Optimization by Moore-Skelboe and similar algorithms

M.H. van Emden

Department of Computer Science,
University of Victoria, Victoria, Canada
vanemden@cs.uvic.ca
http://www.cs.uvic.ca/~vanemden/

Abstract. After a formulation of the minimization problem, we present an algorithm that yields an interval for the global minimum that is narrower than a predetermined tolerance ϵ. We consider the situation that ϵ is chosen too small, in which case the algorithm gives the best possible lower bound for the global minimum.
We extend this algorithm to produce the narrowest possible interval for the global minimum. In addition we consider extensions of the Moore-Skelboe algorithm that yield information about the location of objective-function values that are near the global minimum.

1 The optimization problem

Definition 1. *An* optimization setting *has the following components.*
(1) An objective function f*, which is a function of type* $\mathcal{R}^n \to \mathcal{R}$*.*
(2) A domain $\mathcal{D}$*, which is a non-empty subset of* $\mathcal{R}^n$*.*
(3) A set of conditions of the form $g_i(x_1, \ldots, x_n) \leq 0$*, for* $i \in \{1, \ldots, m\}$*. Here* $g_1, , \ldots, g_m$ *are functions of type* $\mathcal{R}^n \to \mathcal{R}$*.*
(4) A feasible set*, which is the intersection of the domain with the subset of* $\mathcal{R}^n$ *that satisfies the conditions.*
(5) The global minimum μ *is defined as the greatest lower bound of* f *restricted to the feasible set.*
(6) The δ-minimizer *is defined as the intersection of the feasible set with* $\{\langle x_1, \ldots, x_n \rangle \mid abs(f(x_1, \ldots, x_n) - \mu) \leq \delta\}$*, for some* $\delta \geq 0$*. The* minimizer *without qualification is the 0-minimizer.*
(7) An optimization setting is regular *if the minimizer consists of a single point.*

We use *lb* and *ub* to denote the lower and upper bounds of an interval, respectively. An interval's width is denoted by w.

The global minimum may fail to exist because f is not bounded from below or because the feasible set is empty. Given an optimization setting, an optimization *problem* is to obtain information about the global minimum or about the δ-minimizer. As we will see below, this can take several forms.

We are interested in optimization settings where there exists an unknown and possibly large number of local minima. As we are only interested in methods that ignore local minima, their definition need not concern us.

It is essential that the objective function f be given by an expression in terms of rational operations in which the standard functions (exponential, logarithm, trigonometric) may also occur. This rules out, for example, objective functions that are given as sets of observational data. For such data to become usable for the construction of an objective function, approximation or interpolation techniques can often be applied to obtain an objective function of the required form.

Definition 2. *We assume an expression E is given that contains variables $x_1, \ldots, x_n$. E computes f in the sense that E has as value $f(a_1, \ldots, a_n)$ whenever $a_1, \ldots, a_n$ are substituted in E for $x_1, \ldots, x_n$. We assume that E can be evaluated in interval arithmetic.*

The symbol f is overloaded:
(1) if the arguments are reals $x_1, \ldots, x_n$, then f denotes the objective function;
(2) if the arguments are intervals $X_1, \ldots, X_n$, then f denotes the function that has as value the result of evaluating in interval arithmetic E with $X_1, \ldots, X_n$ substituted for $x_1, \ldots, x_n$;
(3) if the argument is $X_1 \times \cdots \times X_n$, then f denotes $f(X_1, \ldots, X_n)$, as defined above.

2 Interval arithmetic for global optimization

Interval arithmetic has the property of producing intervals that contain all possible values. More precisely, we have the following theorem.

Theorem 1. *(The* Fundamental Law of Interval Arithmetic*).*
Letting f be a function of type $\mathcal{R}^n \rightarrow \mathcal{R}$ and let $X_1, \ldots, X_n$ be intervals, then we have

$$\{f(x_1, \ldots, x_n) \mid x_1 \in X_1, \ldots, x_n \in X_n\} \subset f(X_1, \ldots, X_n),$$

Here the two ocurrences of f have different meanings as in Definition 2

The presence of an unknown and possibly large number of local minima may seem to preclude the possibility of finding a lower bound for the global minimum. After all, even if one has identified a thousand local minima, how does one know that there is not yet another one with an objective-function value lower than any found so far? Moreover, it is possible that a sharp global minimum exists that fits entirely between two consecutive floating-point numbers.

The fundamental theorem guarantees that even in such extreme situations, the lower bound of the interval for f computed by interval arithmetic is a lower bound for the global minimum in $X_1 \times \cdots \times X_n$.

However, neither the fundamental theorem nor anything else ensures that this is *useful*: it may be that the lower bound is far away from the global minimum. The narrower the intervals in $X_1 \times \cdots \times X_n$, the closer the left bound of the interval $f(X_1 \times \cdots \times X_n)$ is to the greatest lower bound of $\{f(x_1, \ldots, x_n) \mid x_1 \in X_1, \ldots, x_n \in X_n\}$.

This suggests subdividing $X_1 \times \cdots \times X_n$ into smaller boxes and evaluating f over each of these. The result of subdividing is a set of boxes that covers the set in question.

Definition 3. *Let $N > 0$ be an integer. If $\langle S_0, \ldots, S_{N-1} \rangle$ is a sequence of non-empty subsets of a set S, then the sequence is called a* cover *in S.*

Assume that we have a cover $\langle B_0, B_1, \ldots, B_{N-1} \rangle$ for $X = X_1 \times \cdots \times X_n$ and that the cover is ordered by non-decreasing lower bounds of $f(B_i)$, for $i \in \{1, \ldots, N-1\}$. Initially, we can take $B_0 = X$ and $N = 1$. Such a cover contains the minimizer. We will consider algorithms that change a given cover containing the minimizer to one that has a smaller union yet still contains the minimizer.

Theorem 2. *Let $\langle B_0, B_1, \ldots, B_{N-1} \rangle$ be a cover containing the global minimizer. Let U be the smallest of among the right bounds of the intervals $\langle f(B_0), f(B_1), \ldots, f(B_{N-1}) \rangle$. The interval $[lb(f(B_0)), U]$ contains the global minimum μ.*

Proof. Consider $lb(f(B_0))$. None of the other boxes in the cover has a smaller left bound. As the global minimum has to be achieved in at least one of the boxes (because the number of sets in the cover is finite), the left bound of $f(B_0)$ is a lower bound for μ. Let us now consider U. Suppose it is the right bound of $f(B_i)$ with $i \in \{0, \ldots, N-1\}$. As B_i is nonempty, $f(B_i)$ contains at least one value y of f. We have $U = ub(f(B_i)) \geq y \geq \mu$. We conclude that $[lb(f(B_0)), U]$ contains the global minimum μ.

3 The Moore-Skelboe algorithm

The considerations in the previous section suggest the possibility of obtaining useful information about the global minimum by constructing a suitable cover.

We should now decide what kind of information we desire. One question is: *what* is the global minimum? This requires finding an interval for μ. Another question is: *where* is the global minimum? Or rather, where does the objective function come close to the global minimum? Determining what the global minimum is we call the *fathoming problem.* Obtaining information about where the objective function gets values that are close enough to the global minimum we call the *localization problem.*

We consider these problems in turn.

3.1 The fathoming problem

Suppose we want to find an interval for μ of width at most ϵ, for some real $\epsilon > 0$. Consider the algorithm in Figure 1.

The algorithm in Figure 1 has some positive features. In the first place, it may find a sufficiently narrow interval for μ. Second, it does this by subdividing X in an adaptive way, as explained below.

```
while (w(f(B_0)) > ε) {
        // invariant: μ ∈ f(B_0)
        remove B_0 from the cover
        split B_0 and add the results of splitting to the cover
        re-number the cover
}
// μ ∈ f(B_0) and w(f(B_0)) ≤ ε
```

Fig. 1. The algorithm MS_0. The function w has as value the width of its interval argument. The tolerance ϵ is positive.

Let us consider the operation "split". It assumes that the results are nonempty, are both proper subsets, and have a union that is equal to the box that was split. The box to be split is a cartesian product of n intervals. So the split can be done in up to n different ways. The number of boxes in the cover created by the algorithm typically becomes so large that the cover cannot be stored. So one should be careful what to split. The algorithm splits the box most likely to contain the minimizer. The subdivision resulting from the splits in this algorithm is *adaptive*: boxes far away from the global minimum tend not to be split. This should go some way towards avoiding covers with too many sets.

However, algorithm MS_0 needs improvement in several ways. For example, what happens if one takes the positive ϵ too small? Even if the algorithm does not abort because the number of sets in the cover has become too large to be stored, B_0 may become the result of so many splits that all its projections are intervals that have adjacent floating-point numbers as bounds. The split operation should be such that both of the resulting boxes are smaller than the original, and this is then no longer possible. So let's assume a boolean function "splittable" that tells whether its box argument can be split into two boxes that are both smaller.

Let us also address the problem of the cover becoming too large. Suppose there are boxes B_i and B_j in the cover such that $lb(f(B_i)) > ub(f(B_j))$, then B_i does not intersect with the minimizer and we can remove it from the cover. So it is of interest to maintain in the algorithm an upper bound U for the global minimum. An example of such a U is the least of the right bounds of f applied to the boxes in the cover.

Algorithm MS_1 in Figure 2 is an improvement because it has a defined outcome for every non-negative value of ϵ. Moreover, the "failure" outcome is useful because it produces the best possible lower bound for μ.

The question arises, does it also produce the best possible interval for μ? The answer is that this is not necessarily so. Algorithm MS_1 only splits B_0. The interval also depends on the upper bound U for μ. It may be that splitting boxes of the cover other than B_0 results in a lower U. Hence the algorithm in Figure 3.

Algorithm MS_2 seems to be the best one can do for the fathoming problem.

The main limitation of the algorithms of this type is the large number of sets in the cover. Removing the sets of the cover whose lower bound exceeds U is essential to get good results.

while $(U - lb(f(B_0))) > \epsilon$ and $splittable(B_0))$ {
    // invariant: $\mu \in f(B_0)$
    remove B_0 from the cover
    split B_0 and add the results of splitting to the cover
    update U
    remove all boxes from the cover with lower bound greater than U
    re-number the cover
}
// $f(B_0) \leq \epsilon$ or $\neg splittable(B_0)$
if $(U - lb(f(B_0)) \leq \epsilon)$ then output "success: $\mu \in [lb(f(B_0)), U]$
else output "failure: $lb(f(B_0))$ is the best possible lower bound for μ"

Fig. 2. The algorithm MS_1. In case of success, an interval of the desired width has been found. Otherwise, a lower bound for the global minimum has been found that is the best possible one, given the expression for f and the precision of the arithmetic. So this information can be forced by setting ϵ to zero or to an unrealistically low positive value.

while $(U - lb(f(B_0))) > \epsilon$ and $\exists i$ such that $splittable(B_i))$ {
    // invariant: $\mu \in f(B_0)$
    remove B_i from the cover
    split B_i and add the results to the cover
    update U
    remove all boxes from the cover with lower bound greater than U
    re-number the cover
}
// $f(B_0) \leq \epsilon$ or $\neg\exists i \in \{0, \ldots, N-1\}$ such that $splittable(B_i)$
if $(U - lb(f(B_0)) \leq \epsilon)$ then output "success: $\mu \in [lb(f(B_0)), U]$
else output "failure: $[lb(f(B_0)), U]$ is the best interval for μ"

Fig. 3. The algorithm MS_2. In case of success, an interval of the desired width has been found. Otherwise, an interval for the global minimum has been found that is best, given the expression for f and the precision of the arithmetic. So this information can be forced by setting ϵ to zero or to an unrealistically low positive value.

Algorithms MS_1 and MS_2 are examples of *branch-and-bound* algorithms. The splitting is often referred to as "branching". The removing of the sets from the cover on the basis of U is called "bounding".

One way to get better results is to use as source for U the value of the objective function somewhere inside the boxes of the cover. These are also upper bounds for the global minimum and are less than the upper bound obtained by interval arithmetic. This applies without any problems in the case of unconstrained optimization. However, in the presence of constraints, one would have to prove the existence of a feasible point inside the box B_i. This is often difficult and then the best one can do is to use the upper bound of $f(B_i)$.

3.2 The localization problem

The localization problem is to gain information about the δ-minimizer. This can take two forms: an *outer* approximation or an *inner* approximation.

An outer approximation is a set of boxes whose union contains the δ-minimizer. This can always be achieved, though the union may be so large as not to be useful. The algorithm in Figure 4 makes this union as small as possible.

```
while (the B_i with least upper bound is splittable){
        remove B_i from the cover
        split B_i and add the results to the cover
        update U
        remove all boxes from the cover with lower bound greater than U + δ
        re-number the cover
}
// the best U has been obtained
while (∃i such that (U + δ) ∈ f(B_i) and splittable(B_i)){
        remove B_i from the cover
        split B_i and add the results to the cover
        remove all boxes from the cover with lower bound greater than U + δ
        re-number the cover
}
```

Fig. 4. The algorithm MS_3. After termination, the best outer approximation to the δ-minimizer is the union of $\{B_0, \ldots, B_i\}$ where i is the greatest i such that $lb(f(B_i)) \leq (U + \delta)$.

Let U be the upper bound of the box in the cover with least upper bound. For an inner approximation, we let $\epsilon = U - lb(f(B_0))$, which is the width of the best interval for μ. At least B_0 will now be contained in the $(\epsilon + \delta)$-minimizer for any positive δ. In fact, all boxes of the cover that have upper bounds less than $U + \delta$ have a union that is contained in the $(\epsilon + \delta)$-minimizer. This inner approximation can be improved (that is, made larger) by splitting certain boxes. This improvement is carried out by the algorithm in Figure 5.

```
while (∃i such that (U + δ) ∈ f(B_i)) {
        remove B_i from the cover
        split B_i and add the results to the cover
        remove all boxes from the cover with lower bound greater than U + δ
        re-number the cover
}
```

Fig. 5. The algorithm MS_4. After termination, the union of boxes with upper bounds below $U + \delta$ is the best inner approximation to the $(\epsilon + \delta)$-minimizer.