Adding Interval Constraints to the Moore-Skelboe Global Optimization Algorithm

H.M. Chen and M.H. van Emden

***Abstract*— Interval constraints are a promising addition to global optimization with interval analysis [8, 5]. We explain the principle of interval constraints and the application of this method to global optimization. We then show how the Moore-Skelboe global optimization algorithm can be made more efficient by adding interval constraints. This yields a simple algorithm in BNR Prolog [2, 1] and achieves a factor of two to five in function evaluations on the unconstrained optimization examples of [8].**

I. Introduction

Although interval constraints, as a method for solving numerical problems, has much in common with interval arithmetic, there are some interesting differences between the two approaches. In this paper we show that one of these differences can be used to improve the Moore-Skelboe algorithm [8] for global optimization. This algorithm is unique in nonlinear optimization by giving an interval for a global minimum while not requiring differentiability, or even continuity, in the objective function.

We briefly introduce interval constraints and explain by means of an example how this method applies to global optimization. We then present our modification of the Moore-Skelboe algorithm followed by a discussion of its performance on the unconstrained global optimization examples in [8].

II. Constraint Processing

Constraint processing [3, 10] is the general approach in computer problem solving where one expects the problem to be solved by merely entering the constraints between the variables of the problem without the need for any algorithm in addition to a general-purpose constraint solver that comes with the system. Of course, in interesting problems the number of constraints is large enough to require an algorithm to generate them. This is also the case in the global optimization problems on which we report in this paper.

A common form of constraints are *range* constraints: where the constraint solver updates the sets of possible values of the variables in view of the relations constraining these values. In problems where the variables range over the real numbers, an efficient way to represent such sets of possible values is as intervals with floating point numbers as bounds. The primitive constraining relations include the usual numerical relations $=$, $\leq$, $\geq$, and so on. They also include the sum relation ($\mathrm{sum}(x, y, z)$ iff $x + y = z$), the product relation ($\mathrm{prod}(x, y, z)$ iff $x \times y = z$) and the exponentiation relation ($\exp(x, n, y)$ iff $x^n = y$ for positive integer exponents).

Typical constraint-solving software allows new constraints to be defined in terms of primitive or previously defined constraints.

III. The Consistency Method for Interval Constraints

The consistency method consists in removing from the intervals of some of the variables certain values that are not consistent with the constraints. It was first used in discrete combinatorial problems [7, 11, 6]. Here is an example of how it works with interval constraints.

Suppose that initial conditions or other constraints determine that the values for the variables x, y, and z have to lie in the intervals $[3, 5]$, $[1, 3]$, and $[2, 5]$. It is then clear that all three intervals contain values for their variables that are not consistent with, for instance, the constraint $x + y = z$. For example, z has to be at least 4. Interestingly, the constraint not only causes the interval for z to change, but also those for x and y. As a result, the constraint causes the above intervals to narrow to $[3, 4]$, $[1, 2]$, and $[4, 5]$. Thus the *sum* constraint includes the addition operation and its inverse. Similarly for the product constraint.

Such considerations may also yield the conclusion that an interval is empty. In that case, when the usual precaution of outward rounding is observed, a proof of nonexistence of a solution is obtained.

The authors are with the University of Victoria, Victoria, B.C., Canada. The authors gratefully acknowledge support from the Natural Science and Engineering Research Council of Canada.

In this way, interval constraints shares with interval analysis the property of being able to find complete sets of solutions to nonlinear equations as well as global optima.

The purpose of computation in the consistency method is to use all constraints to narrow as much as possible the intervals for all of the variables. The narrower the intervals, the more information they convey about the solution, if one exists. It is quite common to obtain intervals that are not much wider than is allowed by the precision of the floating point system. It is also quite common to obtain intervals that are so wide as not to contain any useful information about possibly existing solutions. In the application reported here we obtained intervals not exceeding the tolerances specified.

We have used the BNR Prolog system [2, 1], which embeds interval constraints and a consistency algorithm in the Prolog language [4, 9].

IV. Examples of the Role of Interval Constraints in Global Minimization

The Moore-Skelboe algorithm for global minimization subdivides the area of interest in successively smaller boxes (i.e. Cartesian products of intervals) and computes an interval for the objective function over each of these. During the execution of the algorithm a list of disjoint boxes exists of which the union is guaranteed to contain the global minima. At any time during execution we know that global minima have to lie between the lowest upper bound in the list and the lowest lower bound. The algorithm terminates as soon as these bounds differ less than a predetermined tolerance.

We have programmed in BNR Prolog a close counterpart of the Moore-Skelboe algorithm. With the algorithm as stated in [8] we were unable to avoid memory overflow in the examples cited. But with some obvious and minor modifications, but without using constraints, we obtained results much like the ones reported in [8]; see the table below.

The Moore-Skelboe algorithm only uses information about the objective function obtained by evaluation of an inclusion function for it over each of the boxes in the list. With interval constraints we can obtain more information about the values of the objective function. By including such constraints in the Moore-Skelboe algorithm, we increase its effectiveness. Before discussing the algorithm, we show by an example how interval constraints can give such additional information.

To find the global optimum with interval constraints, the objective function is translated to a set of primitive constraints. For example, suppose we wish to find the global minimum of

$$f(x) = x_1^2(4+x_1^2(-2.1+(1/3)x_1^2))+4x_2^2(x_2^2-1)+x_1x_2.$$

This is example 3 on page 93 of Ratschek and Rokne [8]. The BNR Prolog system for interval constraints allows one to enter the definition of the relation `f` defined as

```
f(X1,X2,Y)
:- Y is X1**2*(4+X1**2*(-2.1+X1**2/3))
   +4*X2**2*(X2**2-1)+X1*X2.
```

Read this as: "The relation `f` holds between `X1`, `X2`, and `Y` if `Y` equals the polynomial shown." A constraint containing an arithmetic expression is translated to an equivalent set of primitive constraints, typically introducing auxiliary variables.

We continue with example 3 from Ratschek and Rokne. They start with the independent variables in the intervals $[-2.5, 2.5]$ and $[-1.5, 1.5]$. After suitable type declarations, we can now enter the BNR Prolog command

```
-2.5 =< X1,X1 =< 2.5, -1.5 =< X2,X2 =< 1.5,
 f(X1,X2,Y)
```

using the definition for `f` and setting up constraints for `X1` and `X2` as shown. By leaving `Y` initially unconstrained, it gets an interval of possible values equal to the result of an inclusion function [8] or interval extension [5]. In this example it is $[-70, 40]$.

Now, BNR Prolog allows one to enter additional constraints. So why not probe the large interval for `Y` obtained from the inclusion function by means of a constraint on `Y`? In this way we find, for example, that `Y` cannot be less than -13, because the constraint system

```
-2.5 =< X1,X1 =< 2.5, -1.5 =< X2,X2 =< 1.5,
 f(X1,X2,Y), Y < -13
```

is found not to have any solutions. This establishes -13 as a lower bound for the global minimum of f within the given intervals for x_1 and x_2. When we enter instead -12, the constraint system leaves open the possibility of solutions. This points up an important limitation of the consistency method: all it does is remove inconsistent values. Hence the *absence* of solutions can be reported with certainty; failure to determine absence of solutions does not imply the *presence* of a solution. But of course the fact that f is everywhere defined on the given initial intervals for x_1 and x_2 allows us to obtain an upper bound to the global minimum by arbitrarily selecting point intervals for x_1 and x_2 and again invoking the interval extension of f. This is done in the following BNR Prolog command, where we choose the midpoints of the previous intervals as the arbitrary point intervals for X1 and X2:

```
X1 == 1.23456789, X2 == 0.987654321,
f(X1,X2,Y).
```

which results in the interval

$$[3.52202582505467, 3.52202582505469]$$

for Y. The fact that f is defined for these x-values implies that the upper bound of the interval for Y is an upper bound for the global minimum. Thus the interval constraint system has improved the interval $[-70, 40]$, obtained from the inclusion function used, to, say, $[-13, 4]$. The Moore-Skelboe algorithm relies on the intervals given by an inclusion function. Our modification is to use instead an improved interval by the same method as in this example.

Trying the above command with `Y < -13` was of course a lucky guess. The point is, that with any number equal to about -13 or less, a single constraint evaluation can establish it as a lower bound. This is what we do in our modification of the Moore-Skelboe algorithm.

To further improve the results we have to find a systematic way for subdividing the initial search space. For that we adopt the method of the Moore-Skelboe algorithm.

V. Modification of the Moore-Skelboe Algorithm

In our version of the Moore-Skelboe algorithm, we exploit the fact that in BNR Prolog all computation is performed via interval constraints, so that the extra constraint on the value of the objective function, as discussed above, can be inserted easily.

The Modified Algorithm

input:
(1) a box $X = (X_1, \cdots, X_m)$,
where $X_i = [a_i, b_i] (i = 1, \cdots, m)$.
(2) an inclusion function F of f.
output:
$L = ((V_1, y_1), \cdots, (V_n, y_n))$, where
V_i is a box, $y_i = F(V_i), i = 1, \cdots, n$,
$lb(y_1) \leq lb(y_2) \leq \cdots \leq lb(y_n)$.

1:
$y = F(X)$; $L = ((X, y))$;
$f_{min} = ub(F(c))$, where $c = mid(X)$.
2:
remove the 1st element (V, y) of $L = ((V, y), \cdots)$,
where $V = (Z_1, \cdots, Z_m)$.
3:
find Z_k such that
$width(Z_k) = max(width(Z_i))(i = 1, \cdots, m)$;
bisect Z_k into Z_{k1}, Z_{k2}, and obtain subboxes
$V_1 = (Z_1, \cdots, Z_{k-1}, Z_{k1}, Z_{k+1}, \cdots, Z_m)$
$V_2 = (Z_1, \cdots, Z_{k-1}, Z_{k2}, Z_{k+1}, \cdots, Z_m)$
4:
$v_1 = F(V_1)$; $v_2 = F(V_2)$;
enter $(V_1, v_1), (V_2, v_2)$ into L such that the lower bounds of the 2nd members of the pairs of L do not decrease.
5:
$f_{min} = min(f_{min}, ub(F(c_1)), ub(F(c_2)))$,
where $c_1 = mid(V_1), c_2 = mid(V_2)$.
for each (V, v) of L,there are two different cases:
(1) discard it if interval constraint $v \leq f_{min}$ can not be consistently added to constraint system;
(2) keep it in L and add $v \leq f_{min}$ to constraint system if $v \leq f_{min}$ can be consistently added.
6:
if termination criteria hold then halt
else goto step 2.

VI. Computational Results

The above algorithm was tested on the three examples used by Ratschek and Rokne for testing the Moore-Skelboe algorithm. To compare our modification with the original version, we implemented both in BNR-Prolog. The following tables show the results. Here follows an explanation of the symbols used in the table.

$D_1 \times D_2$	Initial box (Step 1 of the algorithm).
ϵ	Intended absolute accuracy (Termination criterion(3.6)[8]).
N	Number of inclusion function evaluations till termination when Moore-Skelboe algorithm is used.
N'	Number of inclusion function evaluations till termination when the modified algorithm is used.
y^*	Lower bound of the global minimum.
(X^*, y^*)	Leading pair at termination, $X^* = X_1{}^* \times X_2{}^*$.

Example 1

Input	D_1	[-100,100]
	D_2	[-100,100]
	ϵ	1.0e-6
Moore-Skelboe	N	143
	$y*$	
		-9999.0000167
	X_1*	[-100,-99.9999]
	X_2*	[-100,-99.9999]
Modified Moore-Skelboe	N'	57
	$y*$	-9999.000015693
		-9999.000016667
	X_1*	[-100,-99.9999]
	X_2*	[-100,-99.9999]
Comparison	N/N'	2.5

Example 2

Input	D_1	[0,1]
	D_2	[0,1]
	ϵ	1.0e-6
Moore-Skelboe	N	81
	$y*$	
		1.0
	X_1*	[0,9.5367e-7]
	X_2*	[0,9.5367e-7]
Modified Moore-Skelboe	N'	37
	$y*$	1.00000055190
		1.0
	X_1*	[0,5.5189e-7]
	X_2*	[0,4.6928e-7]
Comparison	N/N'	2.1

Example 3

Input	D_1	[-2.5,2.5]
	D_2	[-1.5,1.5]
	ϵ	1.0e-1
Moore-Skelboe	N	1171
	$y*$	
		-1.1313
	X_1*	[0.0391, 0.0781]
	X_2*	[-0.7239,-0.6979]
Modified Moore-Skelboe	N'	223
	$y*$	-1.0316
		-1.1258
	X_1*	[-0.1563,-0.1172]
	X_2*	[0.6094, 0.6563]
Comparison	N/N'	5.2

VII. Conclusions

We combined interval constraints, as implemented in BNR Prolog, with the older method of interval analysis, as embodied in the Moore-Skelboe algorithm for global optimization. We ran the resulting program on several examples. We reported a comparison on the three examples we found in the literature where the performance of the Moore-Skelboe algorithm is given. We found a factor of two to five improvement in the number of function evaluations.

We also attempted to implement the original Moore-Skelboe algorithm in BNR Prolog, making only such modifications as to allow us to get results at all. The performance we found was very nearly the same as reported by Ratschek and Rokne.

We feel this result suggests investigating other numerical applications of interval constraints. As interval constraints translate *any* form of computation to constraint solving, *constrained* global optimization is an attractive area for such new applications.

References

[1] F. Benhamou and W. J. Older, "Applying interval arithmetic to real, integer, and Boolean constraints, *Journal of Logic Programming* (to appear).

[2] BNR, **BNR Prolog user guide and reference manual**, 1988.

[3] A. Borning, "Thinglab — an object-oriented system for building simulations using constraints", *Proc. 5th Int. Joint Conf. on Artificial Intelligence*, 1977.

[4] A. Colmerauer, H. Kanoui, R. Paséro, and P. Roussel, **Un Esystème de communication homme-machine en français**, Université d'Aix-Marseille II, Groupe d'Intelligence Artificielle, Technical Report, 1972.

[5] E. Hansen, **Global Optimization Using Interval Analysis**, Marcel Dekker, 1992.

[6] A. K. Mackworth, "Consistency in networks of relations", *Artificial Intelligence*, 1977, Vol. 8, pp. 99–118.

[7] Ugo Montanari, "Networks of constraints: Fundamental properties and applications to picture processing", *Information Science*, 1974, Vol. 7, No. 2, pp. 95–132.

[8] H. Ratschek and J. Rokne, **New Computer Methods for Global Optimization**. Ellis Horwood/John Wiley, 1988.

[9] L. Sterling and E. Shapiro, **The Art of Prolog**, MIT Press, 1994.

[10] G. J. Sussman and G. L. Steele Jr., "Constraints — a language for expressing almost-hierarchical descriptions", *AI Journal*, 1980, Vol. 14, pp. 1–39.

[11] D. Waltz, "Understanding line drawings in scenes with shadows", In: P. H. Winston (editor), **The Psychology of Computer Vision**, McGraw-Hill, 1975, pp. 19–91.