

Towards an Efficient Bisection of Ellipsoids

Paden Portillo, Martine Ceberio, and Vladik Kreinovich

Department of Computer Science
University of Texas at El Paso, El Paso, TX 79968, USA,
pportillo2@miners.utep.edu, {vladik,mceberio}@utep.edu

Abstract. Constraints are often represented as ellipsoids. One of the main advantages of such constraints is that, in contrast to boxes, over which optimization of even quadratic functions is NP-hard, optimization of a quadratic function over an ellipsoid is feasible. Sometimes, the area described by constraints is too large, so it is reasonable to bisect this area (one or several times) and solve the optimization problem for all the sub-areas. Bisecting a box, we still get a box, but bisecting an ellipsoid, we do not get an ellipsoid. Usually, this problem is solved by enclosing the half-ellipsoid in a larger ellipsoid, but this slows down the domain reduction process. Instead, we propose to optimize the objective functions over the resulting half-, quarter, etc., ellipsoids.

Keywords: constraints; ellipsoids; bisection; computational complexity

Constraints on a single variable. In many practical problems, we have prior *constraint* on the values of different quantities. For each individual quantity x , we usually know a lower bound \underline{x} and an upper bound. Thus, we know that the actual value of this quantity must lie within the interval $[\underline{x}, \bar{x}]$.

Sometimes, we know several lower bounds; in this case, we take the largest of them as \underline{x} . Similarly, if we know several upper bounds, we can take the smallest of these upper bounds as \bar{x} .

Correspondingly, when are looking for a value that satisfies a certain condition (e.g., when we are solving an equation), or if we are looking for the best option (i.e., solving an appropriate optimization problem), we should take this constraint into account. For example, when finding the optimal value of x , we should optimize the corresponding objective function $f(x)$ under the given constraint on x – i.e., under the constraint that $\underline{x} \leq x \leq \bar{x}$.

Constraints on several variables: boxes naturally appear. Usually, we have several different variables x_1, \dots, x_n . For each of these variables x_i , we usually know a lower bound \underline{x}_i and an upper bound \bar{x}_i . Thus, we know that the actual value of the tuple $x = (x_1, \dots, x_n)$ belongs to the *box* $[\underline{x}_1, \bar{x}_1] \times \dots \times [\underline{x}_n, \bar{x}_n]$. Such box constraints and box uncertainty are typical for *interval computations*; see, e.g., [8, 9, 12].

Problem with box constraints: computational complexity. The main problem with box constraints is that already for quadratic objective functions

$$f(x_1, \dots, x_n) = a_0 + \sum_{i=1}^n a_i \cdot x_i + \sum_{i=1}^n \sum_{j=1}^n a_{ij} \cdot x_i \cdot x_j,$$

optimizing them over a box is, in general, an NP-hard problem; see, e.g., [11, 15]. This problem is computational complex not for some exotic quadratic function: as shown in [5], it is actually even NP-hard for the sample variance

$$V(x_1, \dots, x_n) = \frac{1}{n} \cdot \sum_{i=1}^n x_i^2 - \left(\frac{1}{n} \cdot \sum_{i=1}^n x_i \right)^2.$$

Informal explanation of computational complexity. The above computational complexity can be intuitively explained.

Indeed, a function of one variable $f(x)$ attains its optimum (maximum or minimum) on an interval $[\underline{x}, \bar{x}]$ either at one of its endpoints, or at an internal point $x \in (\underline{x}, \bar{x})$. If this optimum is attained at an internal point, then at this point, a derivative $\frac{df}{dx}$ should be equal to 0. Thus, to find the largest and the smallest value of a function $f(x)$ on the interval $[\underline{x}, \bar{x}]$, it is sufficient to consider its value at the endpoints \underline{x} and \bar{x} and at a point x where $\frac{df}{dx} = 0$. When the function $f(x)$ is quadratic, its derivative is a linear function and therefore (unless we have a degenerate case) there is only one point where the derivative is equal to 0. So, to find the optimum of a quadratic function of one variable, it is sufficient to consider at most three values x (two if the point where the derivative is 0 lies outside the given interval).

For optimizing a function $f(x_1, \dots, x_n)$ of several variables on the box $\underline{x}_i \leq x_i \leq \bar{x}_i$, the same trichotomy holds for each of the variables x_i : with respect to this variable, the optimum is attained either at one of the endpoints \underline{x}_i and \bar{x}_i and at a point x where the corresponding partial derivative is equal to 0 $\left(\frac{\partial f}{\partial x_i} = 0 \right)$.

For each variable, we have only 3 options, but together, they form $3 \times \dots \times 3 = 3^n$ options: e.g., when $x_1 = \underline{x}_1$, we still have 3 different options for x_2 , etc. For each of these 3^n combinations of options, we have a system of linear equations to solve – which is relatively easy (see, e.g., [4]), but the sheer amount of such cases makes this straightforward calculus-based algorithm exponential in time. The NP-hardness results proves, in effect, that unless $P=NP$, no other algorithm can solve this problem much faster (in feasible polynomial time).

Ellipsoids: a solution to the computational complexity problem. One known solution to the above computational complexity problem is to use *ellipsoid* constraints instead of the boxes, i.e., to use constrains of the type $J(x_1, \dots, x_n) \leq J_0$,

where

$$J(x_1, \dots, x_n) = b_0 + \sum_{i=1}^n b_i \cdot x_i + \sum_{i=1}^n \sum_{j=1}^n b_{ij} \cdot x_i \cdot x_j.$$

For such constraints, optimizing a quadratic function $f(x)$ means that:

- either the optimum is attained inside the ellipsoid, then we have a system of linear equations $\frac{\partial f}{\partial x_i} = 0$;
- or the optimum is attained on the border $J(x_1, \dots, x_n) = J_0$ of the ellipsoid.

In the second case, the Lagrange multiplier approach leads to the unconstrained optimization of the auxiliary quadratic function $f + \lambda \cdot (J - J_0)$, i.e., again, to a solution of a system of linear equations. As a result, we get a solution $x(\lambda)$ as a function of λ .

The only additional problem is to find a single variable λ . This can be done in a relatively straightforward way, by solving an equation $J(x(\lambda)) = J_0$ with one unknown λ . The complexity of solving such an equation does not grow with the size n of the problem.

Computations can be made even more explicit if we take into account that if we have two quadratic forms, one of which is positive definite, we can move both to a diagonal form by applying an appropriate linear transformation; this linear transformation can be easily computed; see, e.g., [4]. Thus, when we apply an appropriate linear transformations of the coordinates, in the new coordinates y_1, \dots, y_n , the ellipsoid $J \leq J_0$ becomes simply a unit circle $\sum_{i=1}^n y_i^2 = 1$, and the objective function takes the form

$$f(y_1, \dots, y_n) = w_0 + \sum_{i=1}^n w_i \cdot y_i + \sum_{i=1}^n w_{ii} \cdot y_i^2.$$

In this case, the Lagrange functional takes the form

$$f_\lambda = w_0 + \sum_{i=1}^n w_i \cdot y_i + \sum_{i=1}^n w_{ii} \cdot y_i^2 + \lambda \cdot \left(\sum_{i=1}^n y_i^2 - 1 \right),$$

so equating partial derivatives of f_λ to 0 leads to $w_i + 2w_{ii} \cdot y_i + 2\lambda \cdot y_i = 0$, i.e., to $y_i = -\frac{w_i}{2 \cdot (w_{ii} + \lambda)}$, and the equation for λ takes the following explicit form:

$$\sum_{i=1}^n \frac{w_i^2}{4 \cdot (w_{ii} + \lambda)^2} = 1.$$

Because of this drastic reduction in computational complexity, ellipsoids have been successfully used in many applications; see, e.g., [1–3, 6, 7, 13, 14].

Need for bisection. When the optimized function is simple – e.g., linear or quadratic – it does not matter how big or small is the area, the algorithm is the same. However, when the objective function is more complex, then for small areas, we can expand the objective function into Taylor series

$$f(x_1, \dots, x_n) = a_0 + \sum_{i=1}^n a_i \cdot x_i + \sum_{i=1}^n \sum_{j=1}^n a_{ij} \cdot x_i \cdot x_j + \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n a_{ijk} \cdot x_i \cdot x_j \cdot x_k + \dots,$$

and, with reasonable accuracy, keep only quadratic terms in this expansion.

For larger areas, such an approximation may not necessarily be sufficiently accurate. A similar problem occurs when we consider domains described by boxes. For boxes, a solution to this problem is straightforward: we divide (“bisect”) the box into two sub-boxes by dividing one of the side intervals $[x_i, \bar{x}_i]$ into two by a line $x_i = \tilde{x}_i \stackrel{\text{def}}{=} \frac{x_i + \bar{x}_i}{2}$. (In n -dimensional space, the equation $x_i = \tilde{x}_i$ describes a plane.)

We can then try to estimate the optimum of the function over both sub-boxes, and, if necessary, further bisect each of these two sub-boxes into sub-sub-boxes [9, 12].

Bisection for ellipsoids: a problem. With ellipsoids, we can apply a similar idea: divide the ellipsoid into two halves by an appropriate plane. However, in comparison to boxes, here, we have an additional problem:

- when we bisect a box, both halves are boxes;
- however, a half of an ellipsoid is not an ellipsoid.

Thus, even when we know the algorithms for optimizing quadratic functions over ellipsoids, we cannot use them to optimize functions over half- or quarter-ellipsoids.

How this problem is solved now. At present (see, e.g., [2, 3, 10]), this problem is solved by enclosing each of the resulting half-ellipsoids into an ellipsoid. This procedure enables us to apply the same optimizations as before, but it comes with a price – that the enclosures are larger than the halves and thus, the size of the regions decreases slower than in the case of boxes – where, e.g., the volume of an area decreases by 2 on each bisection step. Since the areas do not go as fast, we will need more iterations (and thus, more computation time) to reach the desired small size.

Our proposal. As an alternative, we propose to explicitly optimize quadratic functions over half-, quarter-, etc. ellipsoids.

Indeed, suppose that after a small number of bisections d , we have the resulting region. Each bisection j , $1 \leq j \leq d$, corresponds to selecting a half-space. Each half-space can be described by a linear inequality $\ell_j(x) \leq 0$, with a linear function $\ell_j(x)$. As before, for each j , the optimum is attained either inside the half-space or on its border, at a plane $\ell_j(x) \leq 0$. Thus, to find the desired

optimum, we must check all 2^d subsets of the set $\{1, \dots, d\}$. For each of these subsets S , we take all the planes $\ell_j(x) = 0$ with $j \in S$. The intersection of all these planes with the original ellipsoid is still an ellipsoid of smaller dimension. We then use the known ellipsoid-optimization algorithm to optimize the objective function over this smaller-dimension ellipsoid. The largest or smallest of the desired values is the desired maximum or minimum of the original objective function over our domain.

When d is small, the value 2^d is also small, so we still get an efficient algorithm.

Acknowledgments. This work was supported in part by NSF grants HRD-0734825 and DUE-0926721, and by Grant 1 T36 GM078000-01 from NIH.

References

1. Belforte, G., Bona, B.: An improved parameter identification algorithm for signal with unknown-but-bounded errors. Proceedings of the 7th IFAC Symposium on Identification and Parameter Estimation, York, U.K. (1985)
2. Chernousko, F.L.: Estimation of the Phase Space of Dynamic Systems. Nauka publ., Moscow (1988) in Russian
3. Chernousko, F.L.: State Estimation for Dynamic Systems. CRC Press, Boca Raton, Florida (1994)
4. Cormen, C.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, MIT Press, Boston, Massachusetts (2009)
5. Ferson S., Ginzburg, L., Kreinovich, V., Longpré, L., Aviles, M.: Exact bounds on finite populations of interval data. *Reliable Computing* 11(3), 207–233 (2005)
6. Filippov, A.F.: Ellipsoidal estimates for a solution of a system of differential equations. *Interval Computations* 2, No. 2(4), 6–17 (1992)
7. Fogel, E., Huang, Y.F.: On the value of information in system identification. Bounded noise case. *Automatica* 18(2), 229–238 (1982)
8. Interval computations website
<http://www.cs.utep.edu/interval-comp>
9. Jaulin, L., Kieffer, M., Didrit, O., Walter, E.: Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics. Springer-Verlag, London (2001)
10. Karmarkar, N.: A new polynomial-time algorithm for linear programming. *Combinatorica* 4, 373–396 (1984)
11. Kreinovich, V., Lakeyev, A., Rohn, J., Kahl, P.: Computational Complexity and Feasibility of Data Processing and Interval Computations. Kluwer, Dordrecht (1998)
12. Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to Interval Analysis. SIAM Press, Philadelphia, Pennsylvania (2009)
13. Schweppe, F.C.: Recursive state estimation: unknown but bounded errors and system inputs. *IEEE Transactions on Automatic Control* 13, 22 (1968)
14. Schweppe, F.C.: Uncertain Dynamic Systems, Prentice Hall, Englewood Cliffs, New Jersey (1973)
15. Vavasis, S.A.: Nonlinear Optimization: Complexity Issues. Oxford University Press, New York (1991)