths of $\boldsymbol{x}$ are large compared to the distance of $\|\dot{x} - x^*\|$, and if the widths of $\boldsymbol{x}$ are sufficiently small. Specifically, the maximum size of $\boldsymbol{x}$ for which the condition in property (3) is satisfied can be thought to be roughly inversely proportional to the condition number of the Jacobian matrix of F near $\dot{x}$; the analysis in [24, §6.2.2] is revealing.

In the context of global optimization, F represents the Kuhn–Tucker conditions or the Fritz John equations. However, the Kuhn–Tucker system can be ill-conditioned or singular, and the condition in property (3) is likely to hold only if good bounds on the Lagrange multipliers are known.

3.4. *Overall Branch and Bound Methods*

Algorithm 1 (The general branch and bound procedure)

In general, relaxations, constraint propagation, interval Newton methods, and other such techniques alone cannot yield acceptable bounds on the global optimizers. In such instances, the region $\boldsymbol{x}$ is typically subdivided, to achieve better accuracy over the smaller subregions. Here, for further discussion, we present a prototypical such subdivision scheme.

INPUT*: The objective φ, the constraints c and g for problem 1, and the initial search region $\boldsymbol{x}$.*

OUTPUT*: An enclosure $[\underline{\varphi}^*, \overline{\varphi}]$ for the global optimimum and (depending on requirements), a list $\mathcal{C}$ of boxes containing some (or all) global optimizers.*

(1) Establish an upper bound $\overline{\varphi}$ on the global optimum over the feasible set (defined by the constraints).
(2) (Branching) Subdivide the initial region $\boldsymbol{x}$ into two or more subregions $\tilde{\boldsymbol{x}}$. Place all but one of these on a list $\mathcal{L}$ for further processing.

(3) Use various methods to reduce the size of $\tilde{\boldsymbol{x}}$ (possibly producing the empty set).
(4) (Bounding) Bound below the range of the objective function over each subregion $\tilde{\boldsymbol{x}}$, to obtain

$$\underline{\varphi}(\tilde{\boldsymbol{x}}) \leq \{\varphi(x) \mid x \in \tilde{\boldsymbol{x}}, c(x) = 0, g(x) \leq 0.\}\,.$$

(5) If $\underline{\varphi} > \overline{\varphi}$, (1) is infeasible over $\tilde{\boldsymbol{x}}$, if the Kuhn–Tucker[1] conditions do not hold at any point in $\tilde{\boldsymbol{x}}$, or if $\tilde{\boldsymbol{x}}$ cannot contain a global optimizer for some other reason
Then
(Pruning) Discard $\tilde{\boldsymbol{x}}$,
Else If the diameter of $\tilde{\boldsymbol{x}}$ is smaller than a specified tolerance

Then
Put $\tilde{\boldsymbol{x}}$ onto a list of boxes containing possible global optimizers.
Else
Insert $\tilde{\boldsymbol{x}}$ into the list $\mathcal{L}$ for further branching and bounding through steps (2) and (4).
End If
End If

End Algorithm 1.

We emphasize that the above algorithmic description is mainly for our present expository purposes, and lacks subtleties and details important for efficiency in particular contexts.

Branch and bound algorithms based loosely on Algorithm 1 are ubiquitous in the literature on deterministic global optimization. Some (but not the more recent) such algorithms are compared in [40]. Relaxations, constraint propagation, and other techniques are generally combined in Steps (3), (4), and (5), and the details of these techniques, the way they are combined, and efficiency of the computer programming greatly impact the practicality of the overall algorithm. Furthermore, more basic properties of this scheme make a difference. One such property is ordering of the list $\mathcal{L}$ of subregions waiting to be processed. This ordering is especially important if the optimum, as opposed to all optimizers, is desired, and different orderings may be more appropriate in different multiprocessing environments. Similar considerations apply in the way a region $\tilde{\boldsymbol{x}}$ is further subdivided in step (2). Furthermore, regions $\tilde{\boldsymbol{x}}$ that are discarded in step (5) may sometimes be expanded, as explained, e.g. in [24, §4.2]; the way in which this is done and in which the volume of such expanded regions is deleted from further consideration can greatly effect the overall practicality of the algorithm.

In this work, we focus not on the overall branch and bound algorithm and the way the individual techniques are combined, but on the practicality of making the individual techniques rigorous. In this way, we strive to gain insight into how practical it is to make non-rigorous algorithms rigorous, into intrinsic limits of rigorous computations, and into where additional work is worthwhile.

[1] or, alternatively, Fritz John

4. Rigor in Individual Techniques

Here, we examine individually the practicality of implementing the aforementioned techniques in a mathematically rigorous way.

4.1. *Making Linear Relaxations Rigorous*

Some of the computations with linear relaxations can be made mathematically rigorous with negligible performance penalty. However, care needs to be taken in doing so. There are two steps in which roundoff error needs to be considered in implementing linear relaxations in a rigorous way:

(1) computing the coefficients of the linear functions comprising the relaxations, and
(2) providing rigorous bounds on the solution to the linear relaxation.

For step (1), the underestimating functions $\ell(x)$ (e.g. $\ell_\varphi(x)$, $\ell_{g_i}(x)$, $\ell_{c_i}(x)$, $\ell_{-c_i}(x)$, or linear underestimations to individual operations comprising the computation of the objective or constraints) are linear relations

$$\ell : \mathbb{R} \to \mathbb{R}, \quad \ell(u) = au + b,$$

such that $\ell(u)$ forms either a secant line to the function $f(u)$ being underestimated over some interval $\boldsymbol{u} = [\underline{u}, \overline{u}]$, or a tangent line to f at some $u \in \boldsymbol{u}$. The trick is to use directed rounding appropriately in the computation of the coefficients a and b to be able to assert that the stored machine representations of a and b have $\ell(u) \leq f(u)$ for every $u \in \boldsymbol{u}$. This is done rigorously in [5]; see also [18].

The linear relaxation formed in step (1) is typically a large, sparse linear program that is solved approximately by a conventional proprietary or open-source linear programming solver, using well-studied mature technology. The optimal objective value, if exact, would represent a lower bound on the solution to the original problem (1). A successful approach to obtaining a mathematically rigorous lower bound to (1), given an approximate solution returned by conventional software, involves several simple computations (dot products and matrix-vector multiplications), with careful setting of the rounding mode, and is based on the duality gap. An approximate solution to the dual problem is also needed, but these are usually available in current linear programming solvers. Additionally, the technique, using information returned from the approximate linear programming solver, can also, in relevant instances, prove that the relaxation, and hence the original problem (1) is infeasible over $\boldsymbol{x}$. See [39] for a clear explanation of the technique; also see [20], where the technique was independently reported.

Thus, much of the linear relaxation technology used in non-rigorous deterministic branch and bound software can be made rigorous with negligible performance penalty.

The range $\boldsymbol{x}_i = [\underline{x}_i, \overline{x}_i]$ can also be reduced through the linear relaxation, using the linear (relaxed) constraints, using x_i or $-x_i$ as objective, and appending the additional constraint (or relaxation thereof) $\varphi(x) \leq \overline{\varphi}$, where $\overline{\varphi}$ is the current upper bound on the global optimum. This technique can similarly be easily made mathematically rigorous.

Nonetheless, it is less straightforward to relate approximate optimizers to the linear relaxation to rigorous enclosures of optimizers of the original problem (1). Some previous work on bounding the solutions to linear programming problems with interval coefficients, occurs in [3],[33], or [44]. We say more about relating

optimizers of the relaxation to optimizers of the original problem and propose a possible computational framework in sections 4.4.2 and 5.

4.2. *Making Other Relaxations Rigorous*

In addition to linear relaxations, convex, and possibly other, relaxations are commonly used. We consider mathematically rigorous implementations involving such relaxations here.

4.3. *Ensuring the Computed Relaxation is Actually a Relaxation*

Maranas and Floudas, in [36] and then in [14, §12.2] give explicit formulas for convex underestimators for each of the terms in (4). For each of these terms except the general nonconvex term, the techniques used to form the underestimators are similar to the linear case[1]. Thus, there should be no problem implementing computation of these underestimators in such a way that the resulting machine representation is guaranteed with mathematical rigor to be a convex relaxation. (However, to our knowledge, this has not yet been actually documented or tried.) Floudas et al use the "αBB" approach (as in [14, Chapter 10]) to construct a convex underestimator for the general non-convex term. In fact, Floudas himself presents a method for mathematically rigorously constructing an underestimator to this term; see [14, §12.4].

4.4. *Computing Rigorous Bounds, Given Solutions to Relaxations*

Obtaining rigorous bounds on the optimum and rigorous bounds on members of the optimizing set (or enclosing the optimizing set) involve both different techniques and different degrees of difficulty. The techniques and degree of difficulty also depend upon whether or not we require bounds on the exact solution to the problem or whether or not we are enclosing epsilon-optimal solutions or proving that a given point is epsilon-optimal.

4.4.1. Obtaining a Rigorous Lower Bound on the Optimum of the Relaxation

For general relaxations, a lower bound $\underline{\varphi}^*$ on the global optimum φ^* to the relaxation (and hence, to the original problem) can be computed by computing a solution to the relaxation, while an upper bound can be computed by computing the objective value at a feasible point. Approximate optimizers for both linear and convex programs can be computed efficiently (and, in many but not all cases, reliably) with a choice of software packages.

If the relaxation is linear, an approximate optimum of the relaxation can be perturbed to a mathematically rigorous lower bound $\underline{\underline{\varphi}}_\ell$ with the aforementioned duality gap technique of Neumaier and Shcherbina [39]. Also see [10]. To our knowledge, an analogous technique has not yet been developed for general convex programs, but we are optimistic one can be found. One possibility might be to apply an interval Newton method to the Kuhn–Tucker equations for the relaxation to prove existence of a critical point[2], taking heed of the possible pitfalls we have mentioned in §3.3.

---

[1] although the formulas are not quite as simple.
[2] which should be unique for a convex program

4.4.2. Ensuring the Relaxed Optimizer is Near an Optimizer of the Original Problem

Given an optimizer of a relaxation, obtaining mathematically rigorous bounds on a corresponding optimizer of the original problem does not seem to be as simple as obtaining rigorous bounds on the optimum. One possibility would be to use work such as that of Jansson [19, 21], while expressing the original problem or the relaxation as a linear problem with uncertain coefficients. However, we are unaware of widespread success with the techniques from [19] and [21] for bounding the solution sets of linear programs with interval coefficients, let alone using such solution set bounds to bound solution sets of general nonlinear optimization problems.

If the original problem (1) is convex, a possibility worth investigating, as mentioned above for computing bounds on the optimum, is to use an interval Newton method to prove existence of a solution to the Kuhn-Tucker equations. Uniqueness theory associated with convex problems can then be used to prove globality.

In Section 5, we give an analysis of convergence of optimizers of relaxations, along with an example illustrating how close the optimizing set is in a particular case. Additional development of these ideas may lead to more general techniques.

4.4.3. On Epsilon-Optimality

In contrast, proving that a point is epsilon optimal in the sense of problem (2) is very simple: Perform interval evaluations of φ, the c_i, and the g_i. Thus, if a single epsilon-optimal point is desired, it is almost trivial to prove that a point found either by an approximate local optimizer or by such an optimizer applied to a relaxation is epsilon-optimal. Finding enclosing regions to *all* such epsilon-optimal solutions is much more computationally intensive for general nonlinear problems, and in general requires an exhaustive search of the region. Furthermore, the goals of such a search must be carefully defined; see [42].

4.5. *Making Constraint Propagation Rigorous*

In both mathematically rigorous and non-rigorous software, evaluation of the ranges of expressions is typically done with interval computations. This can be made mathematically rigorous with virtually no performance penalty. In fact, constraint propagation based on interval computations is an important component of the award-winning commercial BARON global optimization software [45, 49], even though BARON as a whole is not claimed to give mathematically rigorous results.

4.6. *Proving Feasibility*

A significant component of branch and bound algorithms for continuous global optimization is maintaining an upper bound on the global optimum, to use in determining branch cuts and in eliminating regions. In non-rigorous algorithms, the objective φ may simply be evaluated at an approximate feasible point $\hat{x}$, to obtain an approximate upper bound, where the feasible point is obtained from a local optimization process or by some projection method onto the feasible point. To make such a process rigorous, one must not only bound roundoff error in evaluation of φ, but must also assure that $\hat{x}$ is feasible. If problem (1) has only inequality constraints, and none of them are active (that is, if there are no c_i and if $g_i(\hat{x}) < 0$ for each i, then feasibility can be rigorously ascertained by evaluating the g_i at $\hat{x}$ while taking account of roundoff error[1]. However, if there are equality constraints

[1]such as evaluating using interval arithmetic

or if some of the inequality constraints are approximately active (i.e. if $g_i(\hat{x}) \approx 0$ for one or more i), simple interval evaluation will not prove that $\hat{x}$ is feasible. In fact, the only rigorous method of which we know is to construct a box $\hat{\boldsymbol{x}}$ about $\hat{x}$ within a subspace of dimension equal to the number of active constraints, then use an interval Newton method (or analogous fixed-point-contraction criterion) to prove existence of a feasible point within $\hat{\boldsymbol{x}}$, then evaluate φ over $\hat{\boldsymbol{x}}$ to obtain the upper bound; see [25]. Implementation of such procedures adds complication, and the procedure isn't successful at all approximate feasible points $\hat{x}$. For instance, see our comments in §3.3.

Such problems are less severe if we merely desire to prove we have found epsilon-approximate solutions (and not necessarily to enclose all such epsilon-approximate solutions). In such cases, an interval Newton method is not required, and implementing the algorithm with mathematical rigor should not carry a significant performance penalty.

4.7. *Rigor's Cost: Rigor or General Algorithm Design?*

Current global optimization algorithms use sophisticated combinations of these techniques, and have differing goals (as we outlined in §2). Although software that does not claim to be mathematically rigorous has performed better in certain benchmarks than mathematically rigorous software, this is not convincing proof that mathematical rigorous software cannot be competitive. Many facets have not been taken into account. Except for rigorous verification of feasibility, most aspects of general global optimization algorithms can be made rigorous without significant performance penalty, and there are alternatives to verification of feasibility for obtaining rigorous upper bounds on the optimum. One such alternative may be computing a linear relaxation and bounding the optimizing points of the relaxation. Theorem 5.2 in the next section hints at the possibility of this. Another possibility, if the application allows it, is to replace the original problem by a relaxed decision problem, as in (2). Further design and experimentation is needed.

5. On Convergence of Optimizers of Relaxations

A common technique in continuous global optimization algorithms is approximation by relaxations. In fact, in the work by Floudas et al [12] we mentioned in §3.1.2, large-scale practical non-convex optimization problems have been solved approximately by separately constructing relaxations from subdividing individual coordinates, without branching and bounding in the overall parameter space. Similarly, leading current general software such as BARON [45] also relies on relaxations. Even though such problems are non-convex, careful a priori analysis of the structure of the problem reveals how one might replace the problem by an equivalent problem (in the sense defined [27]), in which the only non-convex constraints are dependent on one or two variables, which are possibly artificial slack variables. In this way, the problem can be approximated arbitrarily closely by a set of convex relaxations, by subdividing the intervals corresponding to variables upon which the non-convex constraints depend. By solving the relaxations, the approximate problems then lead to approximate solutions. We illustrate this subdivision process in a very simple context in Example 5.3, although the technique has been applied to solve important large problems from application areas.

When such a process is used in a non-rigorous way, an approximate optimum (or perhaps, non-rigorous lower and upper bounds on an approximate optimum), as

well as approximate optimizing point or points are obtained from the relaxations. When lower and upper bounds on the optimum are obtained, such bounds can be processed, at least when the relaxation is linear, to obtain mathematically rigorous lower and upper bounds. However, to our knowledge, no attempt has been made to date to formally relate the optimizing set of the relaxations to the optimizing set of the original problem. Theorem 5.2 in this section gives such a relation, highlighting the limitations of relaxations to determine the entire optimizing set in mathematically rigorous computations. Furthermore, the theorem gives weak conditions (namely, that the objective and constraints are approximated well) under which the optimum of the relaxations converge to the optimum of the original problem. Ensuring that such conditions are satisfied for non-convex problems is, in general, problematical, but can be done if either non-convex relaxations whose global optima can be easily obtained can be constructed, or if the problem can be divided into a relatively small but increasing number of subproblems whose relaxations approximate the original subproblem well. This is illustrated in a simple setting in Example 5.3, while other techniques are given in [49] and [50].

Theorem 5.2 is based on a simple continuity argument. We present the theorem, following a definition that clarifies our notation.

DEFINITION 5.1 *Let $\mathcal{T}$ be a set, and let s be a point. The distance $d(s,\mathcal{T})$ is defined as*

$$d(s,\mathcal{T}) = \min_{t\in\mathcal{T}} \|s-t\|.$$

If $\mathcal{S}$ and $\mathcal{T}$ are two sets, then the distance $d(\mathcal{S},\mathcal{T})$ between them is defined as

$$d(\mathcal{S},\mathcal{T}) = \max\left\{\max_{s\in\mathcal{S}} d(s,\mathcal{T}), \max_{t\in\mathcal{T}} d(t,\mathcal{S})\right\}$$

THEOREM 5.2 *Assume the following,*

(1) In problem (1) *assume φ, the g_i, and the c_i are defined over a domain $\boldsymbol{D}$, and that φ is uniformly continuous over $\boldsymbol{D}$. Denote the feasible set of problem* (1) *by $\mathcal{F} \subseteq \boldsymbol{D}$, denote the optimum by φ^*, and denote the optimizing set by $\mathcal{O}$.*

(2) Let $\boldsymbol{P}_\epsilon$ denote any member of a family of relaxations of problem (1)*, with feasible set $\mathcal{F}_\epsilon \subseteq \boldsymbol{D}$, with objective function φ_ϵ, with global optimum φ^*_ϵ, and with set of optimizers $\mathcal{O}_\epsilon$. Assume that each such $\boldsymbol{P}_\epsilon$ has the following properties:*

(i) Each feasible set $\mathcal{F}_\epsilon$ is bounded.

(ii) The family $\{\varphi_\epsilon\}_{\epsilon>0}$ is equicontinuous over $\boldsymbol{D}$.

(iii) $0 \le \varphi(x) - \varphi_\epsilon(x) < \epsilon$ uniformly over $\boldsymbol{D}$.

(iv) $d(\mathcal{F}_\epsilon, \mathcal{F}) < \epsilon$.

Then,

(a) $\lim_{\epsilon\to 0} \varphi^*_\epsilon = \varphi^*$,

(b) For each tolerance $\tau > 0$, there is an $\epsilon_d > 0$ such that $\epsilon < \epsilon_d$ implies that, for any global optimizing point x^*_ϵ of $\boldsymbol{P}_\epsilon$, there is an optimizing point x^* of problem (1) *such that $\|x^*_\epsilon - x^*\| < \tau$.*

Theorem 5.2, although abstract, provides guidance for construction of approximations. The condition $d(\mathcal{F}_\epsilon,\mathcal{F}) < \epsilon$ can be checked in terms of bounds on the approximations on the g_i and c_i, while the condition $|\varphi(x) - \varphi_\epsilon(x)| < \epsilon$ can be similarly checked directly. (Explicit such bounds are given in review in [14].)

Proof To prove (a), first observe that $\mathbf{P}_\epsilon$ is a relaxation of the original problem implies $\varphi_\epsilon^* \le \varphi^*$, and conditions (i) and (iii) imply that each φ_ϵ^* is bounded below by the same number μ, that is, $\varphi_\epsilon^* \in [\mu, \varphi^*]$ for all ϵ, so the φ_ϵ^* have a limit point $\tilde{\varphi} \le \varphi^*$ as $\epsilon \to 0$. Assume $\tilde{\varphi} < \varphi^*$, and set $\Delta = \varphi^* - \tilde{\varphi}$. Then, choose ϵ_0 such that, for $\|y - x\| < \epsilon_0$,

(i) $\|y - x\| < \epsilon_0 \implies \|\varphi_\epsilon(y) - \varphi_\epsilon(x)\| < \Delta/6$, and
(ii) $|\varphi(y) - \varphi(x)| < \Delta/6$.

for every $x, y \in \mathbf{D}$. Now, for any $\epsilon \le \epsilon_0$, any optimizing point $x_\epsilon^* \in \mathcal{F}_\epsilon$ of the relaxation is within ϵ of a point $\hat{x}_\epsilon \in \mathcal{F}$ of problem 1. Then, for $\epsilon = \epsilon_i$, a member of the sequence for which $\varphi_\epsilon^* \to \tilde{\varphi}$,

$$\begin{aligned} |\varphi_\epsilon(x_\epsilon^*) - \varphi(\hat{x}_\epsilon)| &\le |\varphi_\epsilon(x_\epsilon^*) - \varphi_\epsilon(\hat{x}_\epsilon)| + |\varphi_\epsilon(\hat{x}_\epsilon) - \varphi(\hat{x}_\epsilon)| + |\varphi(\hat{x}_\epsilon) - \varphi(x^*)| \\ &< \Delta/6 + \Delta/6 + \Delta/6 = \Delta/2. \end{aligned}$$

Thus,

$$\begin{aligned} 0 \le \varphi(\hat{x}_\epsilon) - \varphi_\epsilon(x_\epsilon^*) < \Delta/2, &\quad \text{whence} \\ \varphi^* - \varphi_\epsilon(x_\epsilon^*) < \Delta/2, &\quad \text{whence} \\ \lim_{\epsilon \to 0} (\varphi^* - \varphi_\epsilon(x_\epsilon^*)) = \Delta < \Delta/2, & \end{aligned}$$

a contradiction. Hence, part (a) is proved.

To prove part (b), assume to the contrary that there is some $s > 0$ such that, for every ϵ, there is an optimizer x_ϵ^* of some problem $\mathbf{P}_\epsilon$ with $d(x_\epsilon^*, \mathcal{O}) > s$. Choose a sequence of points $x_{\epsilon_n}^*$ with $\epsilon_n \downarrow 0$, such that $d(x_{\epsilon_n}^*, \mathcal{O}) > s$ for every member of the sequence. Furthermore, since $\mathcal{F}_\epsilon$ is bounded for each ϵ and since $\mathcal{F}_{\epsilon_n} \subseteq \mathcal{F}_\epsilon$ whenever $\epsilon_n \le \epsilon$, the points $x_{\epsilon_n}^*$ are bounded, so there is a convergent subsequence which, without loss of generality, we will also denote by $x_{\epsilon_n}^*$. Furthermore, $x_{\epsilon_n}^* \to x_*^*$, where $x_*^* \in \mathcal{F}$. However, (a), the fact that $\varphi_\epsilon \to \varphi$ as $\epsilon \to 0$, and continuity of φ and φ_ϵ give $\varphi(x_*^*) = \varphi^*$, so $x_*^* \in \mathcal{O}$. The fact that the sequence $x_{\epsilon_n}^*$ converges to x_*^* thus leads to a contradiction, proving part (b). ■

Although, under the hypotheses of Theorem 5.2, $\mathcal{O}_\epsilon$ tends to a subset of $\mathcal{O}$, it is not true in general that every point in $\mathcal{O}$ is eventually approximated well by a point in $\mathcal{O}_\epsilon$, that is, it is not true in general that $d(\mathcal{O}_\epsilon, \mathcal{O}) \to 0$ as $\epsilon \to 0$. A

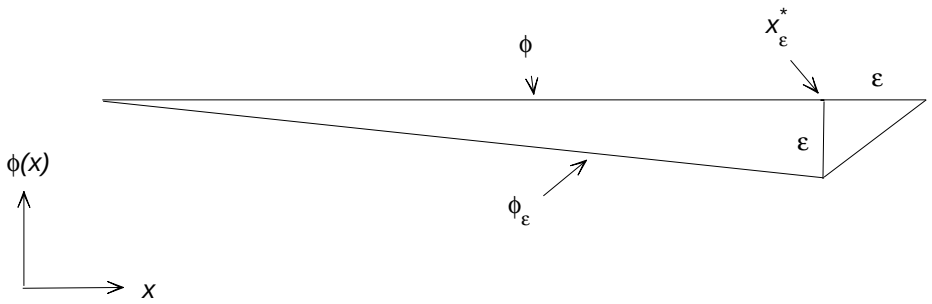

Figure 1. Example where not all optimizers of the original problem are limit points of relaxation optimizers.

counterexample is illustrated in Figure 1, where φ is constant (and hence every point is optimal), while φ_ϵ is constructed so x_ϵ^* converges to a single point (in Figure 1, the right end point) as $\epsilon \to 0$.

Example 5.3 Consider an optimization problem of the form (1), with $n = 2$,

$$\varphi(x) = x_1^2 - x_2^2,$$

and $\mathbf{D} = ([-1, 1], [-1, 1])$. Assuming the constraints are convex, construct smooth convex relaxations $\varphi_\epsilon(x)$, with ϵ explicitly known.

From the assumptions, we need only replace φ by a relaxation. Here, φ has a convex term and a concave term, so convex relaxations need only be given for the univariate concave term $-x_2^2$. Divide $[-1, 1]$ into $2N$ subintervals $S_i = [-1 + (i - 1)/N, -1+i/N]$, $1 \le i \le 2N$. Replace $-x_2^2$ by $I_N(x_2)$, where $I_N(x_2)$ is the piecewise linear interpolant to $-x_2^2$ with sub-intervals S_i, and define

$$\tilde{\varphi}(x) = x_1^2 + I_N(x_2).$$

Then $\tilde{\varphi}(x) \le \varphi(x)$ for $x \in \mathbf{D}$ (since $f(t) = -t^2$ is concave), but $I_N(t)$ is not convex. If a non-convex relaxation (or indeed, the problem without replacing $-x_2^2$ at all) can be solved efficiently, this can be done. However, if the computations are set up to solve linear relaxations, we may consider the $2N$ subproblems obtained by restricting the domain of x_2 to S_i, $1 \le i \le 2N$. It is well-known that the piecewise linear interpolation $I(f)(t)$ of a function f with maximum subinterval length δ has an error bound of

$$|f(t) - I(f)(t)| \le \frac{1}{8}\delta^2 \max |f''(t)|,$$

so

$$\left|-t^2 - I_N(t)\right| \le \frac{1}{4N^2}.$$

Thus, for the hypotheses of Theorem 5.2 to apply to each subproblem, take $N \ge 1/(2\sqrt{\epsilon})$. Note that, in this case, the number of relaxations we need to solve goes up as the square of the accuracy of approximating problem, while the accuracy of the approximate optimum and approximation to a subset of the solution set is related to the closeness of the approximating problem to the original problem through the moduli of continuity of the φ_ϵ, φ, the constraints, and the approximating constraints. However, if there are only several such non-convex terms in a large problem, the total number of relaxations to be solved to achieve a particular accuracy could be much less than would be required with a general branch and bound process over the whole space. Also, if it can be shown that the hypotheses of the theorem can be met through this process, the computations themselves can bound the error in the objective.

(For this example, the uniform continuity of each $\varphi_\epsilon^{(i)}$ corresponding to the i-th subinterval follows from the fact that piecewise linear interpolants are Lipschitz with the same Lipschitz constant as the original function.)

We now illustrate the analysis for relaxed feasible sets $\mathcal{F} \subseteq \mathcal{F}_\epsilon$ for Theorem 5.2.

Example 5.4 Suppose we have a constraint $g(x) = x_1^2 - x_2^2 - 1$ for problem (1), where $\mathbf{D} = ([-1, 1], [-1, 1])$. Assuming the objective and other constraints are convex, construct relaxations $\mathcal{F}_\epsilon$ to apply Theorem 5.2.

We may use techniques that are similar to those we used for Example 5.3. In particular, for this example, we may introduce the same piecewise linear interpolant $I_N(x_2)$ for the term $-x_2^2$, and consider subproblems over each subregion. Replace

the constraint $g(x) \leq 0$ by

$$x_1^2 + \ell_i(x_2) - 1 \leq 0 \quad x_2 \in S_i. \tag{8}$$

We now determine the boundary of the feasible region for the i-th relaxed subproblem defined by the relaxed constraint (8). For a given x_2 in (8), the largest x_1^2 can be is when $x_1^2 = 1 - \ell_i(x_2)$, i.e.

$$|x_1| = \sqrt{1 - \ell_i(x_2)}.$$

In contrast, the analogous bound for the original constraint $x_1^2 - x_2^2 - 1 \leq 0$ is

$$|x_1| = \sqrt{1 + x_2^2}.$$

Thus, an upper bound on the distance between any point x in $\mathcal{F}_\epsilon$ and $\mathcal{F}$ is

$$\begin{aligned} d(x, \mathcal{F}) &\leq \max_{x_2 \in S_i} \left(\sqrt{1 - I_N(x_2)} - \sqrt{1 + x_2^2} \right) \\ &= \max_{x_2 \in S_i} \frac{(-x_2^2) - I_N(x_2)}{\sqrt{1 - I_N(x_2)} + \sqrt{1 + x_2^2}} \\ &\leq \frac{\max_{x_2 \in S_i} \left\{ (-x_2^2) - I_N(x_2) \right\}}{2} \leq \frac{\frac{1}{4N^2}}{2} = \frac{1}{8N^2}, \end{aligned}$$

where we use the well-known fact that a bound on the error in replacing a twice continuously differentiable function f by its linear interpolant with subintervals of length h over the interval $[a, b]$ is $\frac{1}{8}h^2\|f''\|_\infty$. Thus, since $\mathcal{F} \subseteq \mathcal{F}_\epsilon$, $d(\mathcal{F}_\epsilon, \mathcal{F}) < \epsilon$ whenever $N \geq 1/(2\sqrt{2\epsilon})$.

(Of course, if more than one constraint or objective is non-convex, we can apply the above techniques to each of them to obtain our problem $\mathbf{P}_\epsilon$ for Theorem 5.2.)

6. Experiments, Conclusions and Future Work

Studying both algorithms and observing computational performance, we have not discovered significant intrinsic limitations to developing mathematically rigorous versions of successful exhaustive search algorithms for continuous global optimization.

While the author was developing GlobSol [26], a mathematically rigorous software package for finding the entire solution set, he made several observations concerning efficiency. The author did some informal tests in response to reports by Stadtherr et al, in applications such as described in [47], that they were able to compute mathematically rigorous enclosures to entire solution sets by modifying the simpler, earlier software INTBIS [28] with INTLIB [29]; they claimed the computation completed ten times more quickly than with GlobSol. Since INTBIS uses simpler algorithms and fewer heuristics than GlobSol, GlobSol generally completes its branch and bound process with fewer branchings than INTBIS, for a particular problem; thus, the difference in computation time would be due to execution speed in the implementation.

With this motivation, the author did some unpublished profiling studies. One conjecture is that the interval arithmetic implementation (which in GlobSol is based on [23]), or the way the objective and constraint evaluation is set up is a

source of inefficiency. Namely, in GlobSol, all operations required to evaluate the objective and all constraints are recorded before execution of the branch and bound, and, unless sets of values are required simultaneously, the entire set of operations is evaluated interpretively each time any individual objective or constraint value is required. Furthermore, the underlying arithmetic, INTLIB, has elementary functions that are far from optimal, with a factor of 100 or so difference in execution time between floating point and interval values of the trigonometric functions, for instance. However, on some problems, less than half of the total processor time was spent in evaluating functions, with more time spent in dynamic memory allocation and list processing. Also, when we replaced interpretive function evaluation by pre-compiled function evaluation, or when we replaced the arithmetic from [23] by Sun Fortran interval arithmetic (with a very high quality elementary function library), this did not result in significant speedup of the overall algorithm in our informal tests.

Comparisons with other automatically verified software have also revealed large differences in completion time, sometimes due to differences in numbers of boxes processed, and other times not.

Regarding problem structure, in the problem from [43], GlobSol took many hours to complete, since GlobSol needed to enclose an entire line of optimal points, without the benefit of acceleration procedures applicable when the optimal points are isolated and the Kuhn–Tucker system is non-singular at them. In [42], Roy developed a method of computational analysis and an algorithm that computed parametrizations of the solution set and enclosed the solution set with boxes parallel to the parameter directions. This mathematically rigorous algorithm executed in much less time and with several orders of magnitude fewer boxes than GlobSol, despite the fact that it was implemented entirely in MATLAB. The inefficiency observed in GlobSol did not surface in the non-rigorous branch and bound algorithms we tried, since such algorithms did not attempt to locate all globally optimal points, but stopped when one was located.

Another observation we made during development of GlobSol was the importance of an upper bound on the global optimum to be able to quickly reject portions of the parameter space that cannot contain global optimizing points. In non-rigorous software, an approximate feasible point can be computed, and the objective can be evaluated at that approximate feasible point to obtain an approximate upper bound. In contrast, with mathematically rigorous computations, a small region must be constructed about such an approximate feasible point in which it can be proven that a feasible point exists, then an upper bound on the objective over this small region must be computed. We observed failure of this verification step led to no upper bound or a bad upper bound, which in turn led to excessive branching and bounding in regions that could not contain optimizing points.

6.1. *Conclusions*

Our informal explorations and study of various algorithms have led us to believe there isn't a single reason for particular software, automatically verified or not, to perform better on a set of problems than other software. Performance differences are due to many factors, and some of these factors are more important for particular problems than for others. In our own work, the intrinsic speed of the interval arithmetic has not been the primary limiting factor, although it could be in algorithms in which other limitations have been ameliorated or in problems in which those limitations are not important. The speed of interval arithmetic would not be

an issue even then, since various efficient implementations, such as [34], exist.

We have not seen significant differences in the tools and techniques used between software that claims to be mathematically rigorous and software that uses branch and bound, but does not attempt to be mathematically rigorous. Thus, we have not seen significant impediments to implementing any of the techniques used by such non-rigorous software in a way that is mathematically rigorous; however, with more careful definition of the kind of solution the software attempts to deliver, more problems might be seen to be intrinsically difficult if the type of solution to be delivered must satisfy requirements that are too strict. For instance, if it is too difficult to verify exact feasibility of equality constraints, it is usually easy to verify that they are ϵ-feasible, that is, that the magnitudes of their residuals are less than ϵ.

Besides this issue with solution type, it is our belief that any remaining advantages of software that is not mathematically rigorous are due to the skill of the researchers and software developers at incorporating and combining various heuristics and techniques that can be used in either mathematically rigorous or non-rigorous software. That is, the onus is on developers of mathematically rigorous software to increase performance of their products, using the same tools as developers not striving for rigor, but with extra care to encompass roundoff error.

One question concerning how techniques used in non-rigorous software can be incorporated into rigorous software is how to use the optimal points to relaxations. Those optimal points give non-rigorous approximations to optimal points of the original problem. Theorem 5.2 shows that optimal points of relaxations are actually near optimal points of the original problem, thus giving some guidance towards constructing rigorous bounds on optimal points.

6.2. *Future Work*

Some possibilities for improving the performance of mathematically rigorous software are as follows.

- Implement efficient techniques and heuristics, such as the convex approximations from [14]. This will involve careful construction of rigorous versions of libraries for computing these relaxations.
- Carefully investigate alternatives to constructing relaxations. For instance, the relaxation can be constructed based on individual operations (fine granularity, as we have illustrated in [27]), or we may use techniques with coarser granularity, such as those of [14] or [49].
- Develop the ideas we have presented in §5 for automated computation of the distance of the solution of relaxations to the solutions of the practical problem.
- Study applications to specific problems. For example, in [12], a large-scale nonconvex problem was presented that, with appropriate preliminary analysis, could be solved effectively without *any* subdivision of the full-dimensional region in $\mathbb{R}^n$, but by simply solving carefully constructed relaxations. It should be straightforward to implement such a computation so the result is mathematically rigorous, even for instances of large-scale problems.
- Software designers are encouraged to state in clear mathematical terms the quality of solutions (as we have outlined in §2.2). Along these lines, rigorous software might be made to perform better by designing it to enclose verified ϵ-approximate solutions as in (2), rather than exact solutions.
- Continue the work in [42] to analyze and enclose non-isolated solution sets.

Acknowledgements

I wish to thank the referees, whose very careful reading and comments led to both significant clarification of the paper and avoidance of embarrassing errors.

References

[1] A.S. Ackleh et al., *Classical and Modern Numerical Analysis: Theory, Methods, and Practice*, Taylor and Francis, Boca Raton, Florida (2009).
[2] M. Affenzeller et al., *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications*, CRC Press, Boca Raton, FL, USA (2009).
[3] H. Beeck, *Linear programming with inexact data*, Institutsbericht TUM-ISU-7830, Inst. F. Statistik und Unternehmensforschung, Techn. Universität München (1978).
[4] F. Benhamou and A. Colmerauer (eds.), *Constraint Logic Programming: Selected Research*, MIT Press, Cambridge, MA, USA (1993).
[5] G. Borradaile and P.V. Hentenryck, *Safe and tight linear estimators for global optimization*, Math. Program. 102 (2005), pp. 495–517.
[6] M. Ceberio, *"Constraint Solving 2.0" web page* (2009), URL `http://www.constraintsolving.com/`.
[7] J.E. Dennis, Jr. and R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, *Classics in Applied Mathematics*, vol. 16, SIAM, Philadelphia, PA (1996).
[8] L.C.W. Dixon and G.P. Szego (eds.), *Towards Global Optimization*, North-Holland (1975).
[9] ———, *Towards Global Optimization 2*, North-Holland (1977).
[10] M. Fiedler et al., *Linear Optimization Problems with Inexact Data*, Springer (2006).
[11] C. Floudas, *Deterministic global optimization: Theoretical, computational and implementation advances* (2002), plenary Lecture, Global Constrained Optimization and Constraint Satisfaction Conference, Cocos'02, Valbonne - Sophia Antipolis, France.
[12] ———, *Deterministic global optimization: Advances and challenges* (2009), plenary Lecture, First World Congress on Global Optimization in Engineering and Sciences, WCGO-2009.
[13] C. Floudas and P. Pardalos (eds.), *Optimization in Computational Chemistry and Molecular Biology: Local and Global Approaches*, Kluwer Academic Publishers, Dordrecht, The Netherlands (2000).
[14] C.F. Floudas, *Deterministic Global Optimization: Theory, Methods, and Applications*, Kluwer Academic Publishers, Dordrecht, The Netherlands (2000).
[15] R.A. Friesner, *Computational Methods For Protein Folding*, John Wiley and Sons, New York (2002).
[16] P.E. Gill, W. Murray, and M. Wright, *Practical Optimization*, Academic Press, New York, NY, USA (1981).
[17] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1989).
[18] S. Hongthong and R.B. Kearfott, *Rigorous linear overestimators and underestimators*, technical report, `http://interval.louisiana.edu/preprints/estimates_of_powers.pdf`.
[19] C. Jansson, *Zur Linearen Optimierung mit unscharfen Daten*, Ph.D. dissertation, UniversitätKaiserslautern (1985).
[20] ———, *Rigorous lower and upper bounds in linear programming*, SIAM J. on Optimization 14 (2003), pp. 914–935.
[21] C. Jansson and S.M. Rump, *Rigorous solution of linear programming problems with uncertain data*, ZOR — Methods and Models of Operations Research 35 (1991), pp. 87–111.
[22] R.B. Kearfott, *Decomposition of arithmetic expressions to improve the behavior of interval iteration for nonlinear systems*, Computing 47 (1991), pp. 169–191.
[23] ———, *Algorithm 763: INTERVAL_ARITHMETIC: A Fortran 90 module for an interval data type*, ACM Transactions on Mathematical Software 22 (1996), pp. 385–392, see [29].
[24] ———, *Rigorous Global Search: Continuous Problems*, no. 13 in Nonconvex optimization and its applications, Kluwer Academic Publishers, Dordrecht, Netherlands (1996).
[25] ———, *On proving existence of feasible points in equality constrained optimization problems*, Math. Program. 83 (1998), pp. 89–100.
[26] ———, *GlobSol user guide*, Optimization Methods and Software 24 (2009), pp. 687–708.
[27] R.B. Kearfott and S. Hongthong, *Validated linear relaxations and preprocessing: Some experiments*, SIAM J. on Optimization 16 (2005), pp. 418–433.
[28] R.B. Kearfott and M. Novoa III, *Algorithm 681: INTBIS, a portable interval Newton/bisection package*, ACM Transactions on Mathematical Software 16 (1990), pp. 152–157.
[29] R.B. Kearfott et al., *Algorithm 737: INTLIB: A portable Fortran-77 elementary function library*, ACM Transactions on Mathematical Software 20 (1994), pp. 447–459, see companion interval arithmetic package [23].
[30] ———, *Libraries, tools, and interactive systems for verified computations: Four case studies*, Lecture Notes in Computer Science 2991 (2004), pp. 36–63.
[31] S. Kirkpatrick, *Optimization by simulated annealing: Quantitative studies*, J. of Statistical Physics 34 (1984), pp. 975–986.
[32] S. Kirkpatrick et al., *Optimization by simulated annealing*, Science 220 (1983), pp. 671–680.
[33] R. Krawczyk, *Fehlerabschätzung Bei Linearer Optimierung*, in *Interval Mathematics*, K. Nickel, ed., *Lecture Notes In Computer Science*, vol. 29, Springer Verlag, Berlin-Heidelberg-New York (1975), pp. 215–222.
[34] B. Lambov, *Interval arithmetic using SSE-2*, in *Reliable Implementation of Real Number Algorithms: Theory and Practice: International Seminar Dagstuhl Castle, Germany, January 8-13, 2006 Revised Papers*, Springer-Verlag, Berlin, Heidelberg (2008), pp. 102–113.

[35] A.V. Levy and A. Montalvo, *The tunneling algorithm for the global minimization of functions*, SIAM Journal on Scientific and Statistical Computing 6 (1985), pp. 15–29.

[36] C. Maranas and C. Floudas, *Finding all solutions of nonlinearly constrained systems of equations*, Journal of Global Optimization 7 (1995), pp. 143–182.

[37] R.E. Moore, R.B. Kearfott, and M.J. Cloud, *Introduction to Interval Analysis*, SIAM, Philadelphia, PA (2009).

[38] A. Neumaier, *Interval Methods for Systems of Equations*, *Encyclopedia of Mathematics and its Applications*, vol. 37, Cambridge University Press, Cambridge, UK (1990).

[39] A. Neumaier and O. Shcherbina, *Safe bounds in linear and mixed-integer programming*, Math. Prog. 99 (2004), pp. 283–296.

[40] A. Neumaier et al., *A comparison of complete global optimization solvers*, Math. Prog. 103 (2005), pp. 335–356.

[41] F. Rossi, P. van Beek, and T. Walsh, *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*, Elsevier Science Inc., New York, NY, USA (2006).

[42] J. Roy, *Singularities in deterministic global optimization*, Ph.D. dissertation, Department of Mathematics, University of Louisiana, Lafayette, LA, USA (2010).

[43] J. Roy and R.B. Kearfott, *Global optimization and singular nonlinear programs: New techniques* (2009), submitted to the proceedings of SCAN 2008.

[44] S.M. Rump, *Solving algebraic problems with high accuracy*, in *A New Approach To Scientific Computation*, U.W. Kulisch and W.L. Miranker, eds., Academic Press, New York (1983), pp. 51–120.

[45] N.V. Sahinidis, *BARON: A general purpose global optimization software package*, J. Global Optim. 8 (1996), pp. 201–205.

[46] P. Salamon, R. Frost, and P. Sibani, *Facts, Conjectures, and Improvements for Simulated Annealing*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2002).

[47] M.A. Stadtherr, *Interval analysis: Application to chemical engineering design problems*, in *Encyclopedia of Optimization*, A. Iserles, ed., Kluwer Academic Publishers (2001).

[48] P.J. Stuckey (ed.), *Principles and Practice of Constraint Programming, 14th International Conference, CP 2008, Sydney, Australia, September 14-18, 2008. Proceedings*, *Lecture Notes in Computer Science*, vol. 5202, Springer (2008).

[49] M. Tawarmalani and N.V. Sahinidis, *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*, Kluwer Academic Publishers, Dordrecht, Netherlands (2002).

[50] ———, *Global optimization of mixed-integer nonlinear programs: A theoretical and computational study*, Math. Program. 99 (2004), pp. 563–591.

[51] P. Van Hentenryck, L. Michel, and Y. Deville, *Numerica: A Modeling Language for Global Optimization*, The MIT Press, Cambridge, MA, USA (1997).