Journal of Uncertain Systems
Vol.12, No.x, pp.xx-xx, 2018
Online at: www.jus.org.uk

How to Efficiently Compute Ranges Over a Difference Between Boxes, With Applications to Underwater Localization

Luc Jaulin[1], Martine Ceberio[2],
Olga Kosheleva[2], and Vladik Kreinovich[2]
[1] *Bureau D214, ENSTA-Bretagne, 2, rue François Verny*
29806, Brest, Cedex 9 France, luc.jaulin@ensta-bretagne.fr
[2] *University of Texas at El Paso, El Paso, TX 79968, USA*
mceberio@utep.edu, olgak@utep.edu, vladik@utep.edu

Received 1 March 2018; Revised 31 March 2018

Abstract

When using underwater autonomous vehicles, it is important to localize them. Underwater localization is very approximate. As a result, instead of a single location x, we get a set X of possible locations of a vehicle. Based on this set of possible locations, we need to find the range of possible values of the corresponding objective function $f(x)$. For missions on the ocean floor, it is beneficial to take into account that the vehicle is in the water, i.e., that the location of this vehicle is *not* in a set X' describing the under-floor matter. Thus, the actual set of possible locations of a vehicle is a difference set $X - X'$. So, it is important to find the ranges of different functions over such difference sets. In this paper, we propose an effective algorithm for solving this problem.

Keywords: underwater localization, interval uncertainty

1 Underwater Localization: A Practical Problem Where There Is a Need to Compute the Range Over a Difference Between Sets

Underwater engineering is important:

- Underwater exploration of minerals is becoming more and more important for or civilization.
- Underwater cables are a crucial part of the global information infrastructure.
- Underwater food resources constitute an important part of our ration.
- Underwater geological studies are needed to better predict underwater earthquakes and resulting tsunamis, etc.

In all these cases, human exploration is difficult, so more and more attention is paid to autonomous devices.

Underwater localization is important but difficult. For these devices to adequately fulfil their missions, it is important to know the device's location as exactly as possible. Such a localization, however, is not easy:

- This location cannot be easily determined by measuring the forces produced by the device'e engine, since strong currents also affect the position of a device.
- On the surface, location can be very accurately provided by the GPS sensors, sensors that relay on the propagation of electromagnetic waves. However, electromagnetic waves only propagate to the very surface of the ocean.

So, to locate an underwater device, we need to rely on other means. For example:

- we can use sound propagation, or
- we can use gyroscopes.

The resulting localization is not perfect:

- Due to the inhomogeneity of the ocean, sound waves do not follow the straight lines. As a result, corresponding measurements are not very accurate.
- Gyroscopes provide very accurate description of acceleration. However, when we integrate twice to transform these accelerations into coordinates, small measurement errors accumulate. Hence, the resulting coordinate estimates are also not very accurate.

Need for set-valued uncertainty. In all these cases, instead of a *single* location $x = (x_1, x_2, x_3)$, we get a *region* X of possible locations.

When we know the exact accuracy of each measuring instrument, we get a *set* X; see, e.g., [1, 2, 3, 5, 7, 10, 11, 12, 13, 14].

Need to find the range of a function. Underwater autonomous vehicles have missions. Because of this, we are not just interested in finding the coordinates of the vehicle, we want to be able to gauge how this location affects the corresponding mission.

For example, if we want the vehicle to repair an underwater cable, we need to know how far is the vehicle from the cable:

- if it is close, we can start the mission,
- if there is a possibility that the vehicle is not sufficiently close, we need to activate the vehicle's engine to bring it closer to the cable's location.

If the mission is to correct some mineral deposits, then we can compare:

- the information about the current location of the vehicle with
- the known map of the deposits

to decide where to move the vehicle so as to extract the largest amount of the desired mineral deposits on this particular mission.

In all these applications, we have an objective function $f(x)$ depending on the vehicle's location:

- it can be the distance between the location x and the location of the cable;
- it can be the amount of mineral deposits that we can extract in the vicinity of the location x, etc.

We do not know the exact location x, we only know the region X. Thus, instead of the exact value of the objective function $f(x)$, we have a range

$$f(X) \stackrel{\text{def}}{=} \{f(x) : x \in X\}$$

of the function $f(x)$ on this region.

Need to consider ranges over differences between sets. As we have mentioned earlier, underwater localizations are not very accurate. As a result, the corresponding region X is rather big.

In many missions, the vehicle operates close to the ocean floor. For such missions, we can decrease the size of the corresponding region by taking into account that the vehicle is in the water. Indeed, in many locations, we have a good map of the materials below the ocean:

- Such maps – supplemented by the corresponding geophysical information – are often available, especially when the vehicle's mission involves mineral extraction.
- Such maps are also unusually available in the vicinity of underwater cables: when engineers select a location for the cable, they want to make sure that the cable is not above a highly seismic zone, where frequent earthquakes will disrupt the communications.

A simple example can illustrate the fact that such an additional information decreases the size of the region – and thus, makes the vehicle location more accurate. Indeed, suppose that the vehicle is located on a flat floor, and we know the measurement accuracy ε.

- Based on the measured location $\widetilde{x}$, the only thing that e can conclude based on this measurement result is that the actual location x of the vehicle is within distance ε from $\widetilde{x}$. In other words, the corresponding region X of possible locations is a ball (sometimes called sphere) of radius ε with center at the point $\widetilde{x}$.
- If the location $\widetilde{x}$ is exactly on the ocean floor, then the fact that the vehicle is in the water means that we only need to consider locations above the floor. In other words, instead of the whole ball X, we now have only a *half* of the original ball.

In general, if we know:

- a set X of possible locations obtained from measurements, and
- a set X' describing the region below the ocean floor,

then we can conclude that the location x of the vehicle is:

- in the set X, but
- not in the set X'.

In other words, the set of possible locations of the underwater vehicle is the *difference set*

$$X - X' \stackrel{\text{def}}{=} \{x : x \in X \,\&\, x \notin X'\}.$$

Thus, it is important to find the range of the objective function over a different between two sets.

2 From Sets to Boxes: Why New Methods Are Needed

The existing methods of computing the range over a set are based on reducing set to boxes.

Interval computations: a brief reminder. Specifically, as the main tool, these methods use the methods of *interval computations* (see, e.g., [4, 8, 9]), i.e., methods that estimate the range of a given function $f(x) = f(x_1, \ldots, x_n)$ over a *box*

$$[\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_n, \overline{x}_n].$$

Interval computations is NP-hard, hence results are usually approximate. It should be emphasized that the problem of estimating a range of a function over a box is, in general, NP-hard – even for quadratic functions; see, e., [6].

This means, that, unless P=NP (which most computer scientists believe to be impossible), we cannot have a feasible algorithm that always computes the exact range. In other words, every feasible algorithm computes only an approximation to the desired range.

How to improve the estimates. In general:

- the smaller the box,
- the more accurate is the corresponding estimate.

So, if our estimate is too crude, one way to improve the estimate is to:

- divide the box X into several sub-boxes:

$$X = X_1 \cup \ldots \cup X_m;$$

- estimate the range of the function $f(x)$ on each of these sub-boxes, and then
- take the union of the resulting estimates:

$$f(X) = f(X_1) \cup \ldots \cup f(X_m).$$

The last step is straightforward: each range $f(X_i)$ is an interval $[\underline{y}_i, \overline{y}_i]$. Thus, the union $f(X) = [\underline{y}, \overline{y}]$ of these sets is also an interval, in which:

- the lower endpoint of the union is the smallest of the corresponding lower endpoints, and
- the upper endpoint of the union is the largest of the corresponding upper endpoints:

$$f(X) = [\min(\underline{y}_1, \ldots, \underline{y}_m), \max(\overline{y}_1, \ldots, \overline{y}_m)].$$

Boxes are usually disjoint. As we have mentioned, the larger the boxes, the less accurate the estimation. One way to decrease the size of the boxes is to make sure that they are disjoint, i.e., to be more precise, they only intersect by their faces. Indeed:

- if the two boxes intersect,
- then we can decrease one of these boxes – by deleting common area – and still cover the whole area X.

Reduction to boxes. The above idea also helps to compute the range of a function over an arbitrary set. Namely, we can do the following (see, e.g., [4]):

- first, we approximate the set X by a union of several boxes $X_1, \ldots, X_m$:

$$X \subseteq X_1 \cup \ldots \cup X_m;$$

- then, we use interval techniques to find the ranges

$$f(X_1), \ldots, f(X_m)$$

of the function $f(x)$ over all these boxes,

- and, finally,we compute the union of these ranges:

$$f(X) = f(X_1) \cup \ldots \cup f(X_m).$$

So how do we deal with set difference? From this viewpoint:

- if we know that the actual set of possible values is the set difference between two sets $X - X'$,
- what practitioners usually do is approximate the set difference by a union of boxes.

Problem: naive approach can lead to an unfeasible number of boxes. The problem with this approach is that if we do it naively, we end up with exponentially many boxes - and thus, with the exponential (hence non-feasible) time needed to process all these boxes.

Let us illustrate this problem on a simple example. Let X be a cube

$$[-2, 2] \times \ldots \times [-2, 2],$$

and let X' be a cube formed by middle halves of all the corresponding intervals:

$$X' = [-1, 1] \times \ldots \times [-1, 1].$$

To make the resulting computation most accurate, let us represent the set difference $X - X'$ as a union of disjoint boxes.

The original configuration (X, X'):

- is invariant with respect to arbitrary permutations of the variables;
- it is also invariant with respect to changing each variable x_i to $-x_i$.

It is therefore reasonable to look for a division into sub-boxes which is also invariant with respect to all these transformations.

The endpoints of the intervals forming the new sub-boxes should coincide with the borders of one of the original interval, i.e., should be equal to one of these 4 values:

- to -2,
- to -1,
- to 1, or
- to 2.

Let us show that in this case, all these intervals must be the smallest possible, i.e., must be of the type:

- $[-2,-1]$,
- $[-1,1]$, or
- $[1,2]$.

Indeed, suppose that $[-2,1]$ (or any interval containing $[-2,1]$) is one of these intervals. Then, the corresponding sub-box has the form

$$[-2,1]\times\ldots$$

- If the next interval is $[-2,-1]$ (or any interval containing $[-2,-1]$), then the sub-box has the form

  $$[-2,1]\times[-2,-1]\times\ldots$$

  By permutation, we should also have a sub-box

  $$[-2,-1]\times[-2,1]\times\ldots,$$

  but these two sub-boxes have a non-empty intersection

  $$[-2,-1]\times[-2,-1]\times\ldots,$$

  and we assumed that the sub-boxes are disjoint.

- If the next interval is $[-1,1]$ (or anything containing $[-1,1]$), then the sub-box has the form

  $$[-2,1]\times[-1,1]\times\ldots$$

  By permutation, we should also have a sub-box

  $$[-1,1]\times[-2,1]\times\ldots,$$

  but these two sub-boxes have a non-empty intersection

  $$[-1,1]\times[-1,1]\times\ldots,$$

  and we assumed that the sub-boxes are disjoint.

- Finally, if the next interval is $[1,2]$ (or any interval containing $[1,2]$), then the sub-box has the form

  $$[-2,1]\times[1,2]\times\ldots$$

  By using invariance with respect to changing the sign of x_2, we conclude that we have a sub-box

  $$[-2,-1]\times[-2,1]\times\ldots,$$

  and we already know that this is impossible.

Similarly, if $[-1,2]$ (or any interval containing $[-1,2]$) is one of these intervals, then, due to the invariance with respect to changing sign, we would conclude that $[-2,1]$ is also one of the possible intervals – and we have just shown that this is not possible.

Thus, all intervals have the form $[-2,-1]$, $[-1,1]$, or $[1,2]$. One can show that in this case, we have 3^n-1 boxes: indeed,

- by dividing each of n sides into 3 parts, we get 3^n possible sub-boxes;
- here, X' is one of these sub-boxes, so only 3^n-1 sub-boxes remain.

For large n, we get exponentially many sub-boxes, which is not feasible.

Comment.

- Of course, when $n=3$, the value 3^n-1 is simply 26, not that much. So, in principle, if we have one underwater device, we can still use this approach.
- However, for complex underwater tasks, it is often important to have several vehicles acting as a single swarm. In this case, the objective function $f(x)$ depends on the locations of all these vehicles – i.e., for v vehicles, on $3v$ coordinates. Often, the objective function depends also on the vehicle's velocities, so we have $6v$ parameters. Here, $3^{6v}-1$ can becomes very large very fast.

What we do in this paper. In this paper, we propose an efficient algorithm for computing the differences.

3 Analysis of the Problem: Reducing Difference Between Sets to Different Between Boxes

What we want: a reminder. We assume that both sets X and X' are already represented as unions of boxes:

$$X = X_1 \cup \ldots \cup X_m,$$

and

$$X' = X_1 \cup \ldots \cup X'_{m'}.$$

We want to represent the set difference $X - X'$ as a union of boxes.

Reducing the problem to difference between boxes. By definition of the set union, the fact that the actual location x belongs to the union $X = X_1 \cup \ldots \cup X_m$ means that:

- either the location x belongs to the first set X_1,
- or the location x belongs to the second set X_2,
- ...
- or the location x belongs to the last set X_m.

Taking into account that the location x cannot belong to the set X', we conclude that:

- either the location x belongs to the difference $X_1 - X'$,
- or the location x belongs to the difference $X_2 - X'$,
- ...
- or the location x belongs to the difference $X_m - X'$.

In other words,

$$X - X' = (X_1 - X') \cup \ldots \cup (X_m - X').$$

So, to describe the set difference $X - X$" as a union of boxes, it is sufficient to do the following:

- first, we represent each of the differences

$$X_1 - X', \ldots, X_m - X'$$

  as a union of boxes,

- then, we take the union of such unions.

From this viewpoint, the problem of representing the set difference $X - X'$ can be reduced to the the case when the first set X is a box, i.e., to computing the difference

$$X - (X'_1 \cup \ldots \cup X'_{m'})$$

between boxes.

Comment. It is worth mentioning that if the original boxes $X_1, \ldots, X_m$ were disjoint, then the corresponding differences

$$X_1 - X', \ldots, X_m - X'$$

are disjoint too. Thus, we do not lose anything by making this reduction.

Reducing to the case when X' is a single box. Usually, while the set X' may consist of many boxes:

- for each box X_i,
- there are only a few boxes X'_j that has non-empty intersection with X_i and are, therefore, needed to take into account when computing the set difference.

Thus, for each box X_i, we can safely assume that the set X' consists of a few boxes, i.e., that m' is small. Thus, we can compute the difference

$$X - X' = X - (X'_1 \cup \ldots \cup X'_{m'})$$

as follows:

- first, we describe the difference $X - X'_1$ as a union of boxes,
- then, for each of the resulting boxes, we represent its difference with X'_2 as a union of boxes, $\ldots$,
- finally, for each of the resulting boxes, we represent its difference with $X_{m'}$ as a union of boxes.

In view of this reduction, it is sufficient to represent the difference between the two boxes X and X' as a union of boxes.

Reformulating the difference between boxes in terms of inequalities. For a box

$$X = [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_n, \overline{x}_n],$$

the condition that the location $x = (x_1, \ldots, x_n)$ belongs to the box X means that:

- $\underline{x}_1 \le x_1 \le \overline{x}_1$, and
- $\underline{x}_2 \le x_2 \le \overline{x}_2$, and
- $\ldots$, and
- $\underline{x}_n \le x_n \le \overline{x}_n$.

Similarly, for the box

$$X' = [\underline{x}'_1, \overline{x}'_1] \times \ldots \times [\underline{x}'_n, \overline{x}'_n],$$

the condition the location $x = (x_1, \ldots, x_n)$ belongs to the box X' would mean that:

- $\underline{x}'_1 \le x_1 \le \overline{x}'_1$, and
- $\underline{x}'_2 \le x_2 \le \overline{x}'_2$, and

- ..., and
- $\underline{x}'_n \le x_n \le \overline{x}'_n$.

Thus, the condition that the location x *does not* belong to the set X' means that at least one of the following n coordinate conditions are satisfied:

- the first condition is that either $x_1 < \underline{x}'_1$ or $\overline{x}'_1 < x_1$,
- the second condition is that either $x_2 < \underline{x}'_2$ or $\overline{x}'_n < x_n$,
- ...
- the last condition is that either $x_n < \underline{x}'_n$ or $\overline{x}'_n < x_n$.

Towards the algorithm. So, if $x \in X$ and $x_1 < \underline{x}'_1$, i.e., if $x_1 \in [\underline{x}_1, \underline{x}'_1]$, then the location x belongs to the difference

$$X - X'.$$

(Of course, if $\underline{x}'_1 \le \underline{x}_1$, this condition is never satisfied.)

All such locations x correspond to the box

$$[\underline{x}_1, \underline{x}'_1] \times [\underline{x}_2, \overline{x}_2] \times \ldots \times [\underline{x}_n, \overline{x}_n].$$

Thus, this box is a part of the set difference $X - X'$.

Similarly, if $x \in X$ and $x_1 > \overline{x}'_1$, i.e., if $x_1 \in [\overline{x}'_1, \overline{x}_1]$, then the location x belongs to the difference $X - X'$. (Of course, if $\overline{x}_1 \le \overline{x}'_1$, this condition is never satisfied.)

All such locations x correspond to the box

$$[\overline{x}'_1, \overline{x}_1] \times [\underline{x}_2, \overline{x}_2] \times \ldots \times [\underline{x}_n, \overline{x}_n].$$

Thus, this box is a part of the set difference $X - X'$.

In the following, if we want to have disjoint sub-boxes, it is sufficient to consider the remaining part of the set X, i.e., the part corresponding to

$$x_1 \in [\max(\underline{x}_1, \underline{x}'_1), \min(\overline{x}_2, \overline{x}'_1)].$$

After that, we can similarly consider possible values of x_2, etc.

Thus, we arrive at the following algorithm.

4 New Algorithm: Description, Advantage, and Example

What is given: boxes

$$X = [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_n, \overline{x}_n]$$

and

$$X' = [\underline{x}'_1, \overline{x}'_1] \times \ldots \times [\underline{x}'_n, \overline{x}'_n].$$

What we want. We want to represent the set difference $X - X'$ as a union of boxes.

Algorithm. For each i from 1 to n, let us compute the values

$$\ell_i \stackrel{\text{def}}{=} \max(\underline{x}_i, \underline{x}'_i)$$

and

$$u_i \stackrel{\text{def}}{=} \min(\overline{x}_i, \overline{x}'_i).$$

Then, we represent the difference $X - X'$ as the union of the following $2n$ boxes (some of which may be empty):

- n boxes

$$[\ell_1, u_1] \times \ldots \times [\ell_{i-1}, u_{i-1}] \times [\underline{x}_i, \underline{x}_i'] \times [\underline{x}_{i+1}, \overline{x}_{i+1} \times \ldots \times [\underline{x}_n, \overline{x}_n]$$

corresponding to $i = 1, \ldots, n$, and

- n boxes

$$[\ell_1, u_1] \times \ldots \times [\ell_{i-1}, u_{i-1}] \times [\overline{x}_i', \overline{x}_i] \times [\underline{x}_{i+1}, \overline{x}_{i+1} \times \ldots \times [\underline{x}_n, \overline{x}_n]$$

corresponding to $i = 1, \ldots, n$.

Advantage. This algorithm generates at most $2n$ boxes. In this sense, it is clearly more efficient than the above-mentioned traditional approach, which may lead to an unfeasible number of boxes.

Example. Let us trace our algorithm on the above example, when

$$X = [-2, 2] \times \ldots \times [-2, 2]$$

and

$$X' = [-1, 1] \times \ldots \times [-1, 1].$$

In this case, for each i, we have

$$[\ell_i, u_i] = [\max(-2, -1), \min(1, 2)] = [-1, 1].$$

Thus, our algorithm represents the difference $X - X'$ as the union of the following two sequence of boxes:

- the first sequence consists of the boxes

$$[-2, -1] \times [-2, 2] \times \ldots \times [-2, 2],$$

$$[-1, 1] \times [-2, -1] \times [-2, 2] \times \ldots \times [-2, 2],$$

$$[-1, 1] \times [-1, 1] \times [-2, -1] \times [-2, 2] \times \ldots \times [-2, 2],$$

$$\ldots$$

$$[-1, 1] \times \ldots \times [-1, 1] \times [-2, -1] \times [-2, 2],$$

$$[-1, 1] \times \ldots \times [-1, 1] \times [-2, -1];$$

- the second sequence consists of the boxes

$$[1, 2] \times [-2, 2] \times \ldots \times [-2, 2],$$

$$[-1, 1] \times [1, 2] \times [-2, 2] \times \ldots \times [-2, 2],$$

$$[-1, 1] \times [-1, 1] \times [1, 2] \times [-2, 2] \times \ldots \times [-2, 2],$$

$$\ldots$$

$$[-1, 1] \times \ldots \times [-1, 1] \times [1, 2] \times [-2, 2],$$

$$[-1, 1] \times \ldots \times [-1, 1] \times [1, 2].$$

Comment. It it worth mentioning that our algorithm is *not* invariant with respect to permutations. Depending on which coordinate we start with, we get, in general, different descriptions of the set difference as a union of boxes. This is a price that we pay for efficiency, since, as we have mentioned earlier:

- if we require invariance,
- then we end up with exponentially many boxes.

Acknowledgments

This work was supported in part by the US National Science Foundation grant HRD-1242122.

References

[1] Q. Brefort, L. Jaulin, M. Ceberio, and V. Kreinovich, "If we take into account that constraints are soft, then processing constraints becomes algorithmically solvable", *Proceedings of the IEEE Symposium on Computational Intelligence for Engineering Solutions CIES'2014*, Orlando, Florida, December 9–12, 2014, pp. 1–10.

[2] Q. Brefort, L. Jaulin, M. Ceberio, and V. Kreinovich, "Towards fast and reliable localization of an underwater object: an interval approach", *Journal of Uncertain Systems*, 2015, Vol. 9, No. 2, pp. 95–102.

[3] L. Jaulin and B. Descrochers, "Robust localisation using separators", *Proceedings of the Seventh International Workshop on Constraints Programming and Decision Making CoProd'2014*, Würzburg, Germany, September 21, 2014.

[4] L. Jaulin, M. Kiefer, O. Dicrit, and E. Walter, *Applied Interval Analysis*, Springer, London, 2001.

[5] L. Jaulin, E. Walter, O. Lévêque, and D. Meixzel, "Set inversion of χ-algorithms, with applications to guaranteed tobot location", *Mathematics and Computers in Simulation*, 2000, Vol. 52, No. 3–4, pp. 197–210.

[6] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational Complexity and Feasibility of Data Processing and Interval Computations*, Kluwer, Dordrecht, 1998.

[7] O. Lévêque, L. Jaulin, D. Deizel, and E. Walter, "Vehicle location from inaccurate telemetric data: a set inversion approach", *Proceesings of the 5th IFAC Symposium on Robot Control SY.RO.CO'97*, Nantes, France, 1997, Vol. 1, pp. 179–186.

[8] G. Mayer, *Interval Analysis and Automatic Result Verification*, de Gruyter, Berlin, 2017.

[9] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to Interval Analysis*, SIAM, Philadelphia, 2009.

[10] S. G. Rabinovich, *Measurement Errors and Uncertainty: Theory and Practice*, Springer Verlag, Berlin, 2005.

[11] J. Sliwka, L. Jaulin, M. Ceberio, and V. Kreinovich, "Processing interval sensor data in the presence of outliers, with potential applications to localizing underwater robots", *Proceedings of the 2011 IEEE International Conference on Systems, Man, and Cybernetics SMC'2011*, Anchorage, Alaska, October 9–12, 2011, pp. 2330–2337.

[12] A. Welte, L. Jaulin, M. Ceberio, and V. Kreinovich, "Robust data processing in the presence of uncertainty and outliers: case of localization problems", *Proceedings of the IEEE Series of Symposia in Computational Intelligence SSCI'2016*, Athens, Greece, December 6–9, 2016.

[13] A. Welte, L. Jaulin, M. Ceberio, and V. Kreinovich, "Avoiding fake boundaries in set interval computing", *Journal of Uncertain Systems*, 2017, Vol. 11, No. 2, pp. 137–148.

[14] A. Welte, L. Jaulin, M. Ceberio, and V. Kreinovich, "Computability of the avoidance set and of the set-valued identification problem", *Journal of Uncertain Systems*, 2017, Vol. 11, No. 2, pp. 129–136.