Noname manuscript No.
(will be inserted by the editor)

On Rigorous Upper Bounds to a Global Optimum

Ralph Baker Kearfott

Received: date / Accepted: date

Abstract In branch and bound algorithms in constrained global optimization, a sharp upper bound on the global optimum is important for the overall efficiency of the branch and bound process. Software to find local optimizers, using floating point arithmetic, often computes an approximately feasible point close to an actual global optimizer. Not mathematically rigorous algorithms can simply evaluate the objective at such points to obtain approximate upper bounds. However, such points may actually be slightly infeasible, and the corresponding objective values may be slightly smaller than the global optimum. A consequence is that actual optimizers are occasionally missed, while the algorithm returns an approximate optimum and corresponding approximate optimizer that is occasionally far away from an actual global optimizer. In mathematically rigorous algorithms, objective values are accepted as upper bounds only if the point of evaluation is proven to be feasible.

Such computational proofs of feasibility have been weak points in mathematically rigorous algorithms. This paper first reviews previously proposed automatic proofs of feasibility, then proposes an alternative technique. The alternative technique is tried on a test set that caused trouble for previous techniques, and is also employed in a mathematically rigorous branch and bound algorithm on that test set.

Keywords automatic verification, branch and bound algorithms, interval analysis, global optimization, feasibility

1 Introduction and Notation

Branch and bound algorithms for constrained global optimization benefit from utilizing iterative constrained local optimization software. Such solvers find an approximately feasible point $x^{\approx}$ that is near a local optimizer. Thus, objective function evaluations $\varphi(x^{\approx})$ at such points $x^{\approx}$ are approximate upper bounds $\overline{\varphi}$ on the global optimum φ^*;

Department of Mathematics, University of Louisiana at Lafayette, U.L. Box 4-1010, Lafayette, LA 70504-1010 USA
Tel.: 337-482-5270
Fax: 337-482-5346
E-mail: rbk@louisiana.edu

often $|\varphi(x^{\approx}) - \varphi^*|$ is also small. Such upper bounds $\overline{\varphi}$ are then used in branch and bound algorithms in conjunction with lower bounds on the range of the objective in the feasible portion of a subregion of the search space: If $\underline{\varphi}(\boldsymbol{x}) > \overline{\varphi}$, where $\underline{\varphi}(\boldsymbol{x})$ is a computed lower bound on the range of the objective φ over a subregion $\boldsymbol{x}$ of the search region, then $\boldsymbol{x}$ is eliminated from further processing in the branch and bound algorithm.

When points $x^{\approx}$ are taken to be feasible, it is not unusual that $\varphi(x^{\approx}) < \varphi^*$, although $\varphi(x^{\approx}) \approx \varphi^*$. In such cases, the branch and bound algorithm typically gives a good approximation to φ^* and an optimizing point $x^{\approx}$ that is near a global optimizer x^*. However, there are cases where $x^{\approx}$ is not near any globally optimizing point x^*, and the exact relationship is not revealed by the computation.

The goal of some branch and bound algorithms is to simply find a good approximation to φ^*, while in others, it is to find all optimizing points (or reveal degeneracy in the solution). Occasionally in the first case but often in the second case, use of $x^{\approx}$, rather than a point that is exactly feasible, gives unpredictable results. To avoid such problems, algorithms rigorously prove that a nearby point $x^{\mathrm{f}} \approx x^{\approx}$ is feasible, then obtain an upper bound on $\varphi(x^{\mathrm{f}})$ either by evaluating φ at x^{f} or over a small box $\boldsymbol{x}^{\mathrm{f}}$, $x^{\mathrm{f}} \in \boldsymbol{x}^{\mathrm{f}}$, using interval arithmetic or directed rounding.

Several processes have been proposed for obtaining x^{f} or $\boldsymbol{x}^{\mathrm{f}}$, but their success on published test sets has been limited. This limited success explains, for many problems, the poor performance of mathematically rigorous branch and bound algorithms vis à vis algorithms utilizing values $\varphi(x^{\approx})$ to obtain (only approximate) upper bounds on the global optimum. This work deals with such methods for obtaining feasibility at points x^{f} or within boxes $\boldsymbol{x}^{\mathrm{f}}$. Previously proposed methods are briefly reviewed in §2, along with explanations of why they tend to fail on standard test sets. We propose an improved technique in §3, and we test the new technique's ability to prove feasibility on published test problems over which the other techniques failed in §4.1 and §4.4. The effectiveness of this improved availability of a sharp rigorous upper bound in a branch and bound algorithm is illustrated with the experimental results in §4.2. We compare our scheme to feasibility-restoration computations in penalty-function-free nonlinear programming packages in §5. We summarize the work in §6.

2 Previous Work and Pitfalls

In a general global optimization problem of the form

$$\begin{aligned} &\text{minimize } \varphi(x) \\ &\text{subject to } c_i(x) = 0, i = 1, \ldots, m_1, \\ &\qquad\qquad g_i(x) \le 0, i = 1, \ldots, m_2, \\ &\text{where } \varphi : \mathbb{R}^n \to \mathbb{R} \text{ and } c_i, g_i : \mathbb{R}^n \to \mathbb{R}, \end{aligned} \tag{1}$$

checking feasibility of the inequality constraints $g(x) \le 0$ at a point $\check{x}$ in principle can be done by simply evaluating $g(\check{x})$ using directed rounding or interval arithmetic, while it is impossible with floating point arithmetic to determine feasibility of equality constraints $c(x)$ at a point. Even in the absence of equality constraints, some of the inequality constraints are active at optimizing points x^*, and an evaluation using floating point arithmetic (or floating point with directed rounding or interval arithmetic) cannot rigorously verify feasibility of a point $\check{x}$. Furthermore, the point $\check{x}$ may only be approximately feasible, and thus need be either perturbed to a feasible point or enclosed in a box that contains feasible points. In this section, we discuss techniques for

1. rigorously handling only approximately feasible points, and
2. handling equality constraints and active inequality constraints

that have previously been proposed.

2.1 Direct Proof of Feasibility

In [5] and [4, §5.2.4], we assumed there were only $m_1 \le n$ active constraints, that is, inactive inequality constraints $g_i(\check{x}) < 0$ can be proven so with an interval evaluation at $\check{x}$ and ignored, active inequality constraints $g_i(\check{x}) \approx 0$ are lumped with the equality constraints and treated as equality constraints, and the total number m_1 of such active inequality constraints and equality constraints is less than the number of variables n. Then, examining the gradients of the active constraints, we choose those m_1 coordinates $\{i_1, \ldots, i_{m_1}\}$ of $\check{x}$ in which the active constraints are most sensitive, and do the following:

1. Form $\check{x}^{(m_1)} = (\check{x}_{i_1}, \ldots, \check{x}_{i_{m_1}}) \in \mathbb{R}^{m_1})$ and define $\tilde{c}(x_{i_1}, \ldots, x_{i_{m_1}}) \in \mathbb{R}^{m_1}) : \mathbb{R}^{m_1} \to \mathbb{R}^{m_1}$ by fixing the remaining $n - m_1$ parameters x_i, $i \notin \{i_1, \ldots, i_{m_1}\}$ of c at $\check{x}_i$.
2. Construct a small box $\boldsymbol{x}^{(m_1)} \subset \mathbb{R}^{m_1}$, $\check{x}^{(m_1)} \in \boldsymbol{x}^{(m_1)}$ large enough likely to contain a feasible point of $\tilde{c}$ but small enough that all of the inequality constraints inactive at $\check{x}$ remain active at points in $\mathbb{R}^n$ corresponding to points in $\boldsymbol{x}^{m_1}$.
3. Apply an interval Newton method over $\boldsymbol{x}^{m_1}$ to prove existence to a solution of $\tilde{c} = 0$ within $\boldsymbol{x}^{(m_1)}$.
4. Evaluate $\tilde{\varphi}(\boldsymbol{x}^{(m_1)})$ (that is, use interval arithmetic to evaluate φ over the possibly degenerate box whose i_j-th coordinate is the corresponding coordinate of $\boldsymbol{x}^{(m_1)}$ for $i_j \in \{i_1, \ldots, i_{m_1}\}$ and whose remaining coordinates are cooresponding coordinates of $\check{x}$), to obtain a mathematically rigorous upper bound on the global minimum of φ.

In principle, this scheme should work well if $m_1 \le n$ and the gradients of the active constraints are linearly independent. However, despite theory to the contrary (such as the "usual" case for the fundamental theorem of linear programming, for linear problems), there are many cases within published test sets where there are more than n approximately active constraints, or where the gradients of the active constraints are linearly dependent. Thus, we have found this scheme to be unsuccessful for many problems. We believe the underlying scheme we propose in the present work to be simpler to implement in a robust way, for general problems.

2.2 Use of the Kuhn–Tucker or Fritz John Conditions

In principle, one may use an interval Newton method to prove existence of a solution to the system of $n + m_1 + m_2$ equations in in the $n + m_1 + m_2$ unknowns corresponding to the Kuhn–Tucker conditions (or the $n + m_1 + m_2 + 1$ equations and unknowns corresponding to the Fritz John conditions). However, our experience is that this often fails. Success of an interval Newton method naively applied requires non-singularity of the Jacobian matrix of the system of equations; it is known, for example, that the Fritz John system corresponding to common reformulations of minimax problems is ill-conditioned or singular if the fit is good or perfect, respectively (see [9]). Furthermore, the Kuhn–Tucker or Fritz John Jacobian matrix necessarily is singular if the matrix of active constraint gradients is singular, and we have found the active constraint gradients to be commonly linearly dependent in practice.

2.3 Relaxation and Point Evaluation

To avoid the issue of rigorously proving exact feasibility of equality constraints (and corresponding pitfalls of interval Newton methods), some experts in mathematically rigorous branch and bound methods have proposed transforming all equality constraints to inequality constraints, then relaxing them to obtain a feasible region with non-empty interior, then simply evaluating the objective φ with interval arithmetic at points in the interior of that region to obtain mathematically rigorous upper bounds $\overline{\varphi}$ to the global optimum. That is, we do the following:

1. Replace each $c_i = 0$ by two inequality constraints $c_i \leq 0$ and $-c_i \leq 0$.
2. Replace each $c_i \leq 0$ by $c_i - \epsilon_r \leq 0$ and $-c_i \leq 0$ by $-c_i - \epsilon_r \leq 0$, where the relaxation parameter ϵ_r is small in relation to the accuracy of the coefficients in the problem but large in relation to the domain tolerance passed to the iterative local optimization software.

For example Messine, Ninin et al have reported substantial success with a novel branch and bound algorithm that also employs this technique [11].

Step 1 of this process results in an equivalent problem, while step 2 replaces the problem by a nearby problem whose optimum is lower than the optimum of the original problem. One may argue that mathematically rigorous solution of a nearby problem may not say much rigorously about solution of the original problem. However, due to the inability to obtain sharp yet correct upper bounds $\overline{\varphi}$ for the original problem, the nearby problem may be much easier to solve with mathematical certainty. Furthermore, the meaning of the solution of the nearby problem can be interpreted precisely in terms of classical backward error analysis (that is, we know exactly what perturbation of the original problem gives us the solutions we compute), whereas it is uncertain what the meaning of the solution is when solving the original problem without mathematical rigor, that is, by only using approximate upper bounds $\overline{\varphi}$. We have done some analysis of the relationship of the solution of nearby problems to the solutions of the original problem in [7].

We adopt this philosophy, that is, we work with a nearby relaxed problem containing only inequalities, in the alternative technique we propose.

3 An Alternative Technique

First, we have observed that standard test problems, in addition to having equality constraints, are sometimes presented in a form in which at least some pairs of inequality constraints correspond to equality constraints. In such cases, not only should we replace each $c_i = 0$ by $c_i - \epsilon_r \leq 0$ and $-c_i - \epsilon_r \leq 0$, but we should also replace each original inequality constraint $g_i \leq 0$ by $g_i - \epsilon_r \leq 0$.

A second issue is how to obtain an exactly feasible point x^{f} by perturbing an approximately feasible point $x^{\approx}$. To develop the scheme presented here in Algorithm 1, we considered various properties of practical problems as they are commonly formulated. These properties are as follows:

1. Problems are often formulated with many redundant constraints , resulting in more approximately active constraints than variables.
2. Only a few constraints are active in some problems.
3. Pairs of constraints corresponding to the two sides of an inequality constraint often occur.
4. At an approximate optimizing point, some constraints that would be active at the exact optimizing point can be only approximately active and, while other constraints can be approximately active but infeasible.

We would like to perturb an approximately feasible point in a direction in which all of the infeasible constraint values are decreasing past 0. If the number of constraints equalled the number of variables, an overrelaxation of the Newton step for the system obtained by setting the constraint gradients equal to zero would be logical. However, in many problems, we have Property 1, while in others, we have Property 2; this suggests replacing an overrelaxation of the Newton step by an overrelaxation of a Gauss–Newton step. Property 4 suggests we consider subspaces of feasible and infeasible approximately active constraints, and perform the overrelaxed Gauss–Newton step in the orthogonal complement of the space spanned by the gradients of the feasible approximately active constraints. Finally, Property 3 suggests that, in many if not most cases, we may want to take a full Gauss–Newton step anyway (into the bands represented by relaxed equality constraints), but we need to carefully balance the amount by which such constraints has been relaxed, the size of the overrelaxed Gauss–Newton step, and the tolerance of the approximate optimizer. These considerations logically result in Algorithm 1.

In summary, Algorithm 1 does the following:

1. Over-project into the interior of the feasible region of the violated constraints if that will not make any approximately active constraints infeasible.
2. If approximately active constraints would be made infeasible by such an over-projection, take a cautious step to a measured distance from the boundaries of the approximately active constraints in the direction of the projection to the boundary of the infeasible constraints, and follow that by a step in the null space of the approximately active constraints slightly beyond the minimum of sum of squares of the infeasible constraints within that null space.
3. Using interval arithmetic or appropriately directed rounding, evaluate the constraints at the so-perturbed point to rigorously check feasibility.

In Algorithm 1, for succinctness, we use the notation $\boldsymbol{g}(x)$ to denote an interval evaluation of g at x, used to rigorously determine that the components of g are non-positive, once this has already been determined with a floating point approximation or after a perturbation likely to make x feasible.

Algorithm 1 is robust in the sense that it uses the natural scaling of Newton's method, and it will tend to work even in the presence linearly dependent constraint gradients. There are only several heuristic parameters in this algorithm: the factor of 10, used in step 3 in determining which constraints are approximately active, the factor

```
Input  : An approximately feasible point $x^{\approx}$, an overrelaxation parameter $\omega$, and
         the inequality constraints $g : \mathbb{R}^n \to \mathbb{R}^m$, $g(x) \le 0$.
Output : OK = true and the feasible point $x^{\mathrm{f}}$ if a feasible point was found,
         OK = false if a feasible point was not found.
1  Using floating point arithmetic, identify possibly infeasible constraints $g_i(x^{\approx}) > 0$,
   $i \in \mathcal{I} = \{i_j\}_{j=1}^{m_{\mathcal{I}}}$ and set $M_{\mathcal{I}} = \max_{i \in \mathcal{I}}\{g_i(x^{\approx})\}$;
2  if $\boldsymbol{g}(x^{\approx}) \le 0$ then return OK ← true and $x^f \leftarrow x^{\approx}$ else return OK ← false;
   /* Using floating point arithmetic, identify approximately active but not
      infeasible constraints:                                                 */
3  for $= 1$ to $m$, $i \notin \mathcal{I}$ do if $|g_i|(x^{\approx}) < 10 M_{\mathcal{I}}$ then append $i$ to $\mathcal{A}$;
4  Form a matrix $G_{\mathcal{I}} \in \mathbb{R}^{m_{\mathcal{I}} \times n}$ whose rows are the floating point gradients at $x^{\approx}$ of
   the infeasible constraints and a vector $g_{\mathcal{I}}(x^{\approx})$ whose components are the values of
   the infeasible constraints;
5  Compute the Gauss–Newton step $w_{\mathrm{full}}$ for $g_{\mathcal{I}} = 0$ by computing (in general) the
   least squares solution of minimum norm to $G_{\mathcal{I}} w_{\mathrm{full}} = g_{\mathcal{I}}$;
   /* Compute the maximum multiple of the Gauss--Newton step which will not
      make approximately active constraints infeasible:                      */
6  $T \leftarrow \min_{i \in \mathcal{A}}\{-g_i(x^{\approx})/(\nabla g_i(x^{\approx})^T w_{\mathrm{full}})\}$;
7  if $T \ge \omega$ then            /* Take a full overrelaxed Gauss--Newton step */
8      $T \leftarrow \min\{T, \omega\}$; $x^{\mathrm{f}} \leftarrow x^{\approx} + T w_{\mathrm{full}}$;
9      if $\boldsymbol{g}(x^f) \le 0$ then return OK ← true and $x^f$ else return OK ← false;
10 else   /* Take a partial Gauss--Newton step then a subspace correction */
11     $x^{\mathrm{f}} \leftarrow x^{\approx} + 0.9 T w_{\mathrm{full}}$;
12     Remove from $\mathcal{A}$ those $i$ with $(\nabla g_i)^T w_{\mathrm{full}} < 0$, leaving $|\mathcal{A}| = m_{\mathcal{A}} > 0$ elements;
13     Form the matrix $G_{\mathcal{A}}$ whose rows are the gradients $(\nabla g_i(x^{\approx}))^T$, $i \in \mathcal{A}$;
14     Compute a matrix $V \in \mathbb{R}^{n \times n_{\mathcal{N}}}$ whose columns form an orthonormal basis for
       the null set of $G_{\mathcal{A}}$;
15     Form $G_{\mathcal{I},V} = G_{\mathcal{I}} V \in \mathbb{R}^{m_{\mathcal{I}} \times n_{\mathcal{N}}}$      /* Jacobian matrix of the violated
       constraints with respect to the null space coordinates */;
16     Compute a Gauss–Newton step restricted to the null space of the
       approximately active constraints by computing the least squares solution of
       minimum norm to $G_{\mathcal{I},V} p = -g_{\mathcal{I}}$;
17     $x^{\mathrm{f}} \leftarrow x^{\mathrm{f}} + \omega V p$;
18     if $\boldsymbol{g}(x^f) \le 0$ then return OK ← true and $x^f$ else return OK ← false;
19 end
```

Algorithm 1: Perturbing an approximate feasible point to feasibility

of 0.9 in step 11 used to determine the maximum amount of movement of the approximately active constraint values towards infeasibility, and the criterion used to determine the numerical rank of the matrices in the Gauss–Newton steps and in computing null spaces. The factor of 10 in step 3 can be replaced, with additional complication, by a logically computed value depending on the gradients of the constraints, the underrelaxation parameter, presently 0.9, in step 11, and the overrelaxation parameter ω. We have used the singular value decomposition in computing the Gauss–Newton steps and null spaces, and have deemed a singular value σ_i to be effectively zero if $\sigma_i/\sigma_1 < 100\epsilon_m$, where ϵ_m is the machine epsilon.

In Algorithm 1, a single Gauss–Newton step should be adequate in correcting an output $x^{\approx}$ of a floating point constrained optimizer, since the distance between the approximate optimizer and an actual critical point is usually small enough that the constraints can be considered linear.

Even though floating point arithmetic is used in steps 6 and 12, $m_{\mathcal{A}} = |\mathcal{A}|$ must be positive (and not zero) after step 12. This the only way $|\mathcal{A}| = 0$ is if

$$-g_i(x^{\approx})/(\nabla g_i(x^{\approx})^T w_{\text{full}} < 0$$

for each i for which $g_i(x^{\approx}) < -10M_{\mathcal{I}} < 0$, which cannot happen if T was computed to be finite in step 6, and the same floating point values are used in steps 3, 6, and 12.

4 Numerical Experiments

In our recent work [8], we examined those problems from the COCONUT Lib-1 test set [14,10] whose non-convexities lay in subspaces of dimension 4 or less, to compare our GlobSol branch and bound software [6] to a simple process in which subdivision of the domain was determined a priori. GlobSol runs the local solver IPOPT [15] to obtain an upper bound on the global optimum before the beginning of the branch and bound process and at various points, including when a box with sufficiently small diameter is produced during the branch and bound process. In the experiments in [8], GlobSol attempts to prove feasibility from the return of the local solver first by using the Fritz John system as in §2.2, and, failing that, attempts to determine feasibility directly as in §2.1. Of the 77 problems of non-convex dimension 4 or less in the Lib-1 test set, GlobSol was unable to prove feasibility a single time in 28, and of these 28, GlobSol's branch and bound algorithm completed within 2 hours in only 8. However, GlobSol's branch and bound algorithm failed to complete in only 6 of the remaining 49 problems in which feasiblity was rigorously proven, and hence a reasonable upper bound on the global optimum was obtained. This highlights both the importance of a good upper bound on the global optimum in the branch and bound process and the difficulty of obtaining a mathematically rigorous upper bound in general, but also the existence of other reasons for inefficiencies. We explore this further here.

We took the 28 problems for which proof of feasibility in the aforementioned experiments failed as our test set. In addition, we included three additional problems, `ex5.2.2-case-1`, `ex5.2.4`, and `ex9.1.6`, since they were similar to problems for which feasibility verification had failed, and since feasibility verification had failed in earlier versions of GlobSol.

To prepare the problems for Algorithm 1, we transformed the AMPL files posted on [10] into Fortran source files with a text editor, and, to assure blunders were not made, compared results with the posted results on [10]. Following the reasoning in §2.3 and §3, we relaxed both the equality and inequality constraints. Recapitulating:

1. We replaced each equality constraint $c_i = 0$ by two inequality constraints $c_i - \epsilon_r \le 0$ and $-c_i - \epsilon_r \le 0$.
2. We replaced each inequality constraint $g_i \le 0$ by $g_i - \epsilon_r \le 0$. (See our observation in the first paragraph of §3.)

Additionally, several of the variables in problems `ex9.1.3`, `ex9.1.6`, `ex9.1.7`, and `ex9.1.9` were specified to be binary. We ignored these conditions (that is, we relaxed the problems to continuous problems), since conditions for mixed integer programs are not presently utilized in GlobSol's branch and bound algorithm, and since the feasibility verification deals with the continuous variables.

In all of our experiments, we used $\epsilon_r = 10^{-4}$. We included bound constraints as inequality constraints, and relaxed them, too. We used the midpoints of the starting

Table 1 Comparison of Algorithm 1 and use of the Fritz John conditions.

Prob.	n	m	$x^{\approx}$ OK?	$m_{\mathcal{A}}$	$m_{\mathcal{I}}$	$n_{\mathcal{N}}$	$x^{\approx}$ feas?	NC?	A. 1 OK?	FJ OK?
ex14.1.1	3	8	T	0	4	0	F	F	T	F
ex14.1.8	3	8	T	0	1	0	F	F	T	F
ex14.2.1	5	17	T	0	1	0	F	F	T	F
ex14.2.2	4	13	T	0	1	0	F	F	T	F
ex14.2.4	5	17	T	0	1	0	F	F	T	F
ex14.2.5	4	13	T	0	1	0	F	F	T	F
ex14.2.6	5	17	T	0	1	0	F	F	T	F
ex14.2.8	4	13	T	0	1	0	F	F	T	F
ex14.2.9	4	13	T	0	1	0	F	F	T	F
ex2.1.4	6	15	T	0	6	0	F	F	T	F
5.2.2.1	9	28	T	0	9	0	F	F	T	F
5.2.2.2	9	28	T	0	9	0	F	F	T	F
ex5.2.4	7	21	T	0	7	0	F	F	T	T
ex7.3.3	5	12	T	0	5	0	F	F	T	T
ex7.3.5	13	28	T	0	3	0	F	F	T	F
ex8.1.7	5	16	T	0	3	0	F	F	T	T
ex9.1.2	10	28	T	3	5	7	F	T	T	F
ex9.1.3	29	94	F	0	0	0	F	F	F	F
ex9.1.4	10	36	T	0	8	0	F	F	T	F
ex9.1.6	20	76	F	0	0	0	F	F	F	F
ex9.1.7	23	61	F	0	0	0	F	F	F	F
ex9.1.9	17	44	T	0	3	0	F	F	T	F
ex9.2.4	8	26	T	0	6	0	F	F	T	F
ex9.2.8	6	21	T	0	5	0	F	F	T	F
harker	20	34	T	0	7	0	F	F	T	F
haverly	12	30	T	1	2	1	F	T	F	F
house	8	17	T	0	6	0	F	F	T	F
immun	21	35	T	0	0	0	T	F	T	F
qp5	108	169	T	0	8	0	F	F	T	T
sambal	17	20	T	0	5	0	F	F	T	F
sample	4	10	T	0	2	0	F	F	T	T
Totals			28				1	2	27	5

boxes as initial guess for the local solver, when we were testing the feasibility verification techniques outside of the branch and bound algorithm.

4.1 Results for Proof of Feasibility

In this set of results, we ran the point solver IPOPT once, noted if the solver returned without error, then attempted to rigorously prove feasibility. In one set of experiments, we used Algorithm 1, while in another set, we constructed a small box about $x^{\approx}$ (returned by IPOPT) and attempted to use an interval Newton method to prove a solution to the Fritz John equations existed within that box; this is the main technique GlobSol had been using in the past. The results appear in Table 1. In Table 1, the columns are as follows:

– n is the number of variables.
– m is the number of inequality constraints after relaxation of equality constraints.
– "$x^{\approx}$ OK?" is "T" if the local solver returned without an error signal.
– $m_{\mathcal{A}}$ is the number of approximately active constraints at the point $x^{\approx}$ returned by the local solver.

– $m_{\mathcal{I}}$ is the number of infeasible constraints at $x^{\approx}$.
– $n_{\mathcal{N}}$ is the dimension of the null space of the gradients of the approximately active constraints, as in step 14 of Algorithm 1.
– "$x^{\approx}$ feas?" is "T" if the point $x^{\approx}$ returned by the local solver is feasible without further processing.
– "NC?" is "T" if the overrelaxed Gauss–Newton step in n-space from step 8 of Algorithm 1 is not sufficient, and a correction step in the null space of the active constraints is needed.
– "A. 1 OK?" is "T" if Algorithm 1 succeeded in perturbing $x^{\approx}$ to a point x^{f} at which the constraints could be rigorously proven to be feasible.
– "FJ OK?" is "T" if an interval Newton applied to the Fritz John system over a small box about $x^{\approx}$ was able to prove existence of a critical point, as in §2.2.

We observe the following.

- The point solver IPOPT (version 3.10.1) failed for only 3 problems.
- The point solver returned a feasible point in one instance.
- From the 27 problems where $x^{\approx}$ was available but $x^{\approx}$ was not exactly feasible, Algorithm 1 succeeded in 26 of these. In the remaining one, `haverly`, a single constraint remained infeasible, although it had been improved by both the partial Gauss–Newton step and by the correction in the null space.
- The interval Newton method proved existence of a critical point of the Fritz John system in only 5 of the 27 instances, and Algorithm 1 was also successful for these five cases. This is not surprising in view of the fact the Fritz John system is typically singular or ill-conditioned at solutions, such as in [9].
- Although we did not compare handling the inequality constraints directly as in §2.1, examination of the results in [8], where direct feasibility verification was tried if proof of existence of a critical point failed, indicates direct feasibility verification only occasionally succeeded with problems as formulated in this test set.
- The Gauss–Newton step in the full space, that is, step 8 of Algorithm 1, succeeded in all but two cases. In those two cases, the correction step in the null space of the active constraints succeeded in one and failed in the other.

Numerical comparisons with the technique from [5] were skipped. However, preliminary experiments hinted that the technique seldom worked with the formulations in this test set. Typical difficulties are that $m_{\mathcal{A}} + m_{\mathcal{I}}$, the number of equations in the system for the interval Newton method in [5], is larger than n, and that the gradients of the active constraints are linearly dependent. Furthermore, a perturbation is still in general also required for the technique from [5], and the perturbation technique proposed in Algorithm 1 is more general and simpler than that heretofore tried with the technique from [5]. The technique from §2.1 and [5] is appropriate when there is a small number of equality constraints (or potentially active constraints) and the set of gradients of active constraints is known to always be linearly dependent. These conditions do not hold for the selected test set.

4.2 Results within a Branch and Bound Algorithm

Section 4.1 illustrated the effectiveness of Algorithm 1 at proving that a nearby point was feasible, given an approximately feasible point returned by the local optimization

Table 2 Comparison of Alg. 1 and the Fritz John conditions in a branch and bound algorithm.

Problem	P?	FJ?	P F?	FJ F?	P+	FJ+	P T	FJ T	P boxes	FJ boxes
ex14.1.1	T	F	T	T	F	F	0.98	15.66	84	1905
ex14.1.8	T	T	T	T	F	F	0.08	0.64	12	72
ex14.2.1	T	F	T	F	T	F	—	—	—	—
ex14.2.2	T	F	T	T	F	F	0.54	0.54	88	88
ex14.2.4	T	F	T	T	F	F	191.56	191.63	16725	16733
ex14.2.5	T	F	T	T	F	F	0.58	0.58	85	85
ex14.2.6	T	F	T	F	T	F	—	—	—	—
ex14.2.8	T	F	T	F	T	F	—	—	—	—
ex14.2.9	T	F	T	F	T	F	—	—	—	—
ex2.1.4	T	F	T	T	F	F	0.67	2.46	55	146
ex5.2.2.1	T	F	F	F	F	F	—	—	—	—
ex5.2.2.2	T	F	F	F	F	F	—	—	—	—
ex5.2.4	T	T	T	T	F	F	251.73	252.31	34715	34631
ex7.3.3	T	T	T	T	F	F	8.4	10.9	1265	1807
ex7.3.5	T	F	F	F	F	F	—	—	—	—
ex8.1.7	T	T	T	T	F	F	16.05	21	2579	3285
ex9.1.2	T	F	F	F	F	F	—	—	—	—
ex9.1.3	F	F	T	T	F	F	0.53	0.52	1	1
ex9.1.4	T	F	F	F	F	F	—	—	—	—
ex9.1.6	F	F	F	F	F	F	—	—	—	—
ex9.1.7	F	F	F	F	F	F	—	—	—	—
ex9.1.9	T	F	F	F	F	F	—	—	—	—
ex9.2.4	T	F	T	T	F	F	0.43	0.43	32	32
ex9.2.8	T	F	T	T	F	F	0.05	5710.39	3	743702
harker	T	F	F	F	F	F	—	—	—	—
haverly	F	F	F	F	F	F	—	—	—	—
house	T	F	F	F	F	F	—	—	—	—
immun	T	F	F	F	F	F	—	—	—	—
qp5	T	T	F	F	F	F	—	—	—	—
sambal	T	F	F	F	F	F	—	—	—	—
sample	T	T	T	T	F	F	1.17	1.19	299	299
Totals	27	6	17	13	4	0	472.77	6208.25	55943	802786
Ratios								8%		7%

software. In this section, numerical results illustrate the consequences of this effectiveness to the efficiency within an overall branch and bound algorithm.

In particular, we compared Algorithm 1 and verification of a critical point with an interval Newton method applied to the Fritz John system within GlobSol. The overall tolerance ϵ_d, representing the smallest scaled-diameter box produced during the branch and bound algorithm, was set to $\epsilon_d = 10^{-5}$. The experiments were done on a Dell Dimension E-310 with a 3GHz Pentium 4 processor and 2GB of memory, Ubuntu 12.04 with the GNU compiler suite (gfortran, C, and C++) version 4.6 and optimization level 3. The results appear in Table 2.

The columns of Table 2 are as follows:

– "P?" is true if an only if Algorithm 1 was successful at least one time during the branch and bound run when Algorithm 1 was used. Note that this can happen more often than in Table 1, since the local optimizer is run more than once during the branch and bound process.

- "FJ?" is true if an only if verification with the Fritz John equations was successful at least one time during the branch and bound run when the Fritz John equations were used.
- "P F?" is "T" if an only if the branch and bound run using Algorithm 1 finished successfully.
- "FJ F?" is "T" if an only if the branch and bound run using verification a solution to the Fritz John equations finished successfully.
- "P+" is "T" if and only if the Algorithm 1 version finished but the Fritz John version did not.
- "FJ+" is "T" if and only if the Fritz John version finished but the Algorithm 1 version did not.
- "P T" gives the processor time in seconds for the Algorithm 1 version, in those cases where both versions finished.
- "FJ T" gives the processor time in seconds for the Algorithm 1 version, in those cases where both versions finished.
- "P boxes" gives the number of boxes traversed in the Algorithm 1 version, in those cases where both versions finished.
- "FJ boxes" gives the number of boxes traversed in the Fritz John version, in those cases where both versions finished.

We make the following observations.

- Algorithm 1 did not verify feasibility for any more problems than those initially identified with a single approximate feasible point, but verification with the Fritz John equations succeeded for one more problem, `ex14.1.8`.
- The branch and bound process utilizing Algorithm 1 succeeded in four problems in which the branch and bound process utilizing Fritz John system verification did not, but the Fritz John version did not succeed for any problems for which the Algorithm 1 did not.
- The branch and bound process finished in only one case, `ex9.1.3` in which Algorithm 1 failed, although the branch and bound process finished in 8 cases in which existence of a critical point could not be verified.
- GlobSol failed to complete for reasons other than availability of a good upper bound on the global optimum in 10 cases, but the results clearly show availability of a good upper bound is important.
- Comparing execution times when the branch and bound process completed using either version of feasibility verification, execution times were comparable when both verification schemes succeeded, as well as in several cases when Algorithm 1 but Fritz John system verification failed. However, execution times were much smaller when Algorithm 1 was used for `ex14.1.1` and `ex9.2.8`; thus, overall, the execution time when using Algorithm 1 was only 8% of that when using the Fritz John system, and the total number of boxes considered was only 7% of that when using the Fritz John system.

Upon closer examination of the cases in which GlobSol failed to complete even though a sharp rigorously verified upper bound on the global optimum was obtained, the reasons fell into the following categories.

Existence of curves or hypersurfaces of global optimizers: This happened in `ex9.1.2`, `ex9.1.4`, and `house`. GlobSol obtained a sharp enclosure for the global optimum in these cases, and the unfathomed boxes were clustered around the optimizing

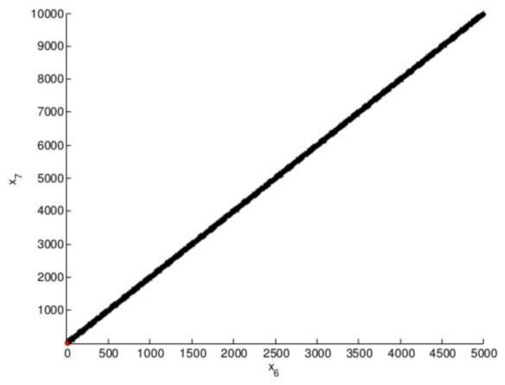

Fig. 1 x_6 vs x_7 cross-section of the globally optimizing set for `ex9.1.2`

sets; for example, see Figure 1. Also, in these cases, the total volume of the as-yet-unfathomed boxes was small.

Other reasons: This occurred for `ex5.2.2.1`, `ex5.2.2.2`, `harker`, `ex7.3.5`, `house`, `immun`, and `qp5` We will more thoroughly investigate the reasons for these failures in the future. In the case of `qp5`, apparently the processing time per box is simply too large, due to the number of variables and constraints; GlobSol's original design is for small problems and for study of algorithm details, rather than efficiency on larger problems. In other cases, it may be due to the inability to obtain sharp lower bounds on subregions, because the linear relaxations used in GlobSol are not the best possible.

4.3 On the Accuracy of the Optimum Value

Our "perturb to feasibility" Algorithm 1 requires the problem have only equality constraints. For these experiments, we have relaxed each equality constraint slightly (by an absolute number 10^{-4}), and, because some of the problems in the COCONUT Lib 1 test set [10] have already been reformulated, with equality constraints replaced by pairs of inequality constraints, we also relaxed inequality constraints by 10^{-4}. Thus, we are getting rigorous bounds to an approximate problem. The question arises concerning whether or not using an approximate algorithm, that is, simply using approximately feasible points returned by the floating point optimizer on the original non-perturbed problem will give reasonable bounds to the actual global optimum value, and which bounds are better.

Here, we empirically study use of an approximate optimizer versus our rigorously verified feasible points. To do this within an otherwise similar environment, we introduced branches in GlobSol at points where the approximate point from the approximate optimizer are processed to determine feasibility, controlling the flow with an additional configuration variable. Not trusting the return code from IPOPT to indicate an approximate solution, we accepted the point as approximately feasible provided the constraints were within a tolerance of 10^{-5} of feasible.

Table 3 Comparison of use of exact and non-rigorous approximate feasible points.

Prob.	N. F?	P. F?	Reported Opt.	UB - R NP	UB - R P	UB-LB NP	UB-LB P
ex14.1.1	T	T	0	-1.00×10^{-8}	1.00×10^{-9}	$1.00 \times 10^{+4}$	5.60×10^{-3}
ex14.1.8	T	T	0	-1.00×10^{-8}	1.00×10^{-9}	1.81×10^{-3}	1.21×10^{-2}
ex14.2.1	F	T	0	$3.48 \times 10^{+3}$	1.00×10^{-9}	$3.47 \times 10^{+3}$	1.00×10^{-5}
ex14.2.2	T	T	0	$6.53 \times 10^{+3}$	1.00×10^{-9}	$1.53 \times 10^{+3}$	1.00×10^{-5}
ex14.2.4	T	T	0	$3.48 \times 10^{+3}$	1.00×10^{-9}	$9.75 \times 10^{+2}$	1.00×10^{-5}
ex14.2.5	T	T	0	$6.53 \times 10^{+3}$	1.00×10^{-9}	$1.53 \times 10^{+3}$	1.00×10^{-5}
ex14.2.6	F	T	0	$3.48 \times 10^{+3}$	1.00×10^{-9}	$3.48 \times 10^{+3}$	1.00×10^{-5}
ex14.2.8	T	T	0	$6.53 \times 10^{+3}$	1.00×10^{-9}	$1.53 \times 10^{+3}$	1.00×10^{-5}
ex14.2.9	T	T	0	$6.53 \times 10^{+3}$	1.00×10^{-9}	$1.53 \times 10^{+3}$	1.00×10^{-5}
ex2.1.4	T	T	$-1.10 \times 10^{+1}$	0	0	$2.16 \times 10^{+0}$	$2.16 \times 10^{+0}$
ex5.2.2.1	T	F	$-4.00 \times 10^{+2}$	0	0	6.00×10^{-1}	$1.99 \times 10^{+2}$
ex5.2.2.2	T	F	$-6.00 \times 10^{+2}$	0	0	$2.40 \times 10^{+0}$	$4.04 \times 10^{+2}$
ex5.2.4	T	T	$-4.50 \times 10^{+2}$	0	-2.00×10^{-1}	$2.60 \times 10^{+0}$	1.00×10^{-1}
ex7.3.3	T	T	8.18×10^{-1}	0	0	$1.32 \times 10^{+0}$	5.75×10^{-2}
ex7.3.5	F	F	$1.20 \times 10^{+0}$	3.40×10^{-3}	$-1.20 \times 10^{+0}$	$1.53 \times 10^{+0}$	3.16×10^{-2}
ex8.1.7	F	T	2.93×10^{-2}	1.00×10^{-5}	-3.00×10^{-5}	0	1.53×10^{-3}
ex9.1.2	F	F	$-1.60 \times 10^{+1}$	0	0	0	0
ex9.1.3	T	T	$-5.80 \times 10^{+1}$	$4.01 \times 10^{+4}$	$4.01 \times 10^{+4}$	$6.00 \times 10^{+5}$	$6.00 \times 10^{+5}$
ex9.1.4	F	T	$-3.70 \times 10^{+1}$	0	0	0	$3.99 \times 10^{+4}$
ex9.1.6	F	F	$-5.20 \times 10^{+1}$	$3.00 \times 10^{+0}$	$5.20 \times 10^{+1}$	0	$4.90 \times 10^{+1}$
ex9.1.7	F	F	$-5.00 \times 10^{+1}$	0	$8.01 \times 10^{+4}$	5.80×10^{-1}	$8.00 \times 10^{+4}$
ex9.1.9	T	F	$2.00 \times 10^{+0}$	0	0	3.20×10^{-2}	0
ex9.2.4	T	T	5.00×10^{-1}	0	$3.48 \times 10^{+0}$	1.00×10^{-3}	1.25×10^{-1}
ex9.2.8	—	T	$1.50 \times 10^{+0}$	—	0	—	9.90×10^{-2}
harker	F	F	$-9.87 \times 10^{+2}$	1.35×10^{-2}	1.35×10^{-2}	$3.90 \times 10^{+3}$	$5.22 \times 10^{+3}$
haverly	T	F	$-4.00 \times 10^{+2}$	0	$1.04 \times 10^{+4}$	$3.93 \times 10^{+1}$	$1.08 \times 10^{+4}$
house	T	F	$-4.50 \times 10^{+3}$	$2.45 \times 10^{+4}$	0	$2.28 \times 10^{+4}$	0
immun	F	F	0	—	3.64×10^{-10}	5.58×10^{12}	3.64×10^{-10}
qp5	F	F	4.32×10^{-1}	0	-1.90×10^{-3}	4.32×10^{-1}	4.30×10^{-1}
sambal	F	F	$3.97 \times 10^{+0}$	-2.00×10^{-4}	-2.00×10^{-4}	9.86×10^{-1}	$1.17 \times 10^{+0}$
sample	T	T	$7.27 \times 10^{+2}$	$2.36 \times 10^{+2}$	$-4.84 \times 10^{+0}$	$7.40 \times 10^{+0}$	$2.70 \times 10^{+0}$

Within this context, we reran the 31 selected test problems in Table 2, comparing the best upper bound on the global optimum using the approximate optimizing point, the best lower bound obtained using the approximate optimizing point, the best upper and lower bounds obtained from the exactly feasible point of the nearby problem, and the global optimum reported in [10].

The results appear in Table 3. The columns of Table 3 are as follows:

– N. F is “T” only if the algorithm using the approximate optimizer finished within 7200 CPU seconds.
– P. F is “T” only if the algorithm using proven feasible points of the perturbed problem finished within 7200 CPU seconds.
– Reported Opt. Is the optimum value reported on the web site [10].

- UB - R NP is the difference between the best upper bound and the reported optimum when the only approximately feasible points are are used to obtain the approximate upper bound.
- UB - R N is the difference between the best upper bound of the relaxed problem and the reported optimum of the original problem when the exactly feasible points of the relaxed problem are used to obtain the upper bound.
- UB-LB NP Is the difference between the upper bound and the best lower bound on the optimum, when the algorithm with only approximate feasible points was applied to the original non-relaxed problem.
- UB-LB P Is the difference between the upper bound and the best lower bound on the optimum, when the algorithm with exactly feasible points was applied to the relaxed problem.

In Table 3:

1. The approximate feasible point version failed to complete in 7200 seconds on 12 problems, while the exact feasible point version failed to complete on 13 problems. However, the correlation between completions is only about 0.4.
2. In the non-relaxed version of problem ex9.2.8, IPOPT apparently diverged to a point with coordinates on the order of 10^{14}, but nonetheless reported successful completion. No similar difficulties were observed with the relaxed problem.
3. The difference between the best upper bound and reported global optimum was relatively small for 25 of the problems using an exact feasible point of the relaxed problem but only for 19 of the problems when the approximate feasible point of the non-perturbed problem was used. The correlation between when the difference was small between the two data sets was about -0.22.
4. The difference between the upper bound and reported optimum was negative for the algorithm using exactly feasible points of the relaxation for problems ex5.2.4, ex7.3.5, ex8.1.8, qp3, sambal, and sample. For all of these except ex7.3.5, the difference from the reported optimum is sufficiently small to be attributed to either the accuracy to which the reported solution was given or the perturbation resulting from relaxing the problem. In ex7.3.5, there is evidence of some kind of ill-conditioning, as evidenced in difficulties the solver within GlobSol for linear relaxations (C-LP) encountered during its solution process; it is possible that small problem perturbations could lead to large changes in the global optimum.
5. The difference between the upper bound and reported optimum was negative for the algorithm using the approximately feasible points of the original problem for ex14.1.1, ex14.1.8, ex9.2.4, and sambal. Except for ex9.2.8, already discussed, the differences are sufficiently small to attribute them to using an only approximate feasible point to compute upper bounds.

We infer from Table 3 that using verified feasible points of the perturbed problems is somewhat more robust than using approximate feasible points of the original problem; the perturbations even seem to aid our approximate solver IPOPT. Furthermore, the perturbations do not greatly affect the accuracy of the optimum values in the problems in this test set. However, this varies from problem to problem.

We have not attempted to compare the sets of optimizing points returned by the two schemes.

4.4 Comparison with the Feasibility Verification Scheme in [5]

Finally, in Table 4, we compare the scheme from [5] (in which we use an interval Newton method in a subspace) to the scheme in Algorithm 1, as we compare use of the Fritz John conditions to Algorithm 1 in Table 1. In Table 4, the columns labelled "Prob." and "n" are the problem name and number of variables, as in Table 1. In the other columns:

- "N_a" is the number of approximately active constraints at the optimizing point.
- "applic." is "T" if the number of approximate binding constraints is less than or equal to the number n of variables. (This is necessary for the scheme from [5] to make sense.)
- "[5] OK?" is "T" if the scheme from [5] succeeded in verifying feasibility.

Table 4 Comparison of the scheme from [5] and Algorithm 1

Prob.	n	N_a	applic.?	[5] OK?	A. 1 OK?
ex14.1.1	3	0	T	F	T
ex14.1.8	3	0	T	T	T
ex14.2.1	5	1	T	T	T
ex14.2.2	4	1	T	T	T
ex14.2.4	5	1	T	T	T
ex14.2.5	4	1	T	T	T
ex14.2.6	5	1	T	T	T
ex14.2.8	4	1	T	T	T
ex14.2.9	4	1	T	T	T
ex2.1.4	6	0	T	F	T
ex5.2.2.1	9	4	T	F	T
ex5.2.2.2	9	4	T	F	T
ex5.2.4	7	1	T	F	T
ex7.3.3	5	2	T	T	T
ex7.3.5	13	1	T	F	T
ex8.1.7	5	1	T	F	T
ex9.1.2	10	0	T	F	T
ex9.1.3	29	0	T	F	F
ex9.1.4	10	0	T	F	T
ex9.1.6	20	0	T	F	F
ex9.1.7	23	0	T	F	F
ex9.1.9	17	0	T	F	T
ex9.2.4	8	0	T	F	T
ex9.2.8	6	7	F	F	T
harker	20	7	T	F	T
haverly	12	7	T	F	F
house	8	0	T	F	T
immun	21	0	T	F	T
qp5	108	0	T	F	T
sambal	17	0	T	F	T
sample	4	0	T	T	T
Totals			30	10	27

We see that the number of approximately active constraints exceeded the number of variables in only one problem (ex9.2.8), but the scheme from [5] only succeeded in 10 of the problems. (In contrast, the scheme from Algorithm 1 succeeded in 27 of the

28 problems in which the approximate solver returned a high-quality approximately feasible point.) The scheme from [5] requires use of an interval Newton method with a Jacobian matrix consisting of some of the columns of the Jacobian matrix of the approximately active constraints. This scheme is bound to fail if that Jacobian matrix is singular or even moderately ill-conditioned; we have already noted that, in some of these test problems, the Jacobian matrix at a *must* be singular, since some of the equality constraints are expressed as pairs of inequality constraints in the publicly posted versions of these problems.

Inferences

We have observed that relaxing the equality and inequality constraints does not lead to significantly different bounds on the global optimum, vis a vis using only approximate optimizing points. Furthermore, the relaxation scheme (i.e. perturbing the problem so the feasible set has an interior) has not only made it possible to state mathematically rigorous results[1] for a near by problem but also apparently makes it easier for the floating point iterative method (IPOPT in these experiments) to converge to the global optimum. This was definitely true for ex14.2.1 through ex14.2.9.

Caveats

In this set of experiments, we have merely replaced the mathematically rigorous computation of an upper bound based on an interval evaluation of the objective at a point that has been proven to be feasible with use of an approximately feasible point instead of one proven to be feasible, with minimal changes to other parts of the algorithm. One issue in using an approximately feasible point include use of heuristics to determine if the point returned by the solver is actually near a feasible point. (This was necessary for ex9.2.8 and immun, in our experiments.) Furthermore, the remainder of GlobSol involves other checks, such as formulating linear relaxations rigorously and using interval Newton methods, that can be omitted if only approximate solutions are desired, and much computation can be avoided if only an approximation to the global optimum and one (but not all) optimizing points are desired. Finally, many preprocessing steps, such as identification of linear, quadratic, or convex problems, can lead to fast computation of approximate solutions. The performance of non-rigorous software that incorporates all of these will be significantly better than GlobSol. Examples of this include BARON [13], the techniques of Messine and Ninin [12], [1], or more specialized codes such as is described in [3] and related work.

5 Relationship to Feasibility Restoration

Algorithm 1 can be viewed as a feasibility restoration process. As such, one wonders about its relationship with the feasibility restoration phase in penalty-free nonlinear programming methods, such as those initially described in [2] and in use in the IPOPT solver employed to obtain an approximately feasible approximate optimal point in the experiments in this work [16, 17]. The "restoration-" like technique in Algorithm 1 involves linearizing the constraints, but the conditions under which it is used are different

[1] assuming, of course, there are no programming errors

than those of the restoration phase of approximate nonlinear program solvers. In particular, the steps taken in such nonlinear programming solvers are large in relation to the validity of the linear or quadratic models used, while Algorithm 1 does a tiny correction step from a solver-provided point thought to be near an exact locally optimizing point. Furthermore, requirements in an approximate nonlinear programming solver are merely that an approximately feasible point be obtained, and only progress towards this goal is required in a single restoration step, while, in Algorithm 1, exact feasibility is required, and a single small step should be sufficient to achieve this. Moreover, the correction step in Algorithm 1 is applied to a perturbed problem whose feasible set is known to have a non-empty interior, whereas some merit-function-free nonlinear program solvers are applied to problems with only equality constraints. Due to these conditions, the step length in Algorithm 1 can be chosen rationally, based on the magnitudes of gradients, whereas the restoration phase in nonlinear programming solvers generally has various heuristically chosen parameters. We do not know whether or not our paradigm of identifying feasible and infeasible inequality constraints and using generalized inverses in the subspace orthogonal to the space of feasible but approximately active equality constraints has been used in the feasibility restoration phase of local algorithms, but it is possible to do so, with modifications.

6 Summary

We have proposed a general technique for perturbing approximate local optimizing points to points that can be rigorously verified to be feasible. Depending on working with a relaxation of equality constraints to inequality constraints with a rational choice of relaxation parameter, this technique is designed be robust with respect to number of approximately active constraints and linear dependence of the gradients of the approximately active constraints. We had previously postulated that inability to prove approximate optimizing points feasible was one reason for the performance difference between mathematically rigorous branch and bound algorithms and ones that employ only approximate feasibility. Experimental results with the new technique show it is superior, and also show its use makes a significant difference in the branch and bound algorithm. Remaining performance difficulties in the branch and bound algorithm were found to be due to the existence of continua of solution sets (since the particular branch and bound process considered finds enclosures to all global optimizing points, rather than an enclosure for just one optimizing point and the global optimum) in various cases, and to other issues in the remaining cases.

Acknowledgements

I wish to thank the referees for their numerous comments that substantially improved the content of this work.

References

1. C. S. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier. A global optimization method, αBB, for general twice-differentiable constrained NLPs I. Theoretical advances. *Computers and Chemical Engineering*, 22(9):1137–1158, 1998.

2. Roger Fletcher and Sven Leyffer. Nonlinear programming without a penalty function. *Math. Prog.*, 91(2):239–269, 2002.
3. Christodoulos Floudas. Deterministic global optimization: Advances and challenges, June 2009. Plenary Lecture, First World Congress on Global Optimization in Engineering and Sciences, WCGO-2009.
4. Ralph Baker Kearfott. *Rigorous Global Search: Continuous Problems.* Number 13 in Nonconvex Optimization and its Applications. Kluwer Academic Publishers, Dordrecht, Netherlands, 1996.
5. Ralph Baker Kearfott. On proving existence of feasible points in equality constrained optimization problems. *Math. Prog.*, 83(1):89–100, 1998.
6. Ralph Baker Kearfott. GlobSol user guide. *Optimization Methods and Software*, 24(4-5):687–708, August 2009.
7. Ralph Baker Kearfott. Interval computations, rigour and non-rigour in deterministic continuous global optimization. *Optimization Methods and Software*, 26(2):259–279, April 2011.
8. Ralph Baker Kearfott, Jessie M. Castille, and Gaurav Tyagi. Assessment of a non-adaptive deterministic global optimization algorithm for problems with low-dimensional non-convex subspaces, 2011. accepted for publication in *Optimization Methods and Software*, `http://interval.louisiana.edu/preprints /2011-optimization-by-discretization.pdf`.
9. Ralph Baker Kearfott, Sowmya Muniswamy, Yi Wang, Xinyu Li, and Qian Wang. On smooth reformulations and direct non-smooth computations in global optimization for minimax problems, 2012. Accepted for publication in the *Journal of Global Optimization*.
10. Arnold Neumaier. COCONUT Web page, 2001-2003. `http://www.mat.univie.ac.at/~neum/glopt/coconut`.
11. Jordan Ninin, Frédéric Messine, and Pierre Hansen. A reliable affine relaxation method for global optimization, 2010. Rapport de recherche, RT-APO-10-05, IRIT, March, `ftp://ftp.irit.fr/IRIT/APO/RT-APO-10-05.pdf`.
12. Jordan Ninin and Frdric Messine. A metaheuristic methodology based on the limitation of the memory of interval branch and bound algorithms. *J. Global Optim.*, 50(4):629–644, 2011.
13. Nikolaos V. Sahinidis. BARON: A general purpose global optimization software package. *Journal of Global Optimization*, 8(2):201–205, 1996.
14. O. Shcherbina, A. Neumaier, D. Sam-Haroud, X.-H. Vu, and T.-V. Nguyen. Benchmarking global optimization and constraint satisfaction codes. In C. Bliek, C. Jermann, and A. Neumaier, editors, *COCOS*, volume 2861 of *Lecture Notes in Computer Science*, pages 211–222. Springer-Verlag, 2003.
15. Andreas Wächter. Homepage of IPOPT, 2002. `https://projects.coin-or.org/Ipopt`.
16. Andreas Wächter and Lorenz T. Biegler. Line search filter methods for nonlinear programming: Local convergence. *SIAM Journal on Optimization*, 16(1):32–48, 2005.
17. Andreas Wächter and Lorenz T. Biegler. Line search filter methods for nonlinear programming: Motivation and global convergence. *SIAM Journal on Optimization*, 16(1):1–31, 2005.