Discussion and Empirical Comparisons of Linear Relaxations and Alternate Techniques in Validated Deterministic Global Optimization

R. Baker Kearfott*

April 27, 2004

Abstract

Both theory and implementations in deterministic global optimization have advanced significantly in the past decade. Two schools have developed: the first employs various bounding techniques without validation, while the second employs different techniques, in a way that always rigorously takes account of roundoff error (i.e. with validation). However, convex relaxations, until very recently used without validation, can be implemented efficiently in a validated context. Here, we empirically compare a validated implementation of a variant of convex relaxations (linear relaxations applied to each intermediate operation) with traditional techniques from validated global optimization (interval constraint propagation and interval Newton methods.) Experimental results show that linear relaxations are of significant value in validated global optimization, although further exploration will probably lead to more effective inclusion of the technology.

Keywords: nonconvex optimization, global optimization, linear relaxations, GlobSol, constraint propagation
AMS Subject Classifications: 90C30, 65K05, 90C26, 65G20

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

1 Introduction

1.1 The General Global Optimization Problem

Our general global optimization problem can be stated as

$$\boxed{\begin{array}{l}\text{minimize } \varphi(x)\\ \text{subject to } c_i(x) = 0,\ i = 1,\ldots,m_1,\\ \qquad\qquad g_i(x) \le 0,\ i = 1,\ldots,m_2,\\ \text{where } \varphi : \boldsymbol{x} \to \mathbb{R} \text{ and } c_i, g_i : \boldsymbol{x} \to \mathbb{R}, \text{ and where } \boldsymbol{x} \subset \mathbb{R}^n \text{ is}\\ \text{the hyperrectangle (box) defined by}\\ \qquad\qquad \underline{x}_i \le x_i \le \overline{x}_i,\ 1 \le i \le n,\\ \text{where the } \underline{x}_i \text{ and } \overline{x}_i \text{ are constant bounds.}\end{array}} \tag{1}$$

1.2 Non-Validated Versus Validated Algorithms

Spurred perhaps by the seminal proceedings [2, 3], numerous methods for attacking problem 1 have been developed over the past thirty years. These methods can be classified into heuristic methods, such as genetic algorithms or simulated annealing, in which solutions are found only with a given probability, and deterministic algorithms, in which a systematic search of the domain $\boldsymbol{x}$ is made[1]. Within the class of deterministic algorithms, development has proceeded along two separate lines: (i) algorithms that would certainly give all solutions to problem (1) if exact arithmetic were used, and (ii) algorithms that take account of roundoff error to certainly give all solutions to problem (1), even with finite-precision arithmetic. We refer to the latter two classes as non-validated algorithms and validated algorithms, respectively.

Development in non-validated algorithms in particular has proceeded at an accelerated pace over the past decade, with both an explosion in the literature and increasing numbers of commercial implementations. These implementations have been driven primarily by engineering (chemical engineering in particular) and operations research applications, such as described in the books [4, 19]. To date, the techniques and implementations in non-validated algorithms, such as those described in [4] and [19], have been more successful in applications than techniques and implementations in validated algorithms. However, the perhaps most important element of the non-validated techniques, namely, linear relaxations, can be modified to work in a validated context. We have done this. This raises the question, "In what contexts are linear relaxations necessary, and how do they compare to techniques traditionally used in validated algorithms?" That is the subject of this paper.

[1] Deterministic algorithms can further be classified into complete and incomplete algorithms, as in [18], where the incomplete algorithms seek only one solution of problem (1), while the complete algorithms seek all solutions. In this work, we consider only complete algorithms.

1.3 Outline

In §2, we briefly review the global search algorithm common to all deterministic global optimization routines, and we briefly describe our implementation environment (GlobSol). In §3, we review linear relaxations, used effectively in non-validated deterministic algorithms. We also review interval Newton methods and interval constraint propagation, the primary techniques used to date in validated algorithms, in §3. We explain our implementation of validated linear relaxations and how we fit our implementation into our computational environment in §4; this explanation is brief, and points, where appropriate, to other work we have produced on the subject. We explain what we seek to determine in our numerical experiments and how we will conduct our experiments in §5, while we present the actual results in §6. Conclusions, as well as disclaimers and possible future work, appear in §7.

2 Our Overall Algorithm and Computational Environment

2.1 Structure and Techniques Common to All Algorithms

Deterministic global optimization algorithms in general start by initializing a list of boxes $\mathcal{L}$ by inserting the region $\boldsymbol{x}$ defined by the bound constraints into $\mathcal{L}$. The algorithms then proceed to subdivide and process the boxes in $\mathcal{L}$ as follows:

1. Remove a box $\boldsymbol{x}$ from the list $\mathcal{L}$.
2. Process $\boldsymbol{x}$ to do one of the following:
   (a) Reject $\boldsymbol{x}$ as not containing a solution.
   (b) Replace the coordinate bounds for $\boldsymbol{x}$ by narrower bounds in which any global optima in $\boldsymbol{x}$ must lie.
   (c) Determine that either the coordinate widths of $\boldsymbol{x}$ are within a specified tolerance, or else that $\boldsymbol{x}$ contains a unique solution; if so, then store $\boldsymbol{x}$ on a list $\mathcal{C}$ of "answer" boxes.
3. If $\boldsymbol{x}$ was neither rejected nor placed onto the list $\mathcal{C}$ of answer boxes in step 2, then subdivide $\boldsymbol{x}$, placing the resulting sub-boxes on $\mathcal{L}$.

One common way of rejecting boxes in step 2 is the *lower bound test*[2].

Definition 1 *The lower bound test consists of:*

1. *maintaining an upper bound $\overline{\varphi}$ on the global optimum (such as by evaluating the objective φ at feasible points), and*

[2]In the interval analysis literature, this is sometimes called the *midpoint test*, since the upper bound $\overline{\varphi}$ can be obtained by evaluating the objective at the midpoint of the boxes $\boldsymbol{x}$.

2. *computing, for each* $\boldsymbol{x}$*, a lower bound* $\underline{\varphi}(\boldsymbol{x})$ *on the objective* φ *over the intersection of* $\boldsymbol{x}$ *with the feasible set of problem (1).*

In the past, non-verified and verified algorithms have done the lower bound test in a different way. We will explain this difference in §3 below.

A second way of both narrowing the bounds and possibly rejecting boxes is by constraint propagation, as described throughout the literature on logic programming; a good example that also describes a complete implementation is [20]. One way to view (and implement) constraint propagation is to solve a constraint for a variable x_i. One then substitutes the bounds on the other variables into the constraint and computes the value with interval arithmetic. If any resulting interval $\tilde{\boldsymbol{x}}_i$ for the variable x_i is disjoint from the original interval $\boldsymbol{x}_i$, then $\boldsymbol{x}$ is rejected; if those bounds are not disjoint, but $\tilde{\boldsymbol{x}}_i \cap \boldsymbol{x}_i$ is narrower than $\boldsymbol{x}_i$, then these new bounds are "propagated" in the other constraints that contain x_i, to possibly compute narrower bounds on other variables; the entire process is continued until it becomes stationary (or approximately stationary). We have worked out a simple example of this process in detail in [6]. There are numerous implementations of this process, of varying effectiveness.

2.2 GlobSol

Our test bed for comparing various schemes is GlobSol. Considerations that went into GlobSol's design, as well as description of a precursor to GlobSol, appear in [7], while we succinctly review GlobSol in [9]. GlobSol's overall search algorithm follows the general scheme above. For reference here, we repeat the details of this scheme that we gave in [9]:

Algorithm 1 *(GlobSol's global search algorithm)*
INPUT: A list $\mathcal{L}$ of boxes $\boldsymbol{x}$ to be searched.
OUTPUT: A list $\mathcal{U}$ of small boxes and a list $\mathcal{C}$ of boxes verified to contain feasible points, such that any global mimimizer must lie in a box in $\mathcal{U}$ or $\mathcal{C}$.
DO WHILE *($\mathcal{L}$ is non-empty)*

1. *Remove a box* $\boldsymbol{x}$ *from the list* $\mathcal{L}$*.*
2. IF $\boldsymbol{x}$ *is sufficiently small* THEN
   (a) *Analyze* $\boldsymbol{x}$ *to validate feasible points, possibly widening the coordinate widths (ϵ-inflation) to a wider box within which uniqueness of a critical point can be proven.*
   (b) *Place* $\boldsymbol{x}$ *on either* $\mathcal{U}$ *or* $\mathcal{C}$*.*
   (c) *Apply the complementation process of [7, p. 154].*
   (d) CYCLE

   END IF
3. *(Constraint Propagation)*
   (a) *Use constraint propagation to possibly narrow the coordinate widths of the box* $\boldsymbol{x}$*.*

   (b) IF *constraint propagation has shown that* $\boldsymbol{x}$ *cannot contain solutions* THEN CYCLE

4. *(Interval Newton)*

   (a) Perform an interval Newton method to possibly narrow the coordinate widths of the box $\boldsymbol{x}$.

   (b) IF *the interval Newton method has shown that* $\boldsymbol{x}$ *cannot contain solutions* THEN CYCLE

5. IF *the coordinate widths of* $\boldsymbol{x}$ *are now sufficiently narrow* THEN

   (a) Analyze $\boldsymbol{x}$ *to validate feasible points, possibly widening the coordinate widths (*ϵ*-inflation) to a wider box within which uniqueness of a critical point can be proven.*

   (b) Place $\boldsymbol{x}$ *on either* $\mathcal{U}$ *or* $\mathcal{C}$.

   (c) Apply the complementation process of [7, p. 154].

   (d) CYCLE

6. *(Subdivide)*

   (a) Choose a coordinate index k *to bisect (i.e. to replace* $[\underline{x}_k, \overline{x}_k]$ *by* $[\underline{x}_k, (\underline{x}_k + \overline{x}_k)/2]$ *and* $[(\underline{x}_k + \overline{x}_k)/2, \overline{x}_k]$*).*

   (b) Bisect $\boldsymbol{x}$ *along its* k*-th coordinate, forming two new boxes; place these boxes on the list* $\mathcal{L}$.

   (c) CYCLE

END DO

End Algorithm 1

Thus, in GlobSol, in the narrowing and rejection process, constraint propagation is first applied, then an interval Newton method. Within the interval Newton method, traditional interval techniques, such as interval evaluation of the constraints to possibly reject boxes by proving infeasibility, or such as checking the lower bound on the objective, are applied. For further details, see [7], [9], or the GlobSol source code itself.

3 Linear Relaxations Versus Constraint Propagation and Interval Newton Methods

Constraint propagation alone (in the sense of "substitution-iteration" that we describe in [6]) cannot effectively solve all problems. One way of viewing why is to imagine constraint propagation to be a nonlinear version of Gauss–Seidel iteration, which only converges if the off-diagonal elements (i.e. the coefficients of the variables on the right side of the iteration equation) are sufficiently small. Otherwise stated, constraint propagation alone (without subdivision and other schemes) converges only if the variables are only weakly coupled in the constraints.

Similarly, interval Newton methods can fail over larger boxes, due to overestimation and other phenomena. In particular, for constrained optimization problems, GlobSol uses the Fritz–John system for the interval Newton method. We have found interval extensions of the derivative matrix for this system to often contain singular matrices for all but very small boxes $\boldsymbol{x}$.

Finally, the lower bound test (Definition 1), incorporated in GlobSol in both the constraint propagation and interval Newton steps, is weak in GlobSol, and is not effective at all for certain problems. In particular, in the past, GlobSol has obtained the lower bound $\underline{\varphi}(\boldsymbol{x})$ exclusively from an interval evaluation of φ. We illustrated this problem in [11, p. 54], and give a specific instance of that example in §5 below.

An alternative, prevalent in non-validated algorithms, is to form convex, or, more specifically, linear *relaxations* of problem (1). To obtain a linear relaxation to problem (1) over a box $\boldsymbol{x}$, we replace the objective φ by one or more linear functions that are less than or equal to φ over $\boldsymbol{x}$, and we replace each g_i corresponding to an inequality constraint by one or more linear functions that are less than or equal to g_i; we replace each equality constraint $c_i(x) = 0$ by a pair of inequality constraints $c_i(x) \leq 0$ and $-c_i(x) \leq 0$, then similarly underestimate each of these inequality constraints by linear functions. The result, termed a *linear relaxation*, is a linear program (LP) whose optimum gives a lower bound on φ over the feasible points in $\boldsymbol{x}$. Additionally, the optimizer for the LP is, in some sense, an approximate solution to the original global optimization problem (1), and can be used effectively in the algorithm. Linear relaxations perhaps began with McCormick [13, 14], and have since become popular within the global optimization community. Extensive exposition of how to obtain linear relaxations appears in [19], while techniques for more general convex relaxations appear in [4]. We present a brief introduction followed by an analysis of our view of the process of constructing linear relaxations in [10].

A process in GlobSol (and in other deterministic search algorithms) not explicitly described in Algorithm 1 is initial use of a traditional local optimizer to obtain hypothesized approximate global optima and corresponding approximate optimizers for problem (1). These approximate optimizers are then used in two ways: (i) to obtain an upper bound on the global optimum (by evaluation of φ), and (ii) in a *box complementation* process, as explained in [7, §4.3.1, p. 154]. (In this box complementation process, a small box is constructed about the local optimizer, local optimality conditions are proven to hold at some point within the box, the box is placed on the list of answer boxes, and the box is then removed from the search region.) GlobSol's local optimizer for constrained problems[3] has consisted of a simple generalized Newton method, in which we use a generalized inverse of the Jacobi matrix to project onto the feasible set[4]. GlobSol could perhaps solve certain problems more efficiently with a stronger local optimizer.

[3]For unconstrained problems, GlobSol has used the MINPACK-1 [16] routine `HYBRJ`.

[4]This simple local optimizer is to allow GlobSol to be self-contained and free of license restrictions. We comment more on this in §7.

4 On Our Implementation of Linear Relaxations

The linear relaxations can be formed directly from the constraints in problem (1), or problem (1) can be preprocessed in various ways. As we explain in [10], we have elected to form an `expanded system` by assigning a variable to each intermediate quantity computed during evaluation of the objective and constraints. The resulting constraint matrix, although quite large, is extremely sparse (with at most three non-zero entries per row), and the relaxation can be formed automatically from the code list or "tape[5]" that GlobSol uses internally. Furthermore, in that context, linear underestimations need be done only for the univariate and bivariate functions occurring as elementary operations. We describe the framework for this process in [10].

When we create the LP, we desire to create a machine-representable LP that is a true relaxation of problem (1). To do this, we must take care with the direction of rounding, among other considerations, when forming the coefficients of linear underestimators and overestimators in the process we describe in [10]. We explain the details of this process, as well as put our work into the context of work by others, in [5]. Some of the techniques we describe in [5] are related to previous work in [1] and [15].

Once the machine-representable LP is formed, we solve this LP approximately. In line with our goal to make GlobSol free of license restrictions and in view of the fact that GlobSol is Fortran-based, we have used the publicly available SLATEC routine `DSPLP`, a sparse simplex method solver, for this purpose.

Once we have the approximate solution, we need to validate it. That is, we use the approximate dual variables (i.e. the Lagrange multipliers) and the approximate lower bound obtained with the floating-point LP solver to compute a rigorous lower bound (that takes account of inaccuracies in the approximate solution and roundoff error) to the solution of the LP. In turn, since the LP was constructed to be a rigorous relaxation to problem (1), this validated lower bound to the LP is also a validated lower bound to the global optimum of problem (1). We use the technique in [17] to obtain this validated lower bound.

Our use of the above process consists not only of obtaining a lower bound on the objective and an approximate optimizer, but also of obtaining (possibly narrower) bounds on the independent variables x_i. This is done by simply replacing the objective in the LP by either x_i or $-x_i$. (We do not claim this technique to be unique; for example, we understand that it is used in the BARON [19] software system, and we believe it to be commonplace.)

If the floating-point LP solver does not seem to give a feasible solution, a simple technique in [17] related to the solution validation process is sometimes successful in rigorously showing that there is no feasible point in $\boldsymbol{x}$. This allows us to safely reject $\boldsymbol{x}$ in the overall algorithm.

Unfortunately, even though the floating-point LP solver produces an approximate optimizer $\check{x}$ that, for many problems and for all problems with the coordinate widths of $\boldsymbol{x}$ sufficiently small, is an approximate solution to the

[5] i.e. the list of operations for evaluation of the constraints and objective

global optimization problem (1), rigorous bounds on an actual optimizer are not as easily available as the lower bound on the objective. However, we use the technique we explain in [7, §5.2.4, p. 185] and [8] to construct a box about $\check{x}$ in which a feasible point is rigorously proven to exist. We can then use this small box to update the upper bound $\overline{\varphi}$ on the global optimizer.

The source code for GlobSol is available, and we will make our implementation of the processes described above, in the GlobSol modules `LP_OPERATIONS` and GlobSol routine `LP_FILTER` available as part of our next public release of GlobSol.

5 Issues and Questions

If multiple tangent lines can be used as underestimators, we use an adaptive process that fits the original nonlinear function to a tolerance ϵ_{LP}, as we explain in [5]. We use the chord rule in the "sandwich algorithm" as explained in[19, §4.2]; see figure 1 for an example of multiple underestimators to a function. Although we understand this scheme to be used in the BARON software of [19] and elsewhere, we understand the number of subdivisions used in BARON to be fixed, and we are unaware of any implementation of the adaptive process.

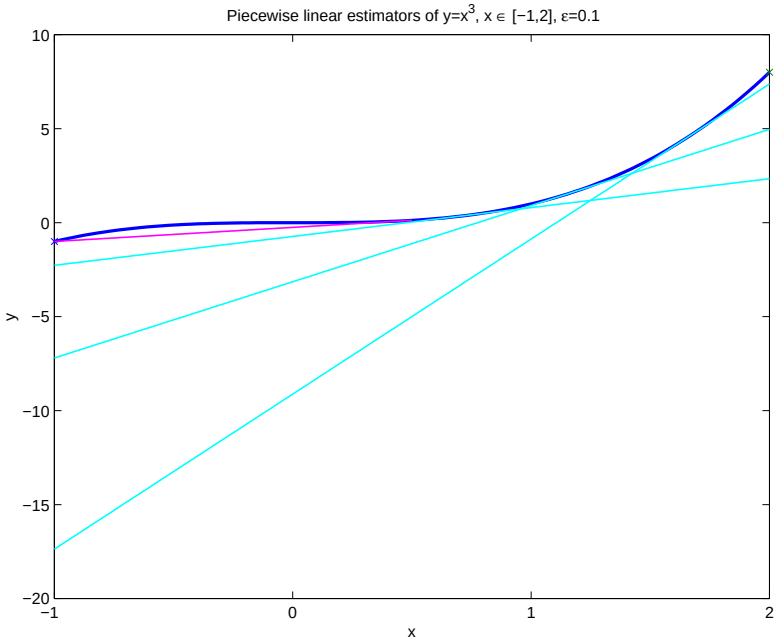

Figure 1: Multiple underestimators to $y = x^3$ obtained with the sandwich algorithm, $\epsilon_{\mathrm{LP}} = 0.1$.

Several questions thus arise concerning the tolerance ϵ_{LP}:

1. Although the convex functions can be underestimated arbitrarily closely by adaptively adding a sufficient number of linear constraints, non-convex functions cannot. Thus, if the objective or constraints contain non-convex operations, highly accurate approximation of the convex part may be overwhelmed by slackness in the approximation of the non-convex parts, and subdivision of $\boldsymbol{x}$ may be necessary anyway. How accurate should ϵ_{LP} be for optimum efficiency?

2. If a very small ϵ_{LP} is demanded, many constraints corresponding to almost parallel lines will be present. This implies ill-conditioning in the constraint matrix. For what values of ϵ_{LP} does this occur, and how does it affect the algorithm?

3. For larger boxes $\boldsymbol{x}$, the actual range of φ is larger, and ϵ_{LP} can possibly be larger and still be effective. Should a relative or an absolute tolerance be used for ϵ_{LP}?

There are various other general issues, as follows:

1. The LP relaxations can be formed and solved at various points in the global search algorithm. In particular, they can be applied before or after the nonlinear constraint propagation and the interval Newton method, and the three processes can be iterated, without bisection of $\boldsymbol{x}$ until the iteration becomes approximately stationary. Is it advantageous to do such iteration?

2. Using the LP relaxations reduces the total number of boxes that need to be processed, but it increases the processing time per box. When and where we actually apply the LP relaxations would thus depend on the balance of the time spent and reduction in total number of boxes processed. Also, there may be skepticism about whether validated implementations of linear relaxations can work as efficiently as non-validated ones. Thus, we desire answers to the following questions:

   - How much CPU time is spent totally in setting up and solving the LP relaxation, compared to other parts of the global search algorithm?
   - How much CPU time is spent actually in solving the LP (i.e. the time within the routine DSPLP, in our implementation)?
   - What effect does solving the LP relaxation have on the total number of boxes $\boldsymbol{x}$ the algorithm must process, for various problems?

   The answers to such questions will also reveal how critical it is to have an efficient LP solver.

Finally, there are particular problems for which GlobSol without LP relaxations is not practical, but for which the BARON [19] does give an answer. We

wish to determine whether the implementation of LP relaxations we have outlined above can be effective on these problems. In particular, we are studying the nonlinear minimax problem

$$\min_x \max_{1\le i\le m} |f_i(x)|, \quad f_i : \mathbb{R}^n \to \mathbb{R}, \quad x \in \mathbb{R}^n, \quad m \ge n. \tag{2}$$

When we use Lemaréchal's technique [12], we convert this problem to

$$\begin{array}{l} \min_{x\in\mathbb{R}^n} v \\ \text{such that } \left\{ \begin{array}{rcl} f_i(x) & \le & v \\ -f_i(x) & \le & v \end{array} \right\}, \quad 1 \le i \le m. \end{array} \tag{3}$$

As we indicated in [11], alternate methods in GlobSol for solving problem (3) have been impractical, while linear relaxations show promise. We will use the following problem as an initial test of how our LP relaxation implementation handles minimax problems:

$$\begin{array}{l} \min_{x\in\mathbb{R}^n} x_5 \\ \text{such that } \left\{ \begin{array}{rcl} x_4 - \left(x_1 t_i^2 + x_2 t_i + x_3\right)^2 - \sqrt{t_i} \;\; - x_5 & \le & 0 \\ -\left\{x_4 - \left(x_1 t_i^2 + x_2 t_i + x_3\right)^2 - \sqrt{t_i}\right\} - x_5 & \le & 0 \end{array} \right\}, \\ 1 \le i \le m, \text{ where } t_i = 0.25 + 0.75(i-1)/(m-1). \end{array} \tag{4}$$

Problem 4, proposed as a test problem in [23], represents a minimax fit to $\sqrt{t}$ with a function of the form $p(t) = x_4 - \left(x_1 t_i^2 + x_2 t_i + x_3\right)^2$, with m equally spaced data points in $[0.25, 1]$.

This problem has at least two solutions, since replacing $(x_1, x_2, x_3, x_4, x_5)$ by $(-x_1, -x_2, -x_3, x_4, x_5)$ results in the same constraint values. In fact our linear relaxation version of GlobSol proves that there are precisely two solutions, for $m = 5$ and $m = 21$.

GlobSol without linear relaxations has been unable to obtain bounds on the solution sets to (4), using starting intervals $x_i \in [-5, 5]$, $i = 1, 2, 3, 4$, and $x_5 \in [-100, 100]$, for $m = 21$ (as originally stated in [23], or even for $m = 5$. However, as we see in §6 below, linear relaxations can solve these problems, although further development is warranted.

6 Experimental Results

We inserted the `LP_FILTER` we explained in §4 between step 3 and step 4 of the GlobSol overall algorithm, i.e. Algorithm 1. We also inserted the option of iterating step 3, `LP_FILTER`, and step 4 to convergence before doing a bisection (step 6). When the `LP_FILTER` process is executed, we also check the approximate optimum of the relaxation by constructing a small box around it and attempting to verify existence of a feasible point within that box; if this approximate optimum corresponds to a feasible point of the original nonlinear

program (1) and the relaxation is a good approximation to program (1), then we can use the constructed box to reduce $\overline{\varphi}$ for the lower bound test.

We compiled this modified version of GlobSol with Compaq Visual Fortran version 6.6, without optimization, on a Dell Inspiron 8200 notebook with a mobile Pentium 4 processor running at 1.60 GHz. (Memory was not an issue in these problems, as the modified version of GlobSol ran in a relatively modest amount of memory for the problems tried.)

6.1 Exploratory Results for Our Initial Problem

We first report some initial exploratory results for problem 4 in table 1. In

Table 1: Results for problem 4. (See text for column headings.)

run #	m	ϵ_d	ϵ_{LP}	rel?	it?	ok?	NB	T_T
1	5	10^{-4}	10^{-2}	N	T	F	11798	60
2	5	10^{-4}	10^{-1}	N	F	T	16372	55
3	5	10^{-4}	10^{-2}	N	F	T	13993	48
4	5	10^{-4}	10^{-3}	N	F	T	14596	66
5	5	10^{-8}	10^{-3}	N	F	T	12672	56
6	5	10^{-8}	10^{-4}	N	F	F	11	171
7	5	10^{-8}	10^{-2}	N	F	T	12677	44
8	5	10^{-4}	10^{-1}	Y	F	T	15268	50
9	5	10^{-8}	10^{-1}	Y	F	T	14797	49
10	5	10^{-8}	2×10^{-2}	Y	F	T	15263	48
11	5	10^{-8}	2×10^{-2}	Y	F	T	15263	49
12	21	10^{-8}	2×10^{-2}	Y	F	T	19822	393

Table 2: Timings and effectiveness for problem 4. (See text.)

run #	T_T	T_{LP}	T_{IN}	R_C	R_{LP}	R_{IN}	N_c	N_a	N_u
1	60	58	1.0	261	158	131	2483	1	0
2	55	53	0.9	307	37	138	0	2	10
3	48	47	0.6	417	157	131	0	2	0
4	66	65	0.8	1076	193	325	0	2	14
5	56	55	0.9	906	230	380	0	2	0
6	171	171	0.0	1	0	0	16	1	0
7	44	43	0.9	494	145	368	0	2	0
8	50	48	0.9	297	20	249	0	2	17
9	49	47	1.0	321	30	337	0	2	0
10	48	47	0.9	353	37	218	0	2	0
11	49	47	0.9	353	37	218	0	2	0
12	393	376	14.3	247	26	379	0	2	0

table 1, m is the number of data points, as in the definition of problem 4, ϵ_d is

the tolerance `EPS_DOMAIN` set in GlobSol (at the top of the box data file), and $\epsilon_{\rm LP}$ is the tolerance in the "sandwich algorithm" as explained in §5 and [5]. That is, creation of additional underestimates to convex functions (or of overestimates to concave functions) completes if the maximum distance d from the underestimate to the function obeys $d < \epsilon_{\rm LP}$ for an absolute tolerance, and $d < \epsilon_{\rm LP} * (b-a)$ for a relative tolerance, where $b-a$ is the total length of the interval over which the underestimates are made. (For example, in figure 1, $a = -1$ and $b = 1$.) The column labeled "rel?" indicates whether a relative or an absolute tolerance was used for $\epsilon_{\rm LP}$; thus, for runs 1 through 7, an absolute tolerance was used, whereas for runs 8 through 12, a relative tolerance was used. The column labeled "it?" indicates whether step 3, `LP_FILTER`, and step 4 of Algorithm 1 are iterated; thus, iteration was done in run 1 only. The column labeled "ok?" indicates whether or not the run completed successfully; in particular, only run 1 and run 6 did not complete successfully (The only reason for an unsuccessful completion is exceeding the maximum CPU time. This was set to 60 minutes for runs 1 through 3 and to 166 minutes or 10000 seconds for the remaining runs, except for run 12, where it was set higher.) The column labeled "NB" gives the total number of subregions $\boldsymbol{x}$ that GlobSol processed, while the last column, labeled "T_T" gives the total execution time in minutes.

Table 2 continues Table 1 with additional data on where time was spent in the algorithm and how effective constraint propagation (step 3), the `LP_FILTER` process, and the interval Newton method were at rejecting or reducing the size of regions $\boldsymbol{x}$. Column 1 gives the problem number as in Table 1, while column 2 repeats the total execution. The column labeled $T_{\rm LP}$ gives the total time spent in `LP_FILTER`; of this, in all cases, roughly 97% was in the actual LP solver (the SLATEC routine `DSPLP`), and the rest was in setting up the LP and in the Neumaier / Shcherbina validation technique. The column labeled $T_{\rm IN}$ gives the total time in the interval Newton method. The column labeled R_C gives the number of times constraint propagation rejected a box, the column labeled $R_{\rm LP}$ gives the number of times `LP_FILTER` rejected a box, while the column labeled $R_{\rm IN}$ gives the number of times the interval Newton method rejected a box; these three columns give measures of the effectiveness of these three processes. The last three columns give measures of the quality of the solution. In particular N_C gives the number of boxes that have not yet been processed, and is non-zero only if the algorithm could not complete within the allocated CPU time. N_a gives the number of boxes in the answer set verified to contain a feasible point, while N_u gives the number of small boxes which were neither verified to contain feasible points nor rejected. For this problem, ideally $N_c = 2$ and $N_u = 0$, since there are precisely two solutions. Larger values of N_u typically happen in GlobSol when ϵ_d is too large.

Preliminary inferences from Table 1 and Table 2 are as follows:

- Iteration of step 3, `LP_FILTER`, and step 4 of Algorithm 1 does not appear to be worth it in this case.
- Most of the CPU time is in solving the LP. Although use of linear relaxations is indispensable for this problem, sparse LP solvers should perhaps

be investigated and compared for this purpose.

- Except as noted below, the performance of the algorithm does not seem to depend critically on the exact value of $\epsilon_{\rm LP}$, although too large a value in conjunction with an absolute, rather than relative tolerance can result in many unresolved boxes (as in run #2).

Although not evident in Table 1 and Table 2, we also observed that the solver `DSPLP` had trouble with ill-conditioning when an absolute tolerance was used with $\epsilon_{\rm LP}$ and $\epsilon_{\rm LP} = 10^{-4}$ or less. This may vary with different LP solvers.

6.2 Systematic Comparisons

For a more systematic comparison (unbiased by our own knowledge of the algorithm), we ran a subset of those problems Shcherbina and Neumaier collected as part of the COCONUT project [18]. We selected those "tiny" problems (10 variables or less) from Library 1 for which we had implemented rigorous linear underestimators and overestimators. (We had not yet implemented rigorous linear underestimators and overestimators for trigonometric functions, nor x^a, nor underestimators and overestimators of x/y except for $x \geq 0$ and $y > 0$) when we ran these experiments.) The process of running these tests uncovered several bugs or inadequacies in our implementation of rigorous underestimators and overestimators. In particular, the relative tolerance used in the exploratory results of §6.1, namely, $d < \epsilon_{\rm LP} * (b - a)$, was inadequate when the values of the function being approximated were large or when the curvature of the function being approximated changed rapidly, and led to an excessive number of underestimating constraints. Instead, we adopted the criterion to stop subdividing and adding new constraints when:

$$\begin{array}{l} \text{either } d < \epsilon_{\rm LP} \max\{|f(\check{x})|, 1\} \\ \text{or } r - \ell < \epsilon_d \max\{|r|, |\ell|\}, \end{array} \tag{5}$$

where f is the function being approximated, and where d, ℓ, and r are as in step 6 of Algorithm 2 of [5]. That is, we replace $d < \epsilon$ in step 6 of Algorithm 2 of [5] with (5).

In all runs, we used (5) with $\epsilon_{\rm LP} = 10^{-1}$, $\epsilon_d = 10^{-8}$, and we terminated execution with failure if either 3600 CPU seconds elapsed or over 100,000 boxes were considered.

The results with linear relaxations occur in Table 3, the results without using linear relaxations appear in Table 4, and a comparison of the two runs appears in Table 5. In Table 3, the column labeled T_{LP}/T_T represents the ratio of the total time spent forming and solving the linear relaxations to the total time spent in GlobSol, the column labeled "name" is the name as given in the COCOS test problem collection, and the other columns in tables 3, 4, and 5 are as in Table 1 or are self-explanatory. In the "Totals or averages" row of Table 5, the percentage in the "ok (LP)?" column represents the proportion of problems that completed successfully with LP filtering, the percentage in the "ok (no

LP)?" column represents the proportion of problems that completed successfully without LP filtering, the percentage in the $T_{\text{LP}}/T_{\text{no LP}}$ column represents the average proportion of total execution times (LP filtering to no LP filtering), and the percentage in the $\frac{\text{NB}_{\text{LP}}}{\text{NB}_{\text{no LP}}}$ column represents the corresponding average proportion of total number of boxes processed[6] in Algorithm 1. (More detailed tables, including execution times for portions of the algorithm and numbers of times various parts of the algorithm succeeded, can be obtained from the author.)

Examining Table 5, we see that, in all but 5 of the 55 problems, the total number of boxes is less with the LP filtering than without the LP filtering, although the execution time is often significantly more. The times when the number of boxes processed is more with the LP filtering must be due to the box complementation process (which can generate more, albeit smaller, boxes) following successful verification of a feasible point associated with the approximate optimum of an LP.

There were 8 problems in which GlobSol completed with both the LP filter and without the LP filter but GlobSol with the LP filter took more than ten times the execution time of GlobSol without the LP filter. (These are `ex4.1.5`, `ex4.1.7`, `ex6.1.2`, `ex6.1.4`, `ex6.2.14`, `ex8.1.5`, `ex8.1.6`, and `least`.) Of these, 4 (50%) were unconstrained (other than the bound constraints, which were handled by preprocessing), in contrast to only 11 unconstrained problems of the 55 total problems (20%). Furthermore, none of the 8 problems that required more than 10 times the time with LP filtering took more than 1/6 the maximum allocated time for GlobSol to solve with the LP filter, and all but three of them took less than 1% of the maximum allocated time.

In contrast, there were 6 problems in which GlobSol with the LP filtering process completed and took less time to complete than GlobSol without LP filtering. (These problems are `ex14.1.1`, `ex14.1.3`, `ex4.1.9`, `ex5.4.2`, `ex7.3,2`, and `sample`.) All of these were equality-constrained problems. The three with the most striking reductions in execution time, namely, `ex14.1.` (7%), `ex5.4.2` (27%), and `sample` (19%) had a large number of inequality constraints (with `ex14.1.3` actually having more equality constraints than variables). They also are among the 23 problems which took the most execution time for GlobSol without the LP filter, and the fastest one (`ex14.1.1` took roughly six times the average of the execution time of the programs that ran faster than it. Thus, the problems on which the LP filter was most effective appear to be the harder highly equality-constrained problems.

In all but one of the 15 failures with LP filtering, GlobSol exceeded the maximum time limit, and an average of 84% of the time was spent in formulating and solving the LP problem (mostly in solving the LP problem; see §6.1 above).

[6]In some cases, the total number of boxes processed is 0. This is because, in all of the problems in the COCOS tiny-1 test set, the "peeling" process of [7, §5.2.3] was applied for each coordinate, resulting in preprocessing, including application of an interval Newton method, which may eliminate all possibilities before Algorithm (1) begins. Because the denominator in the last column for the "rbrock" row of Table 5 was zero for that region, we replaced the infinite percentage by 100% in that case.

In contrast, 4 of the ten failures for GlobSol to complete without the LP filtering were due to exceeding 100,000 boxes processed in the loop.

Of particular interest are problems `ex6.1.1` and `sample`. In `ex6.1.1`, although GlobSol without LP filtering completed, it almost exceeded the allocated execution time and processed 27,610 boxes, while GlobSol with LP filtering had only processed 75 boxes, and nearly 100% of the execution time was in solving the LP problem. It appears that, for `ex6.1.1`, a faster LP solver or, perhaps, better heuristics for choosing when to apply the LP filter and how fine to approximate the original nonlinear program with the linear relaxation would lead to a significantly different result.

For problem `sample`, GlobSol without LP filtering failed, exceeding the allocated execution time and also almost exceeding the 100,000 maximum number of boxes allowable, whereas GlobSol with LP filtering succeeded in only 19% of the allowable CPU time, with only 145 boxes.

In contrast to the minimax problem of §6.1 above, GlobSol without LP filtering completed within an hour of time and 100,000 boxes in slightly more of the problems than GlobSol with the LP filtering (81% versus 73%).

7 Conclusions and Future Work

We have implemented an initial version of linear relaxations in a research validated global optimization code that is nonetheless representative of "industrial strength" codes in its inclusion of a multitude of interacting techniques.

We have demonstrated that linear relaxations can be implemented in a practical way within a validated context, and that they can enable reduction of the search region in contexts where traditional interval methods fail. Although our present implementation leads to more execution time for many problems, the linear relaxations make solution of some problems practical when they would not otherwise be, particularly for highly constrained problems with simple objective functions. There are indications that the execution time for problems for which LP relaxations do not lead to lower execution times could be significantly reduced with a faster LP solver. Also, our experiments hint that the linear relaxations may be more valuable on faster machines used to solve larger equality-constrained problems.

A preliminary study of the details of some of the problems where the present GlobSol implementation results in a large amount of computation appears to indicate that a better approximate local optimizer for constrained problems would help in this regard. To date, GlobSol has not included such an optimizer, since we have included only our own code and code that can be freely distributed, and since our development efforts have kept us busy on other aspects of the global optimization process. However, freely distributable local approximate optimizers such as IPOPT [22, 21] have become available in recent years. We will experiment with these in the near future, as well as continue to explore inclusion of linear relaxation technology into validated global optimization codes.

Another phenomenon revealed to use while examining the details was that,

when many low-quality approximate optimizers (such as mere feasible points) were found, GlobSol's box complementation process (as in [7, §4.3.1]) resulted in many traversals of the linked lists of boxes $\mathcal{L}$, $\mathcal{U}$ and $\mathcal{C}$ of Algorithm 1, and most of the execution time was taken in this process. As a consequence, we implemented a better heuristic (not processing feasible-point-containing boxes if the upper bound on the objective over the box is greater than $\overline{\varphi}$ of the lower bound test). This situation could also be improved with improved programming or data-structure techniques for processing the box lists, but will undoubtedly also be improved by employing a better approximate optimizer.

Finally, in [10], we have proposed an alternate scheme for choosing on which variable to branch in the overall branch and bound algorithm. This scheme is based upon an analysis carried out during the process of constructing the linear relaxations, but the branching scheme used in this paper was the "maximum smear" scheme ([7, p. 157 and p. 175], with a modification to include constraints) that GlobSol has used all along. Additional investigation of the more sophisticated analysis in [10] is warranted.

Acknowledgements

I wish to thank Siriporn Hongthong for Figure 1. I also wish to thank George Corliss, William Dean, Mihye Kim, and Siriporn Hongthong for commenting on the manuscript.

References

[1] G. Borradaile and P. Van Hentenryck. Safe and tight linear estimators for global optimization. In *Workshop on Interval Analysis and Constraint Propagation for Applications (IntCP 2003)*, 2003. `http://www.cs.brown.edu/people/pvh/linest.ps`.

[2] L. C. W. Dixon and G. P. Szego, editors. *Towards Global Optimization.* North-Holland, 1975.

[3] L. C. W. Dixon and G. P. Szego, editors. *Towards Global Optimization 2.* North-Holland, 1977.

[4] C. A. Floudas. *Deterministic Global Optimization: Theory, Algorithms and Applications.* Kluwer, Dordrecht, Netherlands, 2000.

[5] S. Hongthong and R. B. Kearfott. Rigorous linear overestimators and underestimators, 2004. preprint, `http://interval.louisiana.edu/preprints/estimates_of_powers.pdf`.

[6] R. B. Kearfott. Decomposition of arithmetic expressions to improve the behavior of interval iteration for nonlinear systems. *Computing*, 47(2):169–191, 1991.

[7] R. B. Kearfott. *Rigorous Global Search: Continuous Problems.* Kluwer, Dordrecht, Netherlands, 1996.

[8] R. B. Kearfott. On proving existence of feasible points in equality constrained optimization problems. *Math. Prog.*, 83(1):89–100, September 1998.

[9] R. B. Kearfott. Globsol: History, composition, and advice on use. In *Global Optimization and Constraint Satisfaction*, Lecture Notes in Computer Science, pages 17–31, New York, 2003. Springer-Verlag.

[10] R. B. Kearfott and S. Hongthong. A preprocessing heuristic for determining the difficulty of and selecting a solution strategy for nonconvex optimization problems, 2003. preprint, `http://interval.louisiana.edu/preprints/2003_symbolic_analysis_of_GO.pdf`.

[11] R. B. Kearfott, M. Neher, S. Oishi, and F. Rico. Libraries, tools, and interactive systems for verified computations: Four case studies. In R. Alt, A. Frommer, R. B. Kearfott, and W. Luther, editors, *Numerical Software with Result Verification, Lecture Notes in Computer Science no. 2991*, pages 36–63, New York, 2004. Springer-Verlag.

[12] C. Lemaréchal. Nondifferentiable optimization. In M. J. D. Powell, editor, *Nonlinear Optimization 1981*, pages 85–89, New York, 1982. Academic Press.

[13] G. P. McCormick. Converting general nonlinear programming problems to separable nonlinear programming problems. Technical Report T-267, George Washington University, Washington, 1972.

[14] G. P. McCormick. Computability of global solutions to factorable nonconvex programs. *Math. Prog.*, 10(2):147–175, April 1976.

[15] C. Michel, Y. Lebbah, and M. Rueher. Safe embedding of the simplex algorithm in a CSP framework. In *Proceedings of the Fifth International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization*, 2003. `http://www.crt.umontreal.ca/cpaior/article-michel.pdf`.

[16] J. J. Moré, B. S. Garbow, and K. E. Hillstrom. User guide for MINPACK-1. Technical Report ANL-80-74, Argonne National Laboratories, 1980.

[17] A. Neumaier and O. Shcherbina. Safe bounds in linear and mixed-integer programming. *Math. Prog.*, 99(2):283–296, March 2004. `http://www.mat.univie.ac.at/~neum/ms/mip.pdf`.

[18] O. Shcherbina and A. Neumaier. Benchmarking global optimization and constraint satisfaction codes. In Ch. Bliek, Ch. Jermann, and A. Neumaier, editors, *Global Optimization and Constraint Satisfaction*, pages 211–222, New York, 2003. Springer-Verlag. `http://www.mat.univie.ac.at/~neum/ms/bench.pdf`.

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

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

[21] A. Wächter. Homepage of IPOPT, 2002. `http://www-124.ibm.com/developerworks/opensource/coin/Ipopt/`.

[22] A. Wächter. *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering.* PhD thesis, Carnegie Mellon University, 2002. `http://dynopt.cheme.cmu.edu/andreasw/thesis.pdf`.

[23] J. L. Zhou and A. L. Titts. An SQP algorithm for finely discretized continuous minimax problems and other minimax problems with many objective functions. *SIAM J. Optim.*, 6(2):461–487, 1996.

Table 3: Results for the COCOS "tiny-1" test set with linear relaxations. (See text for column headings.)

Name	n	m_1	m_2	ok?	NB	T_T	T_{LP}/T_T
ex14.1.1	3	0	4	T	91	0.1883	96%
ex14.1.2	6	1	8	T	84	0.2137	63%
ex14.1.3	3	0	4	T	278	0.1293	78%
ex14.1.5	6	4	2	T	71	0.1933	83%
ex14.1.9	2	0	2	T	47	0.0132	68%
ex14.2.1	5	1	6	T	159	0.5452	90%
ex14.2.2	4	1	4	T	16	0.0100	40%
ex14.2.3	6	1	8	T	214	1.3413	85%
ex14.2.5	4	1	4	T	68	0.3235	95%
ex2.1.1	5	0	1	T	234	0.0930	79%
ex2.1.2	6	0	2	T	173	0.0768	19%
ex2.1.4	6	0	4	T	222	0.1640	48%
ex3.1.1	8	0	6	F	20229	60.1775	77%
ex3.1.4	3	0	3	T	7	0.0038	35%
ex4.1.2	1	0	0	T	6	0.0118	82%
ex4.1.4	1	0	0	T	9	0.0015	67%
ex4.1.5	2	0	0	T	47	0.0110	82%
ex4.1.6	1	0	0	T	4	0.0003	0%
ex4.1.7	1	0	0	T	9	0.0017	90%
ex4.1.8	2	1	0	T	5	0.0005	33%
ex4.1.9	2	0	2	T	35	0.0067	85%
ex5.2.4	7	1	5	F	36980	60.2352	82%
ex5.4.2	8	0	6	T	515	1.3218	56%
ex6.1.1	8	6	0	F	75	100.9517	100%
ex6.1.2	4	3	0	T	109	0.4143	95%
ex6.1.4	6	4	0	T	420	4.4903	97%
ex6.2.10	6	3	0	F	1901	60.0987	95%
ex6.2.11	3	1	0	F	7686	60.0127	87%
ex6.2.12	4	2	0	F	5820	60.0252	91%
ex6.2.13	6	3	0	F	2719	60.0785	95%
ex6.2.14	4	2	0	T	1383	10.4143	91%
ex6.2.6	3	1	0	F	8871	60.2438	71%
ex6.2.8	3	1	0	F	10637	60.0058	95%
ex6.2.9	4	2	0	F	3558	60.0280	81%
ex7.2.2	6	4	1	T	104	0.4575	91%
ex7.2.5	5	0	6	T	108	0.2740	89%
ex7.2.6	3	0	1	T	39	0.0103	66%
ex7.3.1	4	0	7	T	33	0.1962	93%
ex7.3.2	4	0	7	T	9	0.0043	54%
ex7.3.3	5	2	6	T	33	0.1855	90%
ex8.1.3	2	0	0	F	62334	60.0003	81%
ex8.1.4	2	0	0	T	28	0.0070	79%
ex8.1.5	2	0	0	T	131	0.0560	89%
ex8.1.6	2	0	0	T	29	0.0382	98%
ex8.1.7	5	1	4	F	30105	60.0090	69%
ex8.1.8	6	4	1	T	104	0.4620	91%
ex9.2.4	8	7	0	F	71983	62.8002	70%
ex9.2.5	8	7	0	F	11928	61.1955	87%
house	8	4	4	F	100000	22.7447	56%
least	3	0	0	T	1407	5.0975	92%
meanvar	7	2	0	T	2124	29.2568	98%
mhw4d	5	3	0	T	192	0.3673	92%
nemhaus	5	0	0	T	0	0.0002	0%
rbrock	2	0	0	T	4	0.0013	75%
sample	4	0	2	T	145	11.1345	100%

Table 4: Results for the COCOS "tiny-1" test set without linear relaxations. (See text for column headings.)

Name	n	m_1	m_2	ok?	NB	T_T
ex14.1.1	3	0	4	T	3183	0.2037
ex14.1.2	6	1	8	T	86	0.0803
ex14.1.3	3	0	4	T	4168	1.7860
ex14.1.5	6	4	2	T	100	0.0548
ex14.1.9	2	0	2	T	186	0.0075
ex14.2.1	5	1	6	T	209	0.0550
ex14.2.2	4	1	4	T	16	0.0057
ex14.2.3	6	1	8	T	230	0.1728
ex14.2.5	4	1	4	T	356	0.0715
ex2.1.1	5	0	1	T	241	0.0180
ex2.1.2	6	0	2	T	168	0.0633
ex2.1.4	6	0	4	T	222	0.0855
ex3.1.1	8	0	6	F	67096	60.1745
ex3.1.4	3	0	3	T	27	0.0030
ex4.1.2	1	0	0	T	6	0.0018
ex4.1.4	1	0	0	T	9	0.0003
ex4.1.5	2	0	0	T	28	0.0008
ex4.1.6	1	0	0	T	4	0.0003
ex4.1.7	1	0	0	T	4	0.0002
ex4.1.8	2	1	0	T	5	0.0003
ex4.1.9	2	0	2	T	280	0.0097
ex5.2.4	7	1	5	F	100000	43.9670
ex5.4.2	8	0	6	T	4449	4.9393
ex6.1.1	8	6	0	T	27610	53.3900
ex6.1.2	4	3	0	T	122	0.0242
ex6.1.4	6	4	0	T	675	0.2373
ex6.2.10	6	3	0	F	46493	60.0880
ex6.2.11	3	1	0	T	11647	4.5502
ex6.2.12	4	2	0	T	6241	3.2658
ex6.2.13	6	3	0	F	49787	60.0702
ex6.2.14	4	2	0	T	1196	0.5315
ex6.2.6	3	1	0	T	18506	5.0968
ex6.2.8	3	1	0	T	14005	3.4003
ex6.2.9	4	2	0	T	11237	7.7175
ex7.2.2	6	4	1	T	142	0.0867
ex7.2.5	5	0	6	T	160	0.0383
ex7.2.6	3	0	1	T	53	0.0033
ex7.3.1	4	0	7	T	81	0.0443
ex7.3.2	4	0	7	T	25	0.0062
ex7.3.3	5	2	6	T	69	0.0557
ex8.1.3	2	0	0	F	100000	26.2655
ex8.1.4	2	0	0	T	28	0.0013
ex8.1.5	2	0	0	T	128	0.0045
ex8.1.6	2	0	0	T	27	0.0008
ex8.1.7	5	1	4	F	92983	60.0070
ex8.1.8	6	4	1	T	142	0.0870
ex9.2.4	8	7	0	F	86551	64.0362
ex9.2.5	8	7	0	F	100000	39.0192
house	8	4	4	F	100000	9.5718
least	3	0	0	T	1443	0.4643
meanvar	7	2	0	T	11558	4.1443
mhw4d	5	3	0	T	372	0.0855
nemhaus	5	0	0	T	0	0.0002
rbrock	2	0	0	T	13	0.0003
sample	4	0	2	F	92264	60.0017

Table 5: Comparison of success rates with and without linear relaxations.)

Name	n	m_1	m_2	ok (no LP)?	ok (LP)?	$\frac{\tau_{\text{LP}}}{\tau_{\text{no LP}}}$	$\frac{\text{NB}_{\text{LP}}}{\text{NB}_{\text{no LP}}}$
ex14.1.1	3	0	4	T	T	92%	3%
ex14.1.2	6	1	8	T	T	266%	98%
ex14.1.3	3	0	4	T	T	7%	7%
ex14.1.5	6	4	2	T	T	353%	71%
ex14.1.9	2	0	2	T	T	176%	25%
ex14.2.1	5	1	6	T	T	991%	76%
ex14.2.2	4	1	4	T	T	176%	100%
ex14.2.3	6	1	8	T	T	776%	93%
ex14.2.5	4	1	4	T	T	452%	19%
ex2.1.1	5	0	1	T	T	517%	97%
ex2.1.2	6	0	2	T	T	121%	103%
ex2.1.4	6	0	4	T	T	192%	100%
ex3.1.1	8	0	6	F	F	100%	30%
ex3.1.4	3	0	3	T	T	128%	26%
ex4.1.2	1	0	0	T	T	645%	100%
ex4.1.4	1	0	0	T	T	450%	100%
ex4.1.5	2	0	0	T	T	1320%	168%
ex4.1.6	1	0	0	T	T	100%	100%
ex4.1.7	1	0	0	T	T	1000%	225%
ex4.1.8	2	1	0	T	T	150%	100%
ex4.1.9	2	0	2	T	T	69%	13%
ex5.2.4	7	1	5	F	F	137%	37%
ex5.4.2	8	0	6	T	T	27%	12%
ex6.1.1	8	6	0	T	F	189%	0%
ex6.1.2	4	3	0	T	T	1714%	89%
ex6.1.4	6	4	0	T	T	1892%	62%
ex6.2.10	6	3	0	F	F	100%	4%
ex6.2.11	3	1	0	T	F	1319%	66%
ex6.2.12	4	2	0	T	F	1838%	93%
ex6.2.13	6	3	0	F	F	100%	5%
ex6.2.14	4	2	0	T	T	1959%	116%
ex6.2.6	3	1	0	T	F	1182%	48%
ex6.2.8	3	1	0	T	F	1765%	76%
ex6.2.9	4	2	0	T	F	778%	32%
ex7.2.2	6	4	1	T	T	528%	73%
ex7.2.5	5	0	6	T	T	715%	68%
ex7.2.6	3	0	1	T	T	310%	74%
ex7.3.1	4	0	7	T	T	442%	41%
ex7.3.2	4	0	7	T	T	70%	36%
ex7.3.3	5	2	6	T	T	333%	48%
ex8.1.3	2	0	0	F	F	228%	62%
ex8.1.4	2	0	0	T	T	525%	100%
ex8.1.5	2	0	0	T	T	1244%	102%
ex8.1.6	2	0	0	T	T	4580%	107%
ex8.1.7	5	1	4	F	F	100%	32%
ex8.1.8	6	4	1	T	T	531%	73%
ex9.2.4	8	7	0	F	F	98%	83%
ex9.2.5	8	7	0	F	F	157%	12%
house	8	4	4	F	F	238%	100%
least	3	0	0	T	T	1098%	98%
meanvar	7	2	0	T	T	706%	18%
mhw4d	5	3	0	T	T	430%	52%
nemhaus	5	0	0	T	T	100%	100%
rbrock	2	0	0	T	T	400%	31%
sample	4	0	2	F	T	19%	0%
Totals or averages				81%	73%	617%	66%