12-1-2011

Constraint Optimization: From Efficient Computation of What Can Be Achieved to Efficient Computation of a Way to Achieve The Corresponding Optimum

Ali Jalal-Kamali
University of Texas at El Paso, ajalalkamali@miners.utep.edu

Martine Ceberio
University of Texas at El Paso, mceberio@utep.edu

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

Comments:
Technical Report: UTEP-CS-11-63

Constraint Optimization: From Efficient Computation of What Can Be Achieved to Efficient Computation of a Way to Achieve The Corresponding Optimum

Ali Jalal-Kamali, Martine Ceberio, and Vladik Kreinovich

Dept. Computer Science, University of Texas at El Paso, El Paso, TX 79968, USA
ajalalkamali@miners.utep.edu, mceberio@utep.edu, vladik@utep.edu

Abstract. In many practically useful cases, we know how to efficiently compute the exact range of a function over given intervals (and, possibly, under additional constraints). In other words, we know how to efficiently compute the minimum and maximum of a given function $f(x_1, \ldots, x_n)$ on any box. From the practical viewpoint, it is important not only to find the value of the corresponding maximum or minimum, but also to know for what values of the parameters x_i this optimum is attained. We prove a general result: that if we can efficiently compute the optimum, then we can also efficiently find the values at which this optimum is attained.

1 From Computing Maximum Value to Locating Where Maximum Is Attained

Need for optimization: general reminder. In many practical situations, we need to select the best alternative: a location of a plant, values of the control to apply to a system, etc. Let n be the total number of parameters $x_1, \ldots, x_n$ which are needed to uniquely determine the alternative. For each parameter x_i, we know the range $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$ of its possible values. The "best" alternative" is defined as the one for which an appropriate objective function $f(x_1, \ldots, x_n)$ attains the largest possible value. It is reasonable to assume that the objective function is feasibly computable. Then, the problem is to find the best values $x_1, \ldots, x_n$ for which the objective function attains the largest possible value.

First step: computing the largest possible value of the objective function. Before we start solving the above decision problem, it makes sense to first solve a simpler problem: finding out what we can, in principle, achieve within the given setting.

This checking is often useful: once we (almost) exhausted the possibilities of an idea, it may be better to look for new ideas: e.g., instead of trying to further minimize the pollution caused by a coal-burning steam engine, it may be better to use a different, less polluting engine design.

In precise terms, we need to compute the maximum $\overline{y}$ (and, if needed, the minimum $\underline{y}$) of the given function $f(x_1, \ldots, x_n)$ over given intervals $\mathbf{x}_i$. The

problem of computing the range $[\underline{y}, \overline{y}]$ of the function under interval constraints $x_i \in \mathbf{x}_i$ is known as the problem of *interval computations*; see, e.g., [2].

Comment. The actual minimum and maximum $\underline{y}$ and $\overline{y}$ are, in general, irrational numbers and thus, cannot be exactly represented in the current computers. So, what we need is, given any rational number $\varepsilon > 0$, compute the optima with accuracy ε, i.e., compute rational numbers $\underline{r}$ and $\overline{r}$ for which $|\underline{r} - \underline{y}| \leq \varepsilon$ and $|\overline{r} - \overline{y}| \leq \varepsilon$.

Interval computation is, in general, NP-hard. It is known that in general, the problem of computing the corresponding range is NP-hard; see, e.g., [1]. This means, crudely speaking, that it is not possible to have a feasible algorithm that would always compute the desired range. Because of this, it is important to find practically useful classes of problems for which it is feasibly possible to compute this range. Many such classes are known.

Need to take additional constraints into account. In practice, in addition to the constraints $\underline{x}_i \leq x_i \leq \overline{x}_i$, we often have additional constraints of equality or inequality type. In such situations, it is necessary to restrict ourselves only to values $(x_1, \ldots, x_n)$ which satisfy these constraints.

Need to find location of the maximum. Once we know the value of the maximum, it is necessary to compute the values of the parameters $x_1, \ldots, x_n$ that lead to this maximum value.

How this is done now. At present, once we have developed an algorithm for computing the maximum value $\overline{y}$ of the given function $f(x_1, \ldots, x_n)$, we need to develop a second algorithm – for finding the values $x_1, \ldots, x_n$ at which this maximum is attained.

What we do. In this paper, we describe a general technique for generating the second algorithm once the first one is known.

2 Main Result: If We Can Feasibly Compute the Maximum, Then We Can Also Feasibly Locate Where This Maximum Is Attained

Constraint result: formulation. Let us assume that we can feasibly compute the maximum. To be precise, we assume that for some class $\mathcal{F}$ of functions $f(x_1, \ldots, x_n)$ and for some class $\mathcal{C}$ of constraints, there exists a feasible (polynomial time) algorithm that, given:

- a function $f(x_1, \ldots, x_n) \in \mathcal{F}$ and constraints $C \in \mathcal{C}$,
- rational-valued intervals $[\underline{x}_1, \overline{x}_1], \ldots, [\underline{x}_n, \overline{x}_n]$, and
- a rational number $\varepsilon > 0$,

computes rational values $\underline{r}$ and $\overline{r}$ which are ε-close to the endpoints $\underline{y}$ and $\overline{y}$ of the range

$$[\underline{y}, \overline{y}] = \{f(x_1, \ldots, x_n) : x_1 \in [\underline{x}_1, \overline{x}_1], \ldots, x_n \in [\underline{x}_n, \overline{x}_n], (x_1, \ldots, x_n) \in C\}$$

of the given function under given constraints. Our result is that in this case, we can feasibly locate the point where the maximum is attained. To be precise, we describe a feasible algorithm that, given:

- a function $f(x_1, \ldots, x_n) \in \mathcal{F}$ and constraints $C \in \mathcal{C}$,
- rational-valued intervals $[\underline{x}_1, \overline{x}_1], \ldots, [\underline{x}_n, \overline{x}_n]$, and
- rational numbers $\varepsilon > 0$ and $\delta > 0$,

computes rational values $r_1, \ldots, r_n$ for which there exist δ-close values $x_1, \ldots, x_n$ which satisfy the constraints C and for which $f(x_1, \ldots, x_n) \geq \overline{y} - \varepsilon$.

Mathematical comment. In general, the maximum of a function $f(x_1, \ldots, x_n)$ can be attained at values x_i which are not rational and therefore, cannot be exactly represented in a usual computer. From this viewpoint, the fact that we produce δ-approximations to these points makes perfect sense – in general.

Interval-related comments. In situations when there are no additional constraints and we can also feasibly compute the bound M on all partial derivatives of a function f, we can also feasibly produce, given a rational number $\eta > 0$, rational values $r_1, \ldots, r_n$ for which $f(r_1, \ldots, r_n) \geq \overline{y} - \eta$.

At first glance, the fact that we can use interval computations to locate the maximum is not surprising – such a location is one of the main applications of interval computations in optimization; see, e.g., [2]. The main idea is that we use interval computations to find the enclosure of a function on subboxes while keeping track of its values in the subboxes' midpoints, and then dismiss the subboxes for which the upper bound is smaller than the maximum so far (= the maximum of all values computed at midpoints). The main difference between this idea and what we are proposing is that the traditional idea does not necessarily lead to a feasible algorithm – computations can take exponential time by requiring us to consider 2^n sub-boxes – while the computation time for our algorithm is always feasible (polynomial).

Constraints-related comment. Our result is in line with the fact that solving a constraint *satisfaction* problem – of finding some values that satisfy given constraints – is usually simpler than solving a constraint *optimization* problem – of finding, among all possible solutions of a constraint satisfaction problem, the one withe the largest value of a given objective function $f(x_1, \ldots, x_n)$. Once we can compute the actual constraint maximum $\overline{y}$ of the objective function, the problem of finding the values where the maximum is attained can be solved by adding an additional constraint $f(x_1, \ldots, x_n) = \overline{y}$ (or $f(x_1, \ldots, x_n) \geq \overline{y} - \varepsilon$) to the original list of constraints. Thus, to locate the maximum, it is sufficient to solve an easier-to-solve constraint satisfaction problem.

Description of the algorithm. At each stage of this algorithm, we will have a box B_k. We start with the original box $B_0 = B$. Then, we will decrease the x_1-size of this box in half several times until this size becomes smaller than or equal to 2δ. (In the following text, we describe how this bisection is done.) After this, we decrease the x_2-size of this box in half, etc., until all n sizes are bounded by 2δ.

For each side, we start with the interval $[\underline{x}_i, \overline{x}_i]$ of width $w_i = \overline{x}_i - \underline{x}_i$. After s_i bisection steps, the width decreases to $w_i \cdot 2^{-s_i}$. One can see that we need $\left\lceil \ln\left(\frac{w_i}{2\delta}\right) \right\rceil$ steps to reach the desired size ($\le 2\delta$) of the i-th side. Overall, we need $s = \sum_{i=1}^{n} \ln\left(\left\lceil \frac{w_i}{2\delta} \right\rceil\right)$ bisection steps.

Each bisection step of the algorithm is as follows. Suppose that we have a box B_k with an x_i-size $[\underline{b}_i, \overline{b}_i]$, and we need to bisect the x_i-size. We divide this box in two by bisecting the i-th side into two equal intervals $[\underline{b}_i, m_i]$ and $[m_i, \overline{b}_i]$, where $m_i = \frac{\underline{b}_i + \overline{b}_i}{2}$. This divides the original box

$$B_k = \ldots \times [\underline{b}_{i-1}, \overline{b}_{i-1}] \times [\underline{b}_i, \overline{b}_i] \times [\underline{b}_{i+1}, \overline{b}_{i+1}] \times \ldots$$

into two subboxes

$$B'_k = \ldots \times [\underline{b}_{i-1}, \overline{b}_{i-1}] \times [\underline{b}_i, m_i] \times [\underline{b}_{i+1}, \overline{b}_{i+1}] \times \ldots \text{ and}$$

$$B''_k = \ldots \times [\underline{b}_{i-1}, \overline{b}_{i-1}] \times [m_i, \overline{b}_i] \times [\underline{b}_{i+1}, \overline{b}_{i+1}] \times \ldots$$

To both subboxes, we apply the original range estimation algorithm, and get two rational numbers $\overline{r}'_k$ and $\overline{r}''_k$ which are $\frac{\varepsilon}{2s}$-close to the (constraint) maxima $\overline{y}'_k$ and $\overline{y}''_k$ of the function $f(x_1, \ldots, x_n)$ over these subboxes. As the next box B_{k+1}, we then choose either B'_k or B''_k depending on which of the two rational numbers $\overline{r}'_k$ and $\overline{r}''_k$ is larger:

- if $\overline{r}'_k \ge \overline{r}''_k$, we select B'_k,
- else, we select B''_k.

Once all the sides have been decreased, we return the coordinates of the midpoint of the final box B_s as the desired values $r_1, \ldots, r_n$.

Proof that our algorithm is correct. Let $\overline{y}_k$ denote the (constraint) maximum of the function $f(x_1, \ldots, x_n)$ over the box B_k. We will prove, by induction, that for each box B_k, we have $\overline{y}_k \ge \overline{y} - \frac{k}{s} \cdot \varepsilon$. Then, after all s steps, we will be able to conclude that $\overline{y}_s \ge \overline{y} - \varepsilon$. By definition of $\overline{y}_s$, this means that there exist a point $(x_1, \ldots, x_n) \in B_s$ which satisfies the given constraints and at which $f(x_1, \ldots, x_n) = \overline{y}_s \ge \overline{y} - \varepsilon$. Since the box is of width $\le 2\delta$ in all directions, each value x_i is δ-close to the midpoint r_i. So, to prove correctness, it is sufficient to prove that $\overline{y}_k \ge \overline{y} - \frac{k}{s} \cdot \varepsilon$.

The induction base is clear: for $k = 0$, when B_0 is the original box and therefore, the maximum $\overline{y}_0$ over this box is simply equal to $\overline{y}$.

Let us prove the induction step. Assume that the desired inequality $\overline{y}_k \ge \overline{y} - \frac{k}{s} \cdot \varepsilon$ holds for the box B_k. Let us show that this inequality also holds for the next box B_{k+1}. Indeed, since the box B_k is the union of two subboxes B'_k and B''_k, the maximum $\overline{y}_k$ of the function $f(x_1, \ldots, x_n)$ over the box B_k is equal to the

largest of the two maxima $\overline{y}'_k$ and $\overline{y}''_k$ over the two subboxes: $\overline{y}_k = \max(\overline{y}'_k, \overline{y}''_k)$. For computed approximate maxima $\overline{r}'_k$ and $\overline{r}''_k$, we have $\overline{r}'_k \geq \overline{y}'_k - \dfrac{\varepsilon}{2s}$ and $\overline{r}''_k \geq \overline{y}''_k - \dfrac{\varepsilon}{2s}$. Thus, we have

$$\max(\overline{r}'_k, \overline{r}''_k) \geq \max(\overline{y}'_k, \overline{y}''_k) - \frac{\varepsilon}{2s} = \overline{y}_k - \frac{\varepsilon}{2s}.$$

According to our algorithm, we select the box B_{k+1} for which the corresponding estimate of the maximum is the largest, i.e., for which this estimate is equal to $\overline{r}_{k+1} = \max(\overline{r}'_k, \overline{r}''_k)$. Thus, we conclude that $\overline{r}_{k+1} \geq \overline{y}_k - \dfrac{\varepsilon}{2s}$. Since $\overline{y}_{k+1}$ is $\dfrac{\varepsilon}{2s}$-close to the estimate $\overline{r}_{k+1}$, we conclude that

$$\overline{y}_{k+1} \geq \overline{r}_{k+1} - \frac{\varepsilon}{2s} \geq \left(\overline{y}_k - \frac{\varepsilon}{2s}\right) - \frac{\varepsilon}{2s} = \overline{y}_k - \frac{\varepsilon}{s}.$$

So, from $\overline{y}_k \geq \overline{y} - \dfrac{k}{s} \cdot \varepsilon$, we can now conclude that

$$\overline{y}_{k+1} \geq \overline{y}_k - \frac{\varepsilon}{s} \geq \left(y - \frac{k}{s} \cdot \varepsilon\right) - \frac{\varepsilon}{s} = y - \frac{k+1}{s} \cdot \varepsilon.$$

The inequality is proven, and so is the algorithm's correctness.

Proof that our algorithm is feasible. The number of steps s feasibly (polynomially) depends on the size of the input; the range estimation algorithm that we use on each step is also polynomial-time. Thus, all we do is repeat a polynomial-time algorithm polynomially many times. The computation time of the resulting algorithm is bounded by the product of the two corresponding polynomials and is, thus, itself polynomial. Thus, our algorithm is indeed feasible.

What if know the bounds on the derivatives. In this case, since

$$|f(r_1, \ldots, r_n) - f(x_1, \ldots, x_n)| \leq \sum_{i=1}^{n} \left|\frac{\partial f}{\partial f_i}\right| \cdot |x_i - r_i| \leq n \cdot M \cdot \delta,$$

we conclude that $f(r_1, \ldots, r_n) \geq f(x_1, \ldots, x_n) - n \cdot M \cdot \delta \geq \overline{y} - \varepsilon - n \cdot M \cdot \delta$. So, if we want to find the values $r_1, \ldots, r_n$ for which $f(r_1, \ldots, r_n) \geq \overline{y} - \eta$, it is sufficient to apply the above algorithm with $\varepsilon = \dfrac{\eta}{2}$ and $\delta = \dfrac{\eta}{2 \cdot n \cdot M}$ (so that $n \cdot M \cdot \delta = \eta/2$).

Acknowledgments. This work was supported in part by the National Science Foundation grants HRD-0734825 and DUE-0926721, by Grant 1 T36 GM078000-01 from the National Institutes of Health.

References

1. Kreinovich, V., Lakeyev, A., Rohn, J., Kahl, P.: Computational complexity and feasibility of data processing and interval computations, Kluwer, Dordrecht (1998)
2. Moore, R. E., Kearfott, R. B., Cloud, M. J.: Introduction to Interval Analysis, SIAM Press, Philadelphia, Pennsylvania (2009)