Optimization Methods and Software
Vol. 00, No. 00, Month 200x, 1–14

Research Article

Assessment of a Non-Adaptive Deterministic Global Optimization Algorithm for Problems with Low-Dimensional Non-Convex Subspaces

Ralph Baker Kearfott
Department of Mathematics, University of Louisiana at Lafayette, U.L. Box 4-1010, Lafayette, LA 70504-1010 USA
Email: rbk@louisiana.edu,

Jessie M. Castille
Department of Mathematics, University of Louisiana at Lafayette,
Email: jmc4491@louisiana.edu,

and
Gaurav Tyagi
Department of Mathematics, University of Louisiana at Lafayette,
Email: gxt3687@louisiana.edu

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

The optimum and at least one optimizing point for convex nonlinear programs can be approximated well by the solution to a linear program (a fact long used in branch and bound algorithms). In more general problems, we can identify subspaces of "nonconvex variables" such that, if these variables have sufficiently small ranges, the optimum and at least one optimizing point can be approximated well by the solution of a single linear program. If these subspaces are low-dimensional, this suggests subdividing the variables in the subspace *a priori*, then producing and solving a fixed, known number of linear programs to obtain an approximation to the solution. The total amount of computation is much more predictable than that required to complete a branch and bound algorithm, and the scheme is "embarrassingly parallel," with little need for either communication or load balancing.

We compare such a non-adaptive scheme experimentally to our GlobSol branch and bound implementation, on those problems from the COCONUT project Lib1 test set with non-convex subspaces of dimension 4 or less, and we discuss potential alterations to both the non-adaptive scheme and our branch and bound process that might change the scope of applicability.

AMS Subject Classification: 90C26; 49M20.

Keywords: non-convex optimization, branch and bound algorithms, linear relaxations

1. Introduction

During the past several decades, branch and bound algorithms have become a primary paradigm in deterministic global optimization[1]. Familiar not only in nonconvex global optimization, but also highly important in mixed integer linear programming, branch and bound methods (and variants termed "branch and cut," etc.) consist of a careful, adaptive subdivision of the domain of decision variables, with termination conditions based primarily upon an upper bound for the global optimum and a lower bound for the objective on the feasible portion of the subdomains. Practical branch and bound algorithms and software include numerous

[1]This does not include statistical or heuristic methods, exemplified by evolutionary algorithms, simulated annealing, and the like.

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

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

special techniques to accelerate the process, and such techniques are currently the focus of a considerable amount of research. These techniques largely deal with obtaining better upper bounds on the global optimum and better lower bounds on the objective over subdomains[2]. Salient among these techniques is use of linear relaxations, as in e.g. [19], or, more generally, convex or structured relaxations, as in [3, 4], etc..

Actual software, now amazingly effective in view of the inherent exponential nature of subdividing high-dimensional domains, has been built by including those techniques which have been effective for particular problems of practical interest or which have proven themselves to increase success and efficiency on-average over large test sets [18]. Extensive theoretical study has been made of individual acceleration techniques, but theoretical analysis of their effect, combined with other acceleration techniques, in practical branch and bound software is lacking. (Indeed, such theoretical analysis may not be possible, in general.)

In fact, despite the effectiveness of current software, its behavior on particular new problems is not always easily predictable. In contrast, it has generally been assumed in the software's design that the optima of the relaxations converge to the global optimum of the original problem, and that the solutions to the relaxations become nearer to the solution set of the original problem, as the diameter of the subdomain tends to zero. Indeed, we have written down some formal reasoning for this assumption in [9].

It follows from such reasoning that we may obtain an arbitrarily accurate approximation to the optimum and to at least one optimizing point by subdividing each independent (i.e. decision) variable into sufficiently small intervals and solving a linear program on each of the subdomains consisting of cross-products of these subintervals. Such a scheme has a more predictable time-to-completion than branch and bound schemes, since the total number of subregions is known a priori, and since the size and structure of the linear programs, and thus the time required to solve them, should not vary much over the different subregions. Thus, an answer of some quality will be provided in a known amount of time, or else it can be determined a priori that is not practical to solve the problem in that manner. Furthermore, each linear program solution can be considered independently of the others, so the process can be "embarrassingly parallel[1]." This contrasts with branch and bound processes, where the amount of work to fathom (i.e. to complete processing on) a particular node (i.e. a particular subregion) is not easily predictable, and load balancing may be needed in a multiprocessing context.

On the other hand, while a stopping tolerance is generally input into a branch and bound scheme, and the branch and bound does not complete until the tolerance is met, it is not so easy with the non-adaptive scheme to determine a priori how fine the independent variable subdivision must be for the gap between the upper and lower bounds on the global optimum to be sufficiently small. Furthermore, uniformly subdividing sufficiently in each independent variable will usually generate a prohibitive number of subdivisions for many problems, when the dimension of the domain is more than just several variables.

Nonetheless, there are problems in which the non-adaptive subdivision may be effective, or even superior, to branch and bound. For instance, it is possible in some cases to identify subsets of the independent variables or subsets of the intermediate

[2]An additional important technique is constraint propagation, while feasibility analyses, analysis of Kuhn–Tucker criticality, and various methods based on interval analysis also play a role. However, these will not be the primary focus of this work.

[1]Assembling the results requires communication, but this involves much less processing than each solution process. We elaborate later in this work.

quantities produced during evaluation of the objective and constraints, such that only these quantities need to be subdivided to assure good approximations by linear programs [2, 8–10, 13]. The cardinality of this subset can be viewed as the dimension by which the problem differs from being convex; when the process of [10] or [13] finds no variables needing to be subdivided (subspace dimension 0), the problem must be convex, and can be approximated arbitrarily closely by a single linear program. Even if the problem is non-convex, in some instances a high-dimensional problem has subspace dimension of 5 or less, a dimension for which it may be practical to use non-adaptive subdivision.

We first presented the non-adaptive idea, doing some initial experiments to gauge the potential practicality, in [13]. Here, we present additional algorithm details, present a careful comparison, give concrete observations and advice on when such an approach is practical, and elaborate on pitfalls and remedies.

In the remainder of this paper, we compare our proposed non-adaptive algorithm to traditional branch-and-bound software, GlobSol [12], on those problems from the COCONUT Lib1 test set [18] whose non-convex subspace dimension is 4 or less. Underlying notation appears in §2, we describe our algorithms in §3, and we present the actual experimental results in §4. In §5, we discuss aspects of both our non-adaptive discretization and GlobSol that, with additional programming effort, could improve performance with regard to execution time or quality of result. We summarize in §6.

2. Notation

We denote the general global optimization problem treated by our algorithms as follows.

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

3. Our Algorithms

In the following algorithm, based on Algorithm 3 in [13], we supply important details.

Algorithm 1 (Creating a mesh, computing an ϵ, and finding ϵ-approximate solutions, details of Algorithm 3 from [13])
INPUT*: The code list and search box $\boldsymbol{x}$ for the problem, as well as the number of subdivisions M for each non-convex variable, and a special domain tolerance ϵ_d*
OUTPUT*: A number ϵ indicating constraint satisfaction accuracy at approximate solutions, a list $\mathcal{E}$ of boxes guaranteed to contain all global optimizing points, an interval $[\underline{\varphi}, \overline{\varphi}]$ enclosing the global optimum and an associated objective discrepancy η.*

(1) *(Preprocessing)*
   a) Do the preprocessing as in Step 1 of [13, Algorithm 3], thereby computing a set $\mathcal{V}$ of cardinality N_v of variables to be subdivided, so the total number of subproblems to be handled is M^{N_v}.

b) $\epsilon \leftarrow 0$.

(2) *(Bound solutions to the subproblem for each individual box – details of Step 2 of [13, Algorithm 3])*
FOR $i = 1$ to M^{N_v}

a) Create the i-th box in the subdivision defined by $\mathcal{V}$ and the number of subdivisions.
b) Narrow the bounds on the i-th sub-box using constraint propagation.
c) Create an approximating linear relaxation.
d) Solve the linear relaxation to obtain a lower bound on the objective over the i-th box, and consider the solution to be an approximation to an optimizer.
e) IF the solution to the LP relaxation is in the i-th sub-box
THEN
Store the solution to the LP relaxation as a candidate solution x^*.
ELSE
Store the midpoint of the i-th sub-box as a candidate solution x^*.
END IF
f) IF the LP solver indicates an error return (not including infeasible or unbounded)
THEN
i. Record failure of the LP solver on box i.
ii. Set the lower bound $\underline{\varphi}_i$ on the objective over i-th sub-box to $\underline{\boldsymbol{\varphi}}(\boldsymbol{x}_I)$, the lower bound on the interval evaluation of the objective over i-th sub-box.
iii.
ELSE
Use the solution to the linear relaxation and the Neumaier–Shcherbina computation of [15] to compute a rigorous lower bound on the global optimizer over the i-th sub box, or to possibly rigorously verify that the problem is infeasible over that sub box.
END IF
g) IF the i-th subproblem is proven infeasible by the Neumaier–Shcherbina computation
THEN
Mark i-th sub-box as certainly infeasible, and set the lower bound on the global optimum over i-th sub-box to ∞.
END IF
h) IF the subproblem defined within the box i-th sub-box is not proven infeasible
THEN
i. (Determine a lower bound on the optimum over the i-th sub-box.) Take the maximum of the lower bound on the objective obtained by the Neumaier–Shcherbina computation and the lower bound obtained by interval evaluation over the box as the lower bound $\underline{\varphi}_i$.
ii. (Determine an upper bound on the optimum over the i-th sub-box)
Use a local optimizer with starting point the result of Step (2)e to compute an approximate optimum to the problem (1), subject to the parameters being in the i-th sub-box.
IF the local optimizer's return code indicates some kind of convergence

THEN

Determine whether or not the independent variables returned by the local optimizer are within ϵ_d of the interior of the box. IF they are, THEN

A. Attempt to construct a small box within the i-th sub-box about the returned x^* in which it can be proven that a feasible point exists, according to the scheme in [11] or [7, §5.2.4].

B. IF such a box can be constructed

THEN

Store the constructed box as a candidate solution $\boldsymbol{x}^*$.

ELSE

Store the entire i-th sub-box as a candidate solution $\boldsymbol{x}^*$.

END IF

ELSE

Store the entire i-th sub-box as a candidate solution $\boldsymbol{x}^*$.

END IF

END IF

iii. Evaluate the objective and constraints at the candidate solution $\boldsymbol{x}^*$, using interval arithmetic, to obtain an approximate upper bound $\overline{\varphi}_i$ on the global optimum within the i-th sub-box and to assess the infeasibility in the constraints.

iv. Distill the gap between the lower and upper bounds and the closeness of the constraints to being satisfied at the candidate solution (i.e. the solution to the LP relaxation) to single numbers:

$$\epsilon_i = \max\left\{ \max_{1\le j\le m_1} |c_j(x^*)|, \quad \max_{1\le j\le m_2} g_j(x^*) \right\},$$

$$\eta_i = \max\left\{ \overline{\varphi}_i - \underline{\varphi}_i \right\}.$$

END FOR

(3) *(Postprocessing)*

a) (Compute an overall upper bound.)

$$\overline{\varphi} \leftarrow \min_{\substack{1\le i\le M^{N_v} \\ i\text{-th sub-box not marked infeasible}}} \overline{\varphi}_i.$$

b) (Compute the approximation accuracy.)

$$\epsilon \leftarrow \max_{\substack{1\le i\le M^{N_v} \\ i\text{-th sub-box not marked infeasible, } \underline{\varphi}_i \le \overline{\varphi}}} \epsilon_i, \qquad \eta \leftarrow \max_{\substack{1\le i\le M^{N_v} \\ i\text{-th sub-box not marked infeasible, } \underline{\varphi}_i \le \overline{\varphi}}} \eta_i.$$

c) (Identify candidate boxes for actual optimizers.)

FOR $i = 1$ to M^{N_v}

IF $\underline{\varphi}_i \le \overline{\varphi}$ and the i-th sub-box is not marked infeasible

THEN

Store the i-th sub-box on a list of candidate regions for containing global minimizers.

END IF

END FOR

End Algorithm 1.

The boxes flagged in Step (3)c must contain all global optimizing points to the original problem (1).

Here are some additional details of what we did for the experiments we report in this work:

(1) We effect Step (2)b within our GlobSol environment with the routine `subsit_seeded`, a modification of the GlobSol constraint propagation routine that allows the input of tighter bounds than those computed directly from the independent variables or intermediate variables in the expression tree for the objective and constraints.
(2) In Step (2)c, we use GlobSol's routine `create_lp`. This routine supplies a constraint for each variable in the expression tree, approximating concave operations with secant lines and adaptively approximating convex operations to within a given tolerance with sets of tangent lines.
(3) In Step (2)(h)ii, we use the COIN-OR package IPOPT [20].
(4) The constraint propagation in Step (2)b is a modification of GlobSol's constraint propagation[1] to allow seeding by particular bounds on intermediate variables.
(5) Other supporting computations are from the GlobSol package.

We compare Algorithm 1 to the traditional branch-and-bound scheme in GlobSol [12]. The GlobSol algorithm appears as [1, Algorithm 2.2]. However, there have been incremental improvements to the actual implementation, including bug-fixes and extension of the process that generates the linear relaxations; these improvements could affect the results presented in [1]. A major improvement is use of the COIN-OR linear programming simplex method solver C-LP [6] instead of the old SLATEC [5] routine DSPLP; our careful examination of results using DSPLP showed us numerous failures to compute solutions to the problems GlobSol posed to it, whereas C-LP was both more reliable and faster. We have also used an improved version of IPOPT [20] to compute approximate solution to the original problems constrained to sub-boxes.

4. The Experiments

We did a preliminary analysis of the problem in the COCONUT Library 1 test set [14, 18] using the convexity analysis schemes in [10] and [13], where the scheme in [10] identifies a subset of the independent variables that needs to be subdivided to assure the linear relaxations are tight, while the scheme in [13] identifies such a subset of the intermediate variables. As in Step (1) of Algorithm 1, we took the minimum of the cardinalities of these two subsets, which we term the *reduced space dimension*, to identify the set of variables to be used in Algorithm 1, and, for Algorithm 1 to be practical within the single-processor environment of this initial exploration, we chose only those problems from the Library 1 test set such that the reduced space dimension is at most 4. We identified 83 such problems from the COCONUT Library 1 test set, from which we did our comparisons.

In our experiments, we chose $M = 20$. Also, in the module that computes underestimators to convex operations, there is a parameter ϵ_{LP} that represents the minimum relative distance between points at which underestimating tangent lines to the graph of the operation are constructed for the relaxation; the smaller ϵ_{LP},

[1] in routine `matrixop/subsit.f90`

Table 1. Results of Algorithm 1 and GlobSol for reduced space dimension 0 (convex problems).

Name	n	rd	ϵ	η	DF?	VS?	S/T	GS?	CPU-D	CPU-G	CPU/B r
sambal	17	0	0	6.50×10^{7}	F	F	1.00	F	0.07	7200.13	1.04
ex9.1.9	17	0	6.66×10^{-16}	0	F	F	1.00	F	0.02	7200.07	0.48
harker	20	0	0	1.92×10^{12}	F	F	1.00	F	0.10	7200.11	0.46
immun	21	0	0	3.84×10^{9}	F	F	1.00	F	0.04	7200.24	0.45
ex9.1.6	20	0	4.07×10^{-6}	0	F	T	1.00	F	0.05	7204.31	0.02
ex9.1.7	23	0	6.66×10^{-16}	0	F	F	1.00	F	0.02	7202.44	0.01
ex9.1.3	29	0	1.71×10^{-13}	0	F	F	1.00	F	0.07	7204.51	0.02
qp4	79	0	0	5.59×10^{2}	F	F	1.00	F	1.50	0	
qp5	108	0	0	0	F	F	1.00	F	2.26	7229.29	0.02
sample	4	0	3.49×10^{-2}	3.93×10^{4}	F	F	1.00	T	0.02	4.92	1.15

the tighter the relaxations of such operations, but the more constraints in the linear program. In our overall experiments, we chose $\epsilon_{\text{LP}} = 10^{-2}$.

In both Algorithm 1 and in GlobSol, we chose $\epsilon_d = 10^{-8}$, and we chose the initial search boxes to be as posted with the initial experiments with GlobSol on the COCONUT test sets (such as in [1, 18]), that is, with bounds of -10^4 or 10^4 corresponding to bound constraints missing in the original problem. Also, actual bound constraints were posed as inequality constraints, to avoid problems with exponential processes in the "peeling" process [7, Algorithm 15, page 191], and a slight modification to GlobSol was made to allow this. In cases of excessive computation, we terminated GlobSol's branch and bound algorithm after 7200 CPU seconds or 500,000 boxes, whichever occurs first. Additional configuration parameters used in GlobSol's branch and bound algorithm for these experiments are available through a URL the first author can communicate.

The experiments were done on a Dell 1737 laptop with a Core 2 Duo chip, 2 processors each running at 2.5Ghz, and 4GB of memory[1]. The operating system was Ubuntu 11.04 with corresponding Fortran and C/C++ compilers from the `gcc` suite (gcc version 4.5.2). C-LP and IPOPT were compiled with the default build procedure supplied from COIN-OR, while the GlobSol package, including Algorithm 1, was compiled with optimization level 0 and with debugging hooks.

In Algorithm 1, the total number of boxes is fixed by M and the reduced space dimension, but the size of the linear relaxations (although extremely sparse) depends linearly on the number of operations and on ϵ_{LP}.

Results for reduced space dimension 0 (i.e. convex problems) appear in Table 1, results for reduced space dimensions 1 and and 2 appear in Table 2, while results for reduced space dimensions 3 and 4 appear in Table 3. The columns of these tables are labeled as follows:

- **Name** is the problem identification as given on the COCONUT web page for Library-1.
- n is the number of variables in the original problem.
- **rd** is the reduced space dimension.
- ϵ and η are is as in Algorithm 1.
- **DF?** is "T" if, for some box, Algorithm 1 failed due to a general failure in the LP solver (other than an indication of possible infeasibility or unboundedness).
- **VS?** is "T" if verification of feasibility near the approximate solution succeeded for at least one box in the discretization.

[1]Memory was not an issue in GlobSol's branch and bound algorithm, but was a limiting factor in preliminary experiments with Algorithm 1 with larger reduced space dimensions and M.

Table 2. Results of Algorithm 1 and GlobSol for reduced space dimensions 1 and 2.

Name	n	rd	ϵ	η	DF?	VS?	S/T	GS?	CPU-D	CPU-G	CPU/B r
ex4.1.4	1	1	0	1.42×10^{0}	F	T	0.25	T	0.18	0.24	4.84
ex4.1.6	1	1	0	3.82×10^{1}	F	T	0.20	T	0.21	0.22	5.58
rbrock	2	1	0	2.25×10^{-14}	F	T	0.05	T	0.28	0.02	1.40
ex4.1.8	2	1	2.21×10^{-8}	2.29×10^{-4}	F	T	0.05	T	0.10	0.01	2.00
circle	3	1	6.02×10^{-7}	4.53×10^{0}	F	T	0.05	T	1.60	0.04	10.00
ex2.1.4	6	1	1.33×10^{-15}	0	F	F	1.00	T	0.16	1.17	0.38
ex4.1.7	1	1	0	9.05×10^{-2}	F	T	0.10	T	0.16	0.04	3.20
ex4.1.3	1	1	0	2.44×10^{1}	F	T	0.10	T	0.17	0.09	2.08
ex14.1.9	2	1	7.43×10^{-9}	1.00×10^{4}	F	T	0.40	T	0.40	1.09	1.72
ex4.1.9	2	1	1.43×10^{-5}	3.82×10^{0}	F	T	0.15	T	0.16	0.64	4.38
ex4.1.2	1	1	0	5.91×10^{1}	F	T	0.10	T	0.46	0.34	1.22
ex4.1.1	1	1	0	3.93×10^{1}	F	T	0.10	T	0.24	0.11	3.60
ex8.1.2	1	1	0	1.60×10^{0}	F	T	0.50	T	0.24	0.24	2.50
ex7.3.3	5	1	5.33×10^{-15}	1.00×10^{4}	F	F	1.00	T	0.25	0.16	1.64
haverly	12	1	1.71×10^{-13}	1.17×10^{4}	F	F	1.00	T	6.83	3307.36	14.40
ex8.4.1	22	1	9.54×10^{-8}	1.53×10^{-1}	F	T	0.05	F	0.20	7200.29	0.03
least	3	1	0	7.40×10^{5}	F	T	1.00	T	2.20	754.51	2.80
ex8.1.3	2	1	0	6.75×10^{35}	F	T	0.10	F	3.26	0	
ex14.1.5	6	1	3.50×10^{-8}	1.00×10^{4}	F	T	0.10	T	0.28	1.44	1.57
ex8.1.6	2	2	0	1.50×10^{1}	F	T	0.00	T	7.96	0.57	1.01
ex8.1.4	2	2	0	3.36×10^{11}	T	T	0.10	T	6.28	0.45	1.95
ex4.1.5	2	2	2.50×10^{-8}	5.03×10^{9}	T	T	0.05	T	6.33	0.47	2.63
ex14.1.4	3	2	1.45×10^{-8}	1.00×10^{4}	F	T	0.12	T	3.36	7.68	2.00
ex5.2.2-3	9	2	5.43×10^{-6}	4.22×10^{3}	F	T	0.02	T	0.62	592.43	0.10
ex5.2.2-2	9	2	3.64×10^{-6}	9.70×10^{3}	F	T	0.09	F	3.10	5189.81	0.75
ex5.2.2-1	9	2	3.64×10^{-6}	4.38×10^{3}	F	T	0.04	T	0.66	970.68	0.17
ex8.1.1	2	2	1.00×10^{-8}	4.11×10^{-2}	F	T	0.01	T	2.41	0.06	0.80
ex8.1.5	2	2	0	1.12×10^{11}	T	T	0.01	T	6.45	1.59	2.81
ex14.1.3	3	2	7.55×10^{-5}	1.00×10^{4}	F	T	0.10	T	1.09	17.96	0.50
ex7.3.2	4	2	1.25×10^{-8}	1.00×10^{4}	F	T	0.01	T	3.56	0.04	3.78
ex14.1.8	3	2	8.88×10^{-16}	1.00×10^{4}	F	F	1.00	T	7.88	0.58	2.99
ex14.1.1	3	2	2.67×10^{-15}	1.00×10^{4}	F	F	1.00	T	4.70	10.65	2.27
ex8.1.8	6	2	8.01×10^{-7}	4.02×10^{-1}	F	T	0.07	T	2.99	9.22	0.89
ex7.2.2	6	2	8.01×10^{-7}	4.02×10^{-1}	F	T	0.07	T	3.40	9	1.04
ex5.2.4	7	2	4.23×10^{-6}	2.14×10^{3}	F	T	0.10	T	4.31	27.8	1.76
ex6.1.2	4	2	1.65×10^{-8}	7.80×10^{-1}	F	T	0.05	T	0.34	0.49	0.18
ex8.2.4	55	2	0	∞	T	F	0.36	F	601.15	7287.53	0.06

- **S/T** is the ratio of number of solution boxes returned by the discretization to the total number of boxes returned; this is a measure of the effectiveness of rejecting non-solution-containing boxes.
- **GS?** is "F" if GlobSol's branch and bound algorithm failed due to exceeding the limit on CPU or number of nodes[1].
- **CPU-D** is the total processor time spent in Algorithm 1.
- **CPU-G** is the total processor time spent in GlobSol's branch and bound algorithm.
- **CPU/B r** is the ratio of two ratios — the average processor time per box for Algorithm 1 to the average processor time per box for GlobSol's branch and bound algorithm.

We now analyze these results.

[1]with two exceptions; see the note to follow

Table 3. Results of Algorithm 1 and GlobSol for reduced space dimensions 3 and 4.

Name	n	rd	ϵ	η	DF?	VS?	S/T	GS?	CPU-D	CPU-G	CPU/B r
ex7.2.6	3	3	6.49×10^{-9}	4.24×10^{1}	F	T	0.00	T	19.63	0.14	0.81
ex3.1.4	3	3	1.33×10^{-15}	9.50×10^{-1}	F	T	0.00	T	3.30	4.62	0.06
ex9.2.8	6	3	0	3.00×10^{0}	F	F	0.05	T	0.72	18.62	0.05
house	8	3	1.36×10^{-12}	3.94×10^{3}	F	F	0.00	F	0.10	7200.07	0.00
mhw4d	5	3	1.03×10^{-7}	6.10×10^{12}	F	T	0.00	T	5.16	2.22	0.06
ex8.1.7	5	3	0	4.69×10^{1}	F	F	0.00	F	19.83	5527.09	0.22
dispatch	4	3	1.14×10^{-6}	1.06×10^{-2}	F	T	0.00	T	102.51	0.16	0.88
ex14.2.2	4	3	5.68×10^{-14}	1.00×10^{4}	F	F	0.10	T	7.59	5.6	0.28
ex14.1.2	6	3	2.24×10^{-4}	1.00×10^{4}	F	T	0.12	F	29.21	1694.68	0.27
nemhaus	5	3	5.00×10^{-9}	1.49×10^{-13}	F	T	1.00	T	14.45	0	
ex14.2.5	4	3	1.14×10^{-13}	1.00×10^{4}	F	F	0.10	T	8.40	6.13	0.24
ex14.2.8	4	3	4.26×10^{-14}	1.00×10^{4}	F	F	0.10	T	14.76	1082.66	0.27
ex6.1.4	6	3	1.97×10^{-8}	1.67×10^{0}	F	T	0.01	T	5.92	4.41	0.11
ex14.2.9	4	3	5.68×10^{-14}	1.00×10^{4}	F	F	0.10	T	10.08	503.59	0.17
ex7.3.1	4	3	3.20×10^{-2}	1.00×10^{4}	F	T	0.00	T	3.93	1.57	0.03
ex7.3.5	13	3	7.11×10^{-15}	1.00×10^{4}	F	F	0.01	F	44.30	7200.03	0.16
ex8.4.4	17	3	1.19×10^{-7}	5.25×10^{0}	F	T	0.00	T	48.63	1096.22	0.15
ex6.2.8	3	3	1.50×10^{-8}	1.31×10^{1}	F	T	0.05	T	6.96	34.67	0.12
ex6.2.6	3	3	1.50×10^{-8}	1.41×10^{1}	F	T	0.05	T	7.04	71.55	0.08
ex6.2.11	3	3	1.50×10^{-8}	6.02×10^{1}	F	T	0.04	T	7.80	98.19	0.08
ex9.2.4	8	4	2.27×10^{-13}	1.99×10^{4}	F	F	0.00	F	0.82	7248.23	0.00
ex9.1.2	10	4	0	1.70×10^{1}	F	F	0.00	F	2.54	7200.08	0.00
ex9.1.4	10	4	1.71×10^{-13}	5.90×10^{1}	F	F	0.00	F	4.01	7200.25	0.00
ex2.1.3	13	4	3.00×10^{-8}	1.70×10^{1}	F	T	0.00	T	6107.09	2580.4	0.77
ex7.2.7	4	4	8.33×10^{-17}	6.07×10^{0}	F	T	0.00	T	46.63	2.98	0.02
ex3.1.2	5	4	3.90×10^{-7}	6.27×10^{2}	F	T	0.00	T	856.01	0.24	0.54
himmel11	9	4	6.83×10^{-7}	3.39×10^{3}	F	T	0.00	T	1743.99	32.31	0.47
ex14.2.1	5	4	4.26×10^{-14}	1.00×10^{4}	F	F	0.07	F	271.00	5140.6	0.16
ex5.3.2	22	4	1.71×10^{-13}	6.04×10^{0}	F	F	0.00	F	5.38	0	
ex14.2.4	5	4	5.68×10^{-14}	1.00×10^{4}	F	F	0.07	T	197.24	2447.16	0.14
ex14.2.6	5	4	5.68×10^{-14}	1.00×10^{4}	F	F	0.07	F	213.60	6766.07	0.10
ex6.1.1	8	4	1.53×10^{-8}	9.29×10^{-1}	F	T	0.01	T	25.75	347.42	0.01
ex8.4.5	15	4	3.76×10^{-8}	4.34×10^{-3}	F	T	0.00	T	78.11	1312.81	0.01
ex6.2.12	4	4	1.00×10^{-8}	6.87×10^{0}	F	T	0.01	T	33.19	91.15	0.01
ex6.2.14	4	4	1.00×10^{-8}	3.70×10^{0}	F	T	0.00	T	26.47	35.9	0.01
wall	6	4	0	1.50×10^{4}	F	F	0.00	T	0.76	0.66	0.00

4.1. *Failure modes*

The following have been observed to happen.

(1) The approximate solver (IPOPT in these experiments) can fail to compute a good approximate optimizer, something we occasionally observed when we examined particular problems. This results in an overly pessimistic $\overline{\varphi}$.
(2) More often than (1), the feasibility verification in Step (2)(h)iiA of Algorithm 1 can fail, leading to an overly pessimistic $\overline{\varphi}$. In fact the process did not succeed in any of the sub-boxes generated by Algorithm 1 for 31 of the 83 problems, and notably only succeeded for one of the convex problems. (See column "VS?" of Tables 1, 2, and 3.) This is due to the fact that the verification process requires a non-singular matrix of gradients of active constraints, something that often is not true. See the note following this list for a discussion.
(3) When running the GlobSol branch and bound algorithm, we observed ex-

cessive time ("hanging") or segmentation faults in C-LP for `qp4`, `ex8.1.3`, and `ex5.3.2`, and data for these problems are therefore absent for GlobSol's branch and bound algorithm. The calling sequences, etc., were carefully checked in GlobSol's branch and bound algorithm, such problems did not occur in Algorithm 1, and such problems did not happen elsewhere in GlobSol's branch and bound algorithm.

Regarding failure (2), we examined closely several of the problems on which this failure occurred, and discovered the solution to the linear relaxations from Step (2)e and the approximate solution to the original problem obtained in Step (2)(h)ii were close, and the constraints for the original problem were satisfied, or very nearly satisfied, at the approximate solution from Step (2)(h)ii. Thus, good lower and upper bounds on φ were seemingly obtained, although not provably so with the feasibility verification scheme from [11]. However, the only alternative we presently know for which all solutions to a problem are rigorously enclosed is to relax equality and active inequality constraints, that is, replace Problem (1) by a relaxed problem, then show that these relaxed constraints are satisfied at the point returned by the approximate optimizer, e.g., through an interval evaluation at that point. However, if such a scheme were used to obtain $\overline{\varphi}$, the solution boxes would be near optimizing points of the original problem only if the constraints are continuous and the relaxation is sufficiently tight. Furthermore, the relaxation would need to be used not only in computing $\overline{\varphi}$, but also in forming the linear program and in the constraint propagation. This is because the optimum of the original problem is in general larger than the optimum of the relaxation, and the constraint propagation or lower bound computation may result in some or all of the boxes containing solutions to the relaxation being rejected. In fact, when we implemented relaxed computation of $\overline{\varphi}$ but not relaxed constraints in the linear relaxations and constraint propagation, we observed loss of some or all solutions, both in Algorithm 1 and GlobSol's branch and bound algorithm, in a significant number of problems.

Some specific failures and exceptions are:

(1) In `ex8.2.4`, wide initial ranges (with initial variable bounds absent) and exponentials of these large variables in the objective and constraints led both to inability of interval evaluations to compute meaningful interval evaluations and failure return of C-LP.
(2) The problem `nemhaus` as posed in the COCONUT Library 1 set was exceptional, since the initial bounds have all lower bounds equal to upper bounds; thus every box is optimal, and subdivision was meaningless, and Algorithm 1 both is successful and makes no progress (other than proving feasibility).

4.2. *Notable Conclusions*

(1) As evidenced in the "ϵ" column of Tables 1, 2, and 3, IPOPT was remarkably good at finding approximately feasible points, even though verifying exact feasibility often failed.
(2) As evidenced by η, the choice $M = 20$ and $\epsilon_{\mathrm{LP}} = 10^{-2}$ was insufficient to obtain reasonable bounds on the global optimum in most of the problems, with $\eta < 0.1$ achieved in only 13 of the 83 cases. However, when we experimented with increasing M and decreasing ϵ_{LP} on several particular problems with large η, we found it practical to reduce η to acceptable bounds. See the note following this list.
(3) Even when sharp bounds on the global optimum could not be verified,

Algorithm 1 was effective at eliminating a large portion of the search region. This is evidenced in the column "S/T," an upper bound for the proportion of the original parameter bound space that is rejected[1]. In fact, more than 3/4 of the region was rejected in 62 of the 73 non-convex problems. This suggests Algorithm 1 may be used as a preprocessing algorithm for a more sophisticated branch and bound process.

(4) The GlobSol branch and bound algorithm failed to complete within its given time and iteration limits in 23 of the 83 cases. Of these 23 GlobSol branch and bound failures, Algorithm 1 led to at least a 75% reduction in search region volume in at least 13 cases, and resulted in small η in 5 additional cases; i.e. Algorithm 1 gave useful results in 18 of the 23 cases in which the GlobSol branch and bound algorithm failed. This further suggests a preprocessing or complementary role for Algorithm 1.

(5) There isn't a clear pattern whether GlobSol's branch and bound algorithm, which potentially does a more eclectic mix of processing per box, spends more time per box than Algorithm 1. Averaged over all 83 problems, Algorithm 1 spends 1.25 times as much per box as the branch and bound algorithm, but spends less than half as much time per box on 48 of the 83 problems, while the branch and bound algorithm spends less than half as much time per box in only 16 of the 83 problems. The branch and bound algorithm may be spending less time because it is able to fathom large numbers of small boxes with little processing.

(6) With $\epsilon_{\mathrm{LP}} = 10^{-2}$ and $M = 20$, the average CPU time for these 83 problems was much less for Algorithm 1 than that for GlobSol's branch and bound algorithm. In fact, in no case did the total time for Algorithm 1 exceed the limits set for the branch and bound algorithm. Although the time for Algorithm 1 exceeded that for the branch and bound algorithm in 28 of the 80 cases for which the branch and bound algorithm either ran to completion or ended due to resource limits reached, the total CPU time for all 83 problems was only about 7.2% of that taken by GlobSol's branch and bound algorithm.

Regarding conclusion (2), we observed in cases for which feasibility could not be verified that the lower bound verified from the linear relaxation and the approximate upper bound returned from IPOPT were close. For example, in sambal, feasibility could not be verified, but the coordinates of the lower and approximate upper bound were within approximately 1% of each other. In cases in which feasibility could be verified, we could adjust ϵ_{LP} (for the convex problems), or ϵ_{LP} and M (for non-convex problems) to obtain reasonable η. However, in such cases, we found boxes which could not be proven infeasible and for which the result returned by IPOPT could not be verified feasible. For such problems, the bounds on φ may depend on simple interval evaluations, and convergence may be slow. For example, in `ex5.2.2.case3`, increasing M from 20 to 800 decreased η from 4.22×10^3 to 1.72×10^3, while the CPU time increased from 0.02 to 1715, and decreasing ϵ_{LP} had little effect.

4.3. *Additional Observations*

We observed the following from informal experimentation following study of the formal results in Tables 1, 2, and 3.

[1]For the convex problems, this ratio is 1, since only one box total was considered.

(1) While memory usage in GlobSol's branch and bound algorithm is modest, memory becomes an issue on a laptop or desktop PC for larger M and reduced space dimensions.
(2) The solution time for the linear relaxations increases as ϵ_{LP} is decreased. This can be significant for some problems, but numerical problems in the linear program solver due to ill-conditioning, or removal of constraints during preprocessing, may be a more important factor for smaller ϵ_{LP}.

4.4. *An Overall Conclusion*

It is apparent that a major impediment, both in GlobSol's branch and bound algorithm and in Algorithm 1, is verification of feasibility, something that may be removed by relaxing the equality constraints and the binding inequality constraints. We have left implementation of such a scheme (including also relaxing the original constraints appropriately in the constraint propagation and linear relaxation) to future work.

5. Limitations, Alternatives, and Improvements

The following additional work may further the range of applicability of both GlobSol's branch and bound algorithm and Algorithm 1.

- *Fully implement the constraint relaxation within the GlobSol environment:* As we indicated in §4.1 and §4.4, a major obstacle to practicality, both in the branch and bound algorithm and Algorithm 1, is failure to obtain a mathematically rigorous upper bound from solutions returned by the approximate optimizer, despite the fact that these solutions are of high quality. Replacement of problem 1 by relaxing the constraints would allow feasibility verification of the relaxed problem by simple interval evaluations, but changes need to be made in both the linear relaxation generation process and the constraint propagation to enable mathematically rigorous enclosure of all solutions to the relaxed problem.
- *Implement a better subdivision process for convex constraints within the linear relaxation generation process:* The process used here in Step (2)c of Algorithm 1 essentially supplies a uniform relative distance between the points at which tangent underestimators are constructed for convex operations. Tighter relaxations with less underestimators are obtainable if new points are adaptively placed where bracketing underestimators intersect. Although we had this technique in the experiments in [10], the implementation utilized a Newton method which sometimes failed. Making this method robust by combining it with utilization of the midpoint in cases of failure could improve both GlobSol's branch and bound algorithm and Algorithm 1. Furthermore, the third author of this paper is working on techniques by which a similar subdivision process may be applied to the non-convex variables, thus also reducing the total number of problems that need to be solved, for a given approximation accuracy.
- *Study parallelization:* Depending on practical problems we encounter, running Algorithm 1 within a massively parallel environment may be effective. Note that each iteration of Step (2) is totally independent of the others, and almost all work, especially for larger numbers of parameters, is done in this step. Algorithm 1 could also benefit, with larger M or reduced space dimension, from machines with a large amount of memory.
- *Supply an improved reduced space algorithm in GlobSol's branch and bound algorithm:* Although we did initial experiments in [10], as have Epperly et al in

[2], we hadn't fully integrated the process into the constraint propagation and interval Newton methods.

Another alternative, investigated in [16, 17], uses relaxed constraints and a clever algorithm to efficiently produce tight bounds on the optimum, with small η given a priori, or a posteriori in the case of limited memory. In contrast to this work, the linear relaxations, although dense, have a size proportional only to the number of original variables n and numbers of original constraints m_1 and m_2, and not to the total number of operations in evaluating the objective and constraints. They have shown their approach, based on affine arithmetic, to be highly competitive for a number of practical problems. However, in addition to solving a relaxation of Problem 1, the algorithms they describe guarantee enclosure of only one optimizing point.

6. Summary and Availability

This work has focussed on the effectiveness of approximation of global optimization problems by linear relaxations in relative isolation, but in a context of mathematically rigorous verification. Doing so, we have explored the possibilities for a simplified but highly parallelizable algorithm, appropriate for problems that are "almost convex" in a sense we have defined. We also have been able to study and draw conclusions concerning power of the linear relaxations and the approximate solvers within this simplified environment.

Since we are now using publicly re-distributable software from the COIN-OR project, we hope to package the version of GlobSol used here soon for public distribution, perhaps within COIN-OR; it is presently available through an SVN repository by contacting the first author of this work, along with our implementation of Algorithm 1.

References

[1] Ralph Baker Kearfott. Discussion and empirical comparisons of linear relaxations and alternate techniques in validated deterministic global optimization. *Optimization Methods and Software*, 21:715–731, October 2006.

[2] T. G. W. Epperly and E. N. Pistikopoulos. A reduced space branch and bound algorithm for global optimization. *J. Global Optim.*, 11(3):287–311, October 1997.

[3] Christodoulos Floudas. *Deterministic Global Optimization: Theory, Methods, and Applications*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

[4] Christodoulos Floudas. Deterministic global optimization: Advances and challenges, June 2009. Plenary Lecture, First World Congress on Global Optimization in Engineering and Sciences, WCGO-2009.

[5] Kirby Fong, Thomas Jefferson, Tokihiko Suyehiro, and Lee Walton. Guide to the SLATEC common mathematical library. Technical report, `netlib.org`, April 1990. See `http://www.netlib.org/slatec/`.

[6] Julian Hall. Homepage of C-LP, 2002. `https://projects.coin-or.org/Clp`.

[7] R. Baker Kearfott. *Rigorous Global Search: Continuous Problems*. Number 13 in Nonconvex optimization and its applications. Kluwer Academic Publishers, Norwell, MA, USA, and Dordrecht, The Netherlands, 1996.

[8] R. Baker Kearfott. Erratum: Validated linear relaxations and preprocessing: Some experiments. *SIAM J. Optim.*, 2011.

[9] R. Baker Kearfott. Interval computations, rigour and non-rigour in deterministic continuous global optimization. *Optim. Methods Softw.*, 2011. (to appear).

[10] R. Baker Kearfott and Siriporn Hongthong. Validated linear relaxations and preprocessing: Some experiments. *SIAM J. on Optimization*, 16(2):418–433, 2005.

[11] Ralph Baker Kearfott. On proving existence of feasible points in equality constrained optimization problems. *Math. Program.*, 83(1):89–100, 1998.

[12] Ralph Baker Kearfott. GlobSol user guide. *Optimization Methods and Software*, 24(4–5):687–708, August 2009.

[13] Ralph Baker Kearfott, Jessie Castille, and Gaurav Tyagi. A general framework for convexity analysis and an alternative to branch and bound in deterministic global optimization. *Journal of Global Optimization*, 2011. submitted for a special issue

in honor of Pierre Hansen, from the 2010 Toulouse Global Optimization Workshop, http://interval.louisiana.edu/preprints/2010-alternative-GO-paradigm.pdf.

[14] Arnold Neumaier. COCONUT Web page, 2001-2003. http://www.mat.univie.ac.at/~neum/glopt/coconut.

[15] Arnold Neumaier and Oleg Shcherbina. Safe bounds in linear and mixed-integer programming. *Math. Prog.*, 99(2):283–296, March 2004.

[16] Jordan Ninin. *Optimisation Globale Basée sur l'Analyse d'Intervalles: Relaxations Affines et Techniques d'Accélération*. Ph.D. dissertation, Université de Toulouse, Toulouse, France, December 2010.

[17] Jordan Ninin and Frédéric Messine. A metaheuristic methodology based on the limitation of the memory of interval branch and bound algorithms. *J. Glob. Optim.*, February 2010.

[18] 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.

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

[20] A. Wächter. Homepage of IPOPT, 2002. https://projects.coin-or.org/Ipopt.