8-2007

m Solutions Good, m-1 Solutions Better

Luc Longpre
The University of Texas at El Paso, longpre@utep.edu

William Gasarch

G. W. Walster

Vladik Kreinovich
The University of Texas at El Paso, vladik@utep.edu

Comments:
UTEP-CS-00-40b.
Published in *Applied Mathematical Sciences*, 2008, Vol. 2, No. 5, pp. 223-239.

Applied Mathematical Sciences, Vol. x, 200x, no. xx, xxx - xxx

m SOLUTIONS GOOD, $m-1$ SOLUTIONS BETTER

Luc Longpré, Vladik Kreinovich

Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA
longpre@utep.edu, vladik@utep.edu

William Gasarch

Department of Computer Science
University of Maryland
College Park, MD 20742, USA

G. William Walster

22116 Dean Ct.
Cupertino, CA 95014-2723, USA
walster1@comcast.net

Abstract

One of the main objectives of theoretical research in computational complexity and feasibility is to explain experimentally observed difference in complexity. Empirical evidence shows that the more solutions a system of equations has, the more difficult it is to solve it. Similarly, the more global maxima a continuous function has, the more difficult it is to locate them. Until now, these empirical facts have been only partially formalized: namely, it has been shown that problems with two or more solutions are more difficult to solve than problems with exactly one solution. In this paper, we extend this result and show that for every m, problems with exactly m solutions are more difficult to solve than problems with $m-1$ solutions. Rephrasing Orwell's "Four legs good, two legs better", we can describe this result as "m solutions good, $m-1$ solutions better".

Mathematics Subject Classification: 68Q17, 68Q15, 90C60, 65G20

Keywords: computability, optimization, solving systems of equations, computational complexity

1 Introduction

In many real-life situations, we want to find the *best* decision, the *best* control strategy, etc. The corresponding problems are naturally formalized as *optimization* problems: we have a function $f(x_1, \ldots, x_n)$ of several variables, and we want to find the values $(x_1, \ldots, x_n)$ for which this function attains the largest (or the smallest) possible value.

Many numerical algorithms have been proposed for solving optimization problems. Unfortunately, many of these algorithms often end up in a *local* maximum instead of the desired global one.

- In some practical situations, e.g., in decision making, the use of local maximum simply degrades the quality of the decision but is not, by itself, disastrous.
- However, in some other practical situations, missing a global maximum or minimum may be disastrous.

Let us give two example:

- In *chemical engineering*, global minima of the energy function often describe the stable states of the system. If we miss such a global minimum, the chemical reactor may go into an unexpected state, with possible serious consequences.
- In *bioinformatics*, the actual shape of a protein corresponds to the global minimum of the energy function. If we find a local minimum instead, we end up with a wrong protein geometry. As a result, if we use this wrong geometry as a computer simulation for testing recommendations on the medical use of chemicals, we may end up with medical recommendations which harm a patient instead of curing him.

For such applications, it is desirable to use *rigorous*, *automatically verified* methods of global optimization, i.e., methods which never discard an actual global maximum; for a survey of such methods, see, e.g., [12]. These methods usually start with a large "box" on which a function is defined (and on which global maxima can be located), and produce a list of small-size boxes with the property that every global maximum is guaranteed to be contained in one of these boxes.

Most of such guaranteed methods use (a version of) *interval computations*. The main idea of interval computations is as follows: To solve a given numerical

problem (e.g., an optimization problem), numerical methods typically generate better and better estimates for different quantities related to the problem – such as the actual global maximum of the function, the value of its partial derivatives at different points, etc.

- In *traditional* numerical techniques, for each approximated numerical quantity, an approximation is a real number, with no guarantees on the approximation accuracy.
- In *interval* methods, at any given moment of time, for each approximated quantity x, we compute not only the approximate value $\widetilde{x}$, but also the *upper bound* Δ on the possible approximation error, i.e., a number Δ for which we are guaranteed that $|x - \widetilde{x}| \le \Delta$. In other words, at any step, we have not only an approximate value $\widetilde{x}$ of the approximate quantity, we also have an *interval* $\mathbf{x} = [\widetilde{x} - \Delta, \widetilde{x} + \Delta]$ which is *guaranteed* to contain the (unknown) actual value of x.

As we have mentioned, rigorous methods of global optimization start with a large box as a location of the unknown global maxima and gradually replace it will a small finite collection of small boxes. The decrease in a box size is usually achieved by dividing one of the boxes into several sub-boxes and eliminating some of these sub-boxes.

When can we eliminate a sub-box B? At every stage of the optimization algorithm, we have already computed several values of the optimized function $f(x_1, \ldots, x_n)$, so we know that the global maximum of the function f cannot be smaller than the largest M of these already computed values. Thus, if we can guarantee that the maximum of the function f on a box B is smaller than M, we can thus exclude this box from the list of possible locations of a global maximum. To get such a guarantee, we can find an enclosure for the range of the function on a subbox, e.g., by using methods of *interval arithmetic* [12], when we parse the computation of the function f and replace each arithmetic operation by the corresponding operation with intervals.

In some real-life problems, we are not yet ready for optimization, e.g., because the problem has so many constraints that even finding *some* values $x = (x_1, \ldots, x_n)$ of the parameters x_i which satisfy all these constraints is an extremely difficult task. For such problems, we arrive at the problem of satisfying given constraints, e.g., solving a given system of equations. In many such problems, it is important not to miss a solution.

The complexity of locating global maxima of f is empirically known to depend on the number of these global maxima: the fewer global maxima, the easier the problem [8, 9, 10, 11, 12, 28]. It is desirable to formalize and explain this empirical fact.

Previously, this result have been formalized only partially: namely, it was shown that global optimization is easier when we have exactly one global max-

imum than when have several. In this paper, we extend this result and show that optimization with m global maxima is, in some formal sense, more difficult than optimization with $m-1$ global maxima (and the same is true for solving systems of equations).

2 Formalization of the Problem and Previously Known Results

In order to formulate this result, we must recall some basic definitions of computable ("constructive") real numbers and computable functions from real numbers to real numbers (see, e.g., [1, 3, 4, 5, 24]):

Definition 2.1 *A real number x is called* computable *if there exists an algorithm (program) that transforms an arbitrary integer k into a rational number x_k that is $2^{-k}-$close to x. It is said that this algorithm computes the real number x.*

When we say that a computable real number is given, we mean that we are given an algorithm that computes this real number.

Definition 2.2 *A function $f(x_1,\ldots,x_n)$ from real numbers to real numbers is called* computable *if there exist algorithms U_f and φ, where:*

- *U_f is a rational-to-rational algorithm which provides, for given rational numbers $r_1,\ldots,r_n$ and an integer k, a rational number $U_f(r_1,\ldots,r_n,k)$ which is 2^{-k}-close to the real number $f(r_1,\ldots,r_n)$, and*

$$|U_f(r_1,\ldots,r_n,k)-f(r_1,\ldots,r_n)|\le 2^{-k},\ \textit{and}$$

- *φ is an integer-to-integer algorithm which gives, for every positive integer k, an integer $\varphi(k)$ for which $|x_1-x_1'|\le 2^{-\varphi(k)}$, $\ldots$, $|x_n-x_n'|\le 2^{-\varphi(k)}$ implies that $|f(x_1,\ldots,x_n)-f(x_1',\ldots,x_n')|\le 2^{-k}$.*

When we say that a computable function is given, we mean that we are given the corresponding algorithms U_f and φ.

Theorem 2.1 *[13, 14, 15, 18, 20] There exists an algorithm U such that:*

- *U is applicable to an arbitrary computable function $f(x_1,\ldots,x_n)$ that attains its maximum on a computable box $B=[a_1,b_1]\times\ldots\times[a_n,b_n]$ at exactly one point $x=(x_1,\ldots,x_n)$,*
- *for every such function f, the algorithm U computes the global maximum point x.*

Theorem 2.2 *[17, 18, 19, 20, 21, 22, 23, 24] No algorithm U is possible such that:*

- *U is applicable to an arbitrary computable function $f(x_1, \ldots, x_n)$ that attains its maximum on a computable box $B = [a_1, b_1] \times \ldots \times [a_n, b_n]$ at exactly two points, and*
- *for every such function f, the algorithm U computes one of the corresponding global maximum points x.*

These results *partially* explain the above intuition because they show that the problem of locating global maxima is easier if we have a single global maximum and more difficult if we have several global maxima. These results, however, *do not completely* explain this intuition because they do not explain why, say, a problem with three global maxima is more complex than a problem with two global maxima.

Similar results hold for roots (solutions) of a system of equations:

Definition 2.3 *By a* computable system of equations *we mean a system $f_1(x_1, \ldots, x_n) = 0$, ..., $f_k(x_1, \ldots, x_n) = 0$, where each of the functions f_i is a computable function on a computable box $B = [a_1, b_1] \times \ldots \times [a_n, b_n]$.*

Theorem 2.3 *[13, 14, 15, 18, 20] There exists an algorithm U such that:*

- *U is applicable to an arbitrary computable system of equations which has exactly one solution, and*
- *for every such system of equations, the algorithm U computes its solution.*

Theorem 2.4 *[17, 18, 19, 20, 21, 22, 23, 24] No algorithm U is possible such that:*

- *U is applicable to an arbitrary computable system of equations which has exactly two solutions, and*
- *for every such system of equations, the algorithm U computes one of its solutions.*

Comment. It is known that the general problem of solving a system of polynomial equations with rational coefficients is NP-hard [24] (it is even NP-hard for quadratic equations). The problem of finding the *unique* solution to a system of equations (or the unique point where the maximum is attained) is as complicated as the problem of finding the *unique satisfying vector* for a given propositional formula [24]. The latter problem (it is usually denoted by *USAT*, from *u*nique *sat*isafiability) is known to be "almost" NP-hard in the sense that every

other problem from the class NP can be *probabilistically* reduced to *USAT*; so if we were able to solve all the instances of *USAT* in polynomial time, we would have a *probabilistic* polynomial-time algorithm that solves almost all instances of all problems from the class NP. (Exact definition are somewhat complicated so, due to the lack of space, we refer the interested reader to [7] and [27]. Note that in [2], arguments are given that this problem may not be NP-hard.)

3 New Results

The following modification of the above results provides the desired *complete explanation:* For two global maxima, we cannot pinpoint one of them, but, as we will show, we can compute the next best thing: namely, we can compute *two* locations with a guarantee that one of them contains a global maximum. For functions with three global maxima, we cannot find neither one nor two such locations, but we can find *three* locations one of which contains the global maximum, etc. Thus, we have formalized the intuitive idea that the fewer global maxima, the easier the global optimization problem.

Definition 3.1 *We say that a box $B = [a_1, b_1] \times \ldots \times [a_n, b_n]$ is* of size $\leq \varepsilon$ *if each of its sides is of size $\leq \varepsilon$, i.e., if $b_i - a_i \leq \varepsilon$ for all $i = 1, \ldots, n$.*

Theorem 3.1 *Let $m > 1$ be an integer. Then:*

- *There exists an algorithm U such that:*
  - *U is applicable to an arbitrary computable function $f(x_1, \ldots, x_n)$ on a computable box $B = [a_1, b_1] \times \ldots \times [a_n, b_n]$ that attains its maximum on B at exactly m points, and*
  - *for every computable real number $\varepsilon > 0$, the algorithm U returns m boxes of size ε with a guarantee that each global maximum is contained in one of these boxes.*
- *No algorithm U is possible such that:*
  - *U is applicable to an arbitrary computable function $f(x_1, \ldots, x_n)$ on a computable box $B = [a_1, b_1] \times \ldots \times [a_n, b_n]$ that attains its maximum on B at exactly m points, and*
  - *for every computable real number $\varepsilon > 0$, the algorithm U returns $m-1$ boxes of size ε with a guarantee that one of these boxes contains a global maximum.*

Theorem 3.2 *Let $m > 1$ be an integer. Then:*

- *There exists an algorithm U such that:*
  - *U is applicable to an arbitrary computable system of equations which has exactly m solutions, and*
  - *for every computable real number $\varepsilon > 0$, the algorithm U returns m boxes of size ε with a guarantee that each solution is contained in one of these boxes.*
- *No algorithm U is possible such that:*
  - *U is applicable to an arbitrary computable system of equations which has exactly m solutions, and*
  - *for every computable real number $\varepsilon > 0$, the algorithm U returns $m-1$ boxes of size ε with a guarantee that one of these boxes contains a solution.*

The problem becomes even more complicated if we do not know the actual number $n(f, B)$ of points where the maximum of a function f on the box B is attained – only the upper bound m on this number. In this case, $n(f, B)$ can take any value from the set $\{1, \ldots, m\}$. It turns out that not only we cannot predict $n(f, B)$ for a given f, we cannot even limit the m-element list of possible cardinalities to a smaller sublist:

Theorem 3.3 *Let $m > 1$ be an integer. Then, no algorithm is possible which, given a computable function f on a computable box B which attains its maximum at $n(f, B) \le m$ points, returns a set $S(f, B) \subset \{1, \ldots, m\}$ with $\#S(f, B) < m$ elements which contains the value $n(f, B)$.*

A similar result holds for solutions:

Theorem 3.4 *Let $m > 1$ be an integer. Then, no algorithm is possible which, given a computable system of equations s which has $n(s) \le m$ solutions, returns a set $S(s) \subset \{0, 1, \ldots, m\}$ with $\#S(s) < m+1$ elements which contains the value $n(s)$.*

4 Proofs

4.1 Proof of Theorem 3.1

1. It is known that there exists an algorithm which, given a computable function on a computable box, and a given $\delta > 0$ returns a rational number M which is δ-close to $\max f$ [1, 3, 4, 5, 24]. Let us reproduce the main idea of this proof.

1.1. First, we prove that there exists an integer m for which the 2^{-m}-approximation δ_m to δ exceeds $3 \cdot 2^{-m}$.

Indeed, since $\delta > 0$, we have $\delta > 2^{-k}$ for some k. Therefore, for the $2^{-(k+2)}$-approximation δ_{k+2} to δ, we get $|\delta_{k+2} - \delta| \le 2^{-(k+2)}$ hence

$$\delta_{k+2} \ge \delta - 2^{-(k+2)} > 2^{-k} - 2^{-(k+2)} = 3 \cdot 2^{-(k+2)}.$$

So, the existence is proven for $m = k + 2$.

This m can be algorithmically computed as follows: we sequentially try $m = 0, 1, 2, \ldots$ and check whether $\delta_m > 3 \cdot 2^{-m}$; when we get the desired inequality, we stop.

1.2. Let us now show that for the integer m computed according to Part 1.1 of this proof, we have $\delta > 2 \cdot 2^{-m}$.

Indeed, since $\delta_m > 3 \cdot 2^{-m}$ and $|\delta - \delta_m| \le 2^{-m}$, we can conclude that

$$\delta \ge \delta_m - 2^{-m} > 3 \cdot 2^{-m} - 2^{-m} = 2 \cdot 2^{-m}.$$

So, if we can find a rational number M which is $2 \cdot 2^{-m}$-close to $\max f$, this rational number will thus be also δ-close to $\max f$.

1.3. Let us now use this m to compute the desired δ-approximation to $\max f$.

1.3.1. By using the second algorithm φ in the definition of a computable function, we can find a value $\varphi(m)$ such that if $|x_i - x_i'| \le \varphi(m)$ for all $i = 1, \ldots, n$, then

$$|f(x_1, \ldots, x_n) - f(x_1', \ldots, x_n')| \le 2^{-m}.$$

For each dimension $[a_i, b_i]$ of the box B, we can then take finitely many values

$$r_i^{(1)}, r_i^{(2)} = r_i^{(1)} + \varphi(m), r_i^{(3)} = r_i^{(2)} + \varphi(m), \ldots, r_i^{(N_i)} = r_i^{(N_i - 1)} + \varphi(m)$$

(separated by $\varphi(m)$) which cover the corresponding interval. Then, each value $x_i \in [a_i, b_i]$ will be different by one of these values $r_i^{(k_i)}$ by $\le \varphi(m)$.

1.3.2. Combining the values corresponding to different dimensions, we get a finite list of rational-valued vectors $\left(r_1^{(k_1)}, \ldots, r_n^{(k_n)}\right)$ with the property that every vector $(x_1, \ldots, x_n) \in B$ is $\varphi(m)$-close to one of these vectors.

Due to the definition of $\varphi(m)$, this means that each value $f(x_1, \ldots, x_n)$ is 2^{-m}-close to one of the values $f\left(r_1^{(k_1)}, \ldots, r_n^{(k_n)}\right)$. Therefore, the desired $\max f$ is 2^{-m}-close to the maximum of all the values $f\left(r_1^{(k_1)}, \ldots, r_n^{(k_n)}\right)$.

By using the algorithm U_f, we can compute each of these values with the accuracy 2^{-m}. Thus, the maximum M of thus computed rational values $U_f\left(r_1^{(k_1)}, \ldots, r_n^{(k_n)}, m\right)$. is 2^{-m}-close to the maximum of all the values $f\left(r_1^{(k_1)}, \ldots, r_n^{(k_n)}\right)$, and hence, $2 \cdot 2^{-m}$-close to $\max f$. Thus, M is indeed δ-close to $\max f$. The first part is proven.

2. Let us now prove that the existence of the desired algorithm U.

2.1. First, we prove that for every function f with exactly m global maxima $x^{(1)}, \ldots, x^{(m)}$ in which

$$f\left(x^{(1)}\right) = \ldots = f\left(x^{(N)}\right) = M \stackrel{\text{def}}{=} \max_B f(x),$$

there exists a real number $\delta > 0$ such that for every $x \in B$, if $f(x)$ is δ-close to the maximum M, then x is $(\varepsilon/2)$-close to one of the global maxima $x^{(i)}$ $(1 \le i \le m)$.

Indeed, if such a δ does not exist, then, for every $\delta = 2^{-k}$, there exists a point $p^{(k)}$ for which $f\left(p^{(k)}\right) \ge M - 2^{-k}$ and $\rho\left(p^{(k)}, x^{(i)}\right) > \varepsilon/2$ for all $i = 1, \ldots, m$ (where ρ denotes the distance). Since the box B is closed and bounded, it is a compact, so the sequence $p^{(k)}$ has a convergent subsequence $p^{(k_i)} \to p \in B$. In the limit $k_i \to \infty$, we get:

- $f(p) \ge M = \max f(x)$, hence p is the global maximum point, and
- at the same time, $\rho\left(p, x^{(i)}\right) \ge \varepsilon/2 > 0$, so p is different from all known m global maxima.

The contradiction shows that such a $\delta > 0$ must exist.

2.2. From 2.1, we can conclude that is more than $(\varepsilon/2)$ away from one of the local maxima, then $f(x) < M - \delta$.

2.3. Now, we can present the desired algorithm.

To get the desired boxes, for every $N = 1, 2, \ldots$, we do the following:

- We divide the original box into $N \times N \times \ldots \times N$ subboxes $B_1^{(N)}, B_2^{(N)}, \ldots$.
- We compute the maximum $M_1^{(N)}, M_2^{(N)}, \ldots$ of f on each of these subboxes with an accuracy 2^{-N}. The maximum $M^{(N)}$ of these values is the 2^{-n}-approximation to the actual (unknown) maximum M of the function f.
- We dismiss all the subboxes $B_i^{(N)}$ for which $M_i^{(N)} < M^{(N)} - 2 \cdot 2^{-N}$, because for these subbboxes, the actual maximum is guaranteed to be snaller than M and hence, these subboxes cannot contain any global maxima.
- Finally, we check whether all these subboxes can be subdivided into m groups of size ε.

If they can be thus grouped, we get the desired m subboxes. If not, we increase N by one, and repeat the same procedure.

The fact that this algorithm stops for some N follows from the Part 2.1 of this proof: if N becomes so large that $2^{-N} < \delta/2$ and $2^{-N} < \varepsilon/4$, then for every box $B_i^{(N)}$ which is more than $(\varepsilon/2)$ away from one of the global maxima, its actual maximum is $< M - \delta$. Hence, its 2^{-N}-approximate value $M_i^{(N)}$ is $< M^{(N)} - 2 \cdot 2^{-N}$, so this box will be dismissed. As a result, all remaining boxes will indeed be ε-close to the actual global maxima. The existence is proven.

3. Let us now prove the impossibility of an algorithm which would return $m - 1$ boxes such that one of these boxes contains a global maximum of a given function f.

We will prove this impossibility by reduction to a contradiction. Let us assume that such an algorithm exists.

3.1. First, let us prove that we will then have an algorithm I which, given an arbitrary everywhere defined computable sequence $a : \mathbb{N} \to \mathbb{N}$ from natural numbers to natural numbers, returns an integer $I(a) \in \{1, \ldots, m\}$ with the following property: if $\exists n\, (a(n) \neq 0)$, then $I(a)$ is different from the first non-zero value of $a(n)$.

3.1.1. First, for every such sequence a, we construct a computable function $f_a : [0, 2m-1] \to [0, 1]$.

This function $f_a(x)$ is constructed as the sum of the constructively converging functions

$$f_a(x) = f_a^{(0)}(x) + f_a^{(1)}(x) + \ldots \tag{1}$$

for appropriately defined functions $f_a^{(i)}(x)$. In the following, we describe an algorithm which, given i and x, computes the value $f_a^{(i)}(x)$ of i-th component function. The values of i-th function is from the interval $[0, 2^{-i}]$ and therefore, it is easy to prove that their sum is indeed a computable function in the sense of the above definitions (for exact proofs, see, e.g., [1, 3, 4, 5]).

For constructing the component functions $f_a^{(i)}$, we will use an auxiliary "trapezoid" function $t : [0, 1] \to [0, 0.5]$ which is defined as follows:

- $t(x) = 0$ for $x \leq 0$;
- $t(x) = x$ for $0 \leq x \leq 0.25$;
- $t(x) = 0.25$ for $0.25 \leq x \leq 0.75$;
- $t(x) = 1 - x$ for $0.75 \leq 5 \leq 1$;
- $t(x) = 0$ for $1 \leq x$.

This trapezoid is of height $0.25 = 0.5 \cdot (1 - 2^{-1})$ with an upper horizontal part of length $0.5 = 2^{-1}$.

As long as we have

$$a(0) = \ldots = a(i) = 0, \tag{2}$$

we take, as $f_a^{(i)}(x)$, a function

$$F_i(x) \stackrel{\text{def}}{=} t_i(x) + t_i(x-2) + \ldots + t_i(x - 2 \cdot (m-1)), \tag{3}$$

where

$$t_i(x) \stackrel{\text{def}}{=} 2^{-i} \cdot t\left(2^i \cdot (x - 0.5) + 0.5\right).$$

In particular:

- For $i = 0$, we get $t_0(x) = t(x)$, hence

  $$f_a^{(0)}(x) = t(x) + t(x-2) + \ldots + t(x - 2 \cdot (m-1)),$$

  i.e., $f_a^{(0)}(x)$ is the sum of m identical trapezoids concentrated on the intervals $[0,1]$, $[2,3]$, $\ldots$, $[2 \cdot (m-1), 2 \cdot (m-1) + 1 = 2m - 1]$.

- For $i = 1$, adding $f_a^{(1)}(x)$ to $f_a^{(0)}(x)$ means that we add a small trapezoid on top of each of the previous m trapezoids. The parameters of both trapezoids are selected in such a way that their sides perfectly align, so for each of m intervals $[0,1]$, $\ldots$, $[2m-2, 2m-1]$, the graph of the sum $f_a^{(0)}(x) + f_a^{(1)}(x)$ is also a trapezoid, of height $0.5 \cdot (1 - 2^{-2})$ with an upper horizontal part of length 2^{-2}.

- Similarly, for every $i > 0$, if $a(0) = \ldots = a(i) = 0$, then on each of m intervals, the sum $f_a^{(0)}(x) + \ldots + f_a^{(i)}(x)$ is also a trapezoid, of height $0.5 \cdot (1 - 2^{-(i+1)})$ with an upper horizontal part of length $2^{-(i+1)}$.

When $i \to \infty$, the height tends to 0.5, and the width of the upper horizontal part tends to 0.

We have defined $f_a^{(i)}(x)$ for the case when the condition (2) is satisfied. Let j be the first value for which this condition is not satisfied. Then, as we have mentioned, on each of m intervals $[0,1]$,, $\ldots$, $[2m-2, 2m-1]$, the sum $f_a^{(0)}(x) + \ldots + f_a^{(j-1)}(x)$ is a trapezoid, of height $0.5 \cdot (1 - 2^{-j})$ with an upper horizontal part of length 2^{-j}:

- For the first interval $[0,1]$, the horizontal part is centered around the midpoint 0.5, i.e., it corresponds to the values $[0.5 - 0.5 \cdot 2^{-j}, 0.5 + 0.5 \cdot 2^{-j}]$.

- For k-th interval $[2k-2, 2k-1]$, the horizontal part is similarly centered around the midpoint $2k - 1.5$, i.e., it corresponds to the values $[(2k - 1.5) - 0.5 \cdot 2^{-j}, (2k - 1.5) + 0.5 \cdot 2^{-j}]$.

For the further construction, we compute $k_0 = \min(a(j), m)$; thus defined k_0 is an integer whose possible values range from 1 to m. We will halt the construction of our "pyramids" on all intervals except the interval # k_0, and on this particular interval, we will build m small identical "pyramids" on top

of the horizontal part $[x^-, x^+]$ of the corresponding trapezoid, where $x^- \stackrel{\text{def}}{=} (2k_0 - 1.5) - 0.5 \cdot 2^{-(j-1)}$ and $x^+ \stackrel{\text{def}}{=} (2k_0 - 1.5) + 0.5 \cdot 2^{-(j-1)}$.

Specifically, for every $i \geq j$, we define

$$f_a^{(i)}(x) = \frac{x^+ - x^-}{2m-1} \cdot F_{i-j}\left(\frac{2m-1}{x^+ - x^-} \cdot (x - x^-)\right),$$

where $F_i(x)$ is defined by the formula (3).

3.1.2. Let us now show that, for every computable everywhere defined sequence $a(n)$, the corresponding computable function $f_a(x)$ has exactly m global maxima, and describe the location of these global maxima.

When $a(0) = \ldots = a(i) = \ldots = 0$ for all i, the graph of the limit function (1) is the sum of m triangular functions each of which is located on the the corresponding interval $[0, 1]$, $\ldots$, $[2m-2, 2m-1]$, and each of which attains the maximum 0.5 at the midpoint of the corresponding interval. Thus, for $a(n) \equiv 0$, the function $f_a(x)$ has exactly m global maxima: 0.5, 2.5, $\ldots$, $2(m-1)+0.5$, each of which is located within the corresponding interval $[0, 1]$, $\ldots$, $[2m-2, 2m-1]$.

When $a(n) \not\equiv 0$, the function $f_a(x)$ depends on the first non-zero value $a(j)$: Namely, on j and on $k_0 = \min(a(j), m)$. Similarly to the case $a(n) \equiv 0$, the graph of the sum $f_a^{(i)}(x) + f_a^{(i+1)}(x) + \ldots$ is a sum of m triangular functions. Hence, for $a(n) \not\equiv 0$, the sum (1) attains maximum at exactly m points all of which are located within the k_0-th interval $[2k_0 - 2, 2k_0 - 1]$.

3.1.3. Let us now describe the desired $I(a)$.

The hypothetic algorithm U returns $m-1$ boxes, one of which is guaranteed to contain a global maximum. When ε is small enough (e.g., ≤ 0.5), each box can only contain points from one of the m intervals $[0, 1]$, $\ldots$, $[2m-2, 2m-1]$. Thus, when $a(n) \not\equiv 0$, the $m-1$ boxes resulting from applying U to f_a intersect with at most $m-1$ of these intervals. Thus, at least one of these intervals $[2i-2, 2i-1]$ does not intersect with any of the $m-1$ boxes. As $I(a)$, we then take the ordinal number of one of these non-intersecting intervals.

3.1.4. To complete the proof, let us show that if $a(n) \not\equiv 0$, then $I(a)$ is different from the first non-zero value $a(j)$ of the sequence $a(n)$.

Indeed, if $a(n) \not\equiv 0$, then one of the $m-1$ intervals must contain a global maximum. Since all global maxima are located in the interval $[2k_0 - 2, 2k_0 - 1]$ (where $k_0 = \min(a(j), m)$), one of the intervals from $U(f_1)$ must intersect with this interval $[2k_0 - 2, 2k_0 - 1]$. Since $I(a)$ is the ordinal number of an interval which does not intersect, and the interval $\#\ k_0$ does intersect, we conclude that $I(a) \neq k_0$.

By definition, $I(a) \leq m$. So:

- if $a(j) > m$, then $I(a) \neq a(j)$;

- if $a(j) \le m$, then $k_0 = a(j)$ hence also $I(a) \ne a(j)$.

In both cases, $I(a) \ne a(j)$. The statement is proven.

4. To complete the proof of the theorem, let us prove that the algorithm $I(a)$ described in Part 3 of this proof is impossible.

We will prove this impossibility in a way which is similar to the standard proof of the undecidability of the halting problem. Let f_n by a (partial) recursive function # n (or Turing machine # n, etc.), and let d be an integer. For each t, we can define $a(t)$ as follows:

- $a(t) = 0$ if the computation of f on d has not stopped by time t (i.e., in t steps), and
- $a(t) = f(d)$ is the computation of f on d stopped by time t.

By applying the algorithm I to this sequence a, we get a value $I(a)$ with the property that if f is applicable to d ($!f_n(d)$), then $I(a) \ne a(j) = f(d)$. Let us denote this value $I(a)$ by $v(n, d)$. Thus, we have an algorithm, which, given two integers n and d, always returns a value $v(n, d)$ such that if $!f_n(d)$ then $v(n, d) \ne f_n(d)$. In particular, the diagonal function $v(n, n)$ is also computable and everywhere defined. Thus, it has a number in the ordering of all recursive functions, i.e., there exists a number c for which $v(n, n) = f_c(n)$ for all n. Then, we get the desired contradiction:

- On one hand, from $v(n, n) = f_c(n)$, for $n = c$, we get $v(c, c) = f_c(c)$.
- On the other hand, here $!f_c(c)$, hence from the above property of v, we conclude that $v(c, c) \ne f_c(c)$.

The theorem is proven.

4.2 Proof of Theorem 3.2

Theorem 2 follows from Theorem 3.1 if we take into consideration that the problems of solving a system of equation and of locating global maxima can be naturally (and computably) reduced to each other in such a way that the solutions to the system of equations become global maxima and vice versa (and thus, the *number* of solutions becomes the *number* of global maxima and vice versa):

- If we know how to solve systems of equations, then the problem of locating global maxima of a function $f(x_1, \ldots, x_n)$ can be reformulated as a problem of finding all solutions to an equation $f_1(x_1, \ldots, x_n) = 0$, where

$$f_1(x_1, \ldots, x_n) \stackrel{\text{def}}{=} \max f - f(x_1, \ldots, x_n).$$

- Vice versa, if we know how to locate global maxima, then the problem of solving a system of equations $f_1(x_1,\ldots,x_n)=0$, $\ldots$, $f_k(x_1,\ldots,x_n)=0$ can be reformulated as a problem of finding all global maxima of a function

$$f(x_1,\ldots,x_n) \stackrel{\text{def}}{=} -(|f_1(x_1,\ldots,x_n)|+\ldots+|f_k(x_1,\ldots,x_n)|).$$

4.3 Proof of Theorem 3.3

This proof uses the following result about bounded queries ([25], see also [6]): Let $\#_n^A(p_1,\ldots,p_n)$ denote the number of elements in the intersection $A\cap\{p_1,\ldots,p_n\}$, and let $F\in EN(m)$ mean that there is an algorithm that will, given x, enumerate $\le m$ possibilities for $F(z)$ one of which is correct. It is know that if A is not recursive, then $\#_n^A\notin EN(n)$. In other words, we cannot always eliminate one possibility for $\#_n^A$.

Assume that for some m, there is an algorithm which returns the set $S(f,B)$ with $\#S(f,B)<m$. Let us then show that for the halting set A, we get $\#_{m-1}^A\in E(m-1)$ – in contradiction with the above result. To show this, we will follow the construction from [26], in which we construct, for every program p, a non-negative computable real number x such that $x>0$ if and only if p halts. (This construction follows the spirit of Turing – who introduced the notion of a Turing machine in an attempt to formalize the notion of a computable real number.) Specifically, for every program p, we define x as follows: $x_k=2^{-k}$ if p did not halt by time k, and $x_k=2^{-t}$ if p halted at some moment $t\le k$.

For every $m-1$ Turing machines (or programs) $p_1,\ldots,p_{m-1}$, we can then define the corresponding real numbers $a_1,\ldots,a_{m-1}$ and a computable function

$$f(x)\stackrel{\text{def}}{=}t(x)+(1-a_1)\cdot t(x-1)+\ldots+(1-a_{m-1})\cdot t(x-(m-1)),$$

where $t(x)=x$ for $x\in[0,0.5]$, $t(x)=1-x$ for $x\in[0.5,1]$, and $t(x)=0$ for all other x. This function consists of m triangular-shaped pieces of heights 1 (for $x=0.5$), $1-a_1$ (for $x=1.5$), $\ldots$, $1-a_{m-1}$ (for $x=m-0.5$). Since $a_i\ge 0$, the maximum of this function is 1, and the number of points where this maximum is attained is equal to the number of values $a_1,\ldots,a_{m-1}$ which are equal to 0. Since $a_i=0$ if and only if the program p_i halts, any limitation of the number of points where f attains maximum would thus lead to a limitation on $\#_{m-1}^A$. The theorem is proven.

4.4 Proof of Theorem 3.4

This proof is similar to the proof of Theorem 3.3, with the only difference that instead of the function $f(x)$, we consider the computable system consisting of

a single equation $f_1(x) = 0$ on the interval $[1, m]$, where

$$f_1(x) \stackrel{\text{def}}{=} (1 - a_1) \cdot t(x - 1) + \ldots + (1 - a_m) \cdot t(x - m) - 1,$$

and $t(x)$ is defined as in the proof of Theorem 3. The only possible solutions to this equation are the values $i + 0.5$ where $a_i = 0$ (i.e., where the Turing machine p_i halts), so the existence of $S(s)$ with $\#S(s) < m + 1$ would lead to $\#_m^A \in EN(m)$ and hence, to a contradiction. Q.E.D.

ACKNOWLEDGEMENTS. This work was supported in part by NSF grants HRD-0734825, EAR-0225670, and EIA-0080940, by Texas Department of Transportation grant No. 0-5453, by the Japan Advanced Institute of Science and Technology (JAIST) International Joint Research Grant 2006-08, and by the Max Planck Institut für Mathematik.

References

[1] M.J. Beeson, *Foundations of constructive mathematics*, Springer-Verlag, N.Y., 1985.

[2] R. Beigel, H. Buhrman, and L. Fortnow, "NP might not be as easy as detecting unique solutions", In: *Proceedings of the 30th ACM Symposium on the Theory of Computing*, 1998, pp. 203–208.

[3] E. Bishop, *Foundations of Constructive Analysis*, McGraw-Hill, 1967.

[4] E. Bishop and D.S. Bridges, *Constructive Analysis*, Springer, N.Y., 1985.

[5] D.S. Bridges, *Constructive Functional Analysis*, Pitman, London, 1979.

[6] W.I. Gasarch and G.A. Martin, *Bounded Queries in Recursion Theory*, Birkhäuser, Boston, 1998.

[7] D.S. Johnson, The NP-completeness column, *Journal of Algorithms*, **6** (1985) 291 - 305.

[8] R.B. Kearfott, Some tests of generalized bisection, *ACM Trans. Math. Softw.*, **13** (1987) 197 - 220.

[9] R.B. Kearfott, Interval arithmetic techniques in the computational solution of nonlinear systems of equations: Introduction, examples, and comparisons, In: *Computational solution of nonlinear systems of equations, Proc. SIAM-AMS Summer Semin., Ft. Collins/CO (USA) 1988*, American Math. Society, Providence, RI, Lectures in Appl. Math., **26** (1990) 337 - 357.

[10] R.B. Kearfott, Interval Newton/generalized bisection when there are singularities near roots, *Ann. Oper. Res.*, **25**, No. 1–4 (1990) 181 - 196.

[11] R.B. Kearfott, Preconditioners for the interval Gauss-Seidel method, *SIAM J. Numer. Anal.*, **27**, No. 2 (1990) 804 - 822.

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

[13] U. Kohlenbach. *Theorie der Majorisierbaben ...*, Ph.D. Dissertation, Frankfurt am Main, 1990.

[14] U. Kohlenbach, Effective moduli from ineffective uniqueness proofs. An unwinding of de La Vallée Poussin's proof for Chebycheff approximation, *Annals for Pure and Applied Logic*, **64**, No. 1 (1993), 27 - 94.

[15] U. Kohlenbach, *Applied Proof Theory: Proof Interpretations and their Use in Mathematics*, Springer Verlag, Berlin-Heidelberg, 2008.

[16] V. Kreinovich, What does the law of the excluded middle follow from?, *Journal of Soviet Mathematics*, **8** , No. 1 (1977) 266 - 271.

[17] V. Kreinovich, *Complexity measures: computability and applications*, Master Thesis, Leningrad University, Department of Mathematics, Division of Mathematical Logic and Constructive Mathematics, 1974 (in Russian).

[18] V. Kreinovich, Uniqueness implies algorithmic computability, *Proceedings of the 4th Student Mathematical Conference*, Leningrad University, Leningrad, 1975, 19 - 21 (in Russian).

[19] V. Kreinovich, Reviewer's remarks in a review of D. S. Bridges, *Constructive functional analysis*, Pitman, London, 1979; Zentralblatt für Mathematik, **401** (1979) 22 - 24.

[20] V. Kreinovich, *Categories of space-time models*, Ph.D. dissertation, Novosibirsk, Soviet Academy of Sciences, Siberian Branch, Institute of Mathematics, 1979 (in Russian).

[21] V. Kreinovich, Unsolvability of several algorithmically solvable analytical problems, *Abstracts Amer. Math. Soc.*, **1**, No. 1 (1980), 174.

[22] V.Ya. Kreinovich, *Philosophy of Optimism: Notes on the possibility of using algorithm theory when describing historical processes*, Leningrad Center for New Information Technology "Informatika", Technical Report, Leningrad, 1989 (in Russian).

[23] V. Kreinovich and R.B. Kearfott, Computational complexity of optimization and nonlinear equations with interval data, *Abstracts of the Sixteenth Symposium on Mathematical Programming with Data Perturbation*, The George Washington University, Washington, D.C., 26–27 May 1994.

[24] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational complexity and feasibility of data processing and interval computations*, Kluwer, Dordrecht, 1998.

[25] M. Kummer, A proof of Beigel's Cardinality Conjecture, *Journal of Symbolic Logic*, **57**, No. 2 (1992) 677 - 681.

[26] L. Longpré and V. Kreinovich, A review of [6], *Reliable Computing*, **5**, No. 2 (1999), 201 - 203.

[27] L.G. Valiant and V.V. Vazirani, NP is as easy as detecting unique solutions, *Theoretical Computer Science*, **47** (1986), 85-93.

[28] G.W. Walster, The future of intervals, In: W. Kramer and J. Wolff von Gudenberg (eds.), *Scientific Computing, Validated Numerics, Interval Methods*, Kluwer Academic/Plenum Publishers, New York, Boston, Dordrecht, London, Moscow, 2000, 1 - 15.

Received: August 06, 2007