Journal of Uncertain Systems
Vol.4, No.4, pp.244-249, 2010
Online at: www.jus.org.uk

A Faster Algorithm for Computing the Sum of p-Boxes

Jaruchat Busaba[1], Sirima Suwan[1], Olga Kosheleva[2,*]

[1] *Department of Applied Statistics, Faculty of Applied Science,*
King Mongkut's University of Technology North Bangkok,
1518 Pibulsongkram Rd., Bangsue, Bangkok 10800 Thailand

[2] *Department of Teacher Education, University of Texas at El Paso,*
500 W. University, El Paso, TX 79968, USA

Received 9 January, 2009; Revised 30 January, 2010

Abstract

For many real-life quantities x, we do not have the complete information about their probability distribution. For such quantities, for each value x_0, instead of the *exact* value of the cumulative distribution function $F_x(x_0) = \text{Prob}(x \le x_0)$, we only know the *interval* $[\underline{F}_x(x_0), \overline{F}_x(x_0)]$ of possible values of $F_x(x_0)$. These intervals form a *probability box*, or *p-box*. Once we know p-boxes for x and y, and (which is fairly typical) we have no information about the correlation between x and y, what is the p-box for $x + y$? Algorithms for computing this p-box are known. In this paper, we describe a new faster algorithm for computing this p-box. This algorithm uses Fast Fourier Transform (FFT) to reduce computation time from $O(n^2)$ to $O(n \cdot \log(n))$, where n is the number of points x at which we know the intervals $[\underline{F}(x), \overline{F}(x)]$.

Keywords: imprecise probabilities, p-boxes, efficient algorithm

1 Cumulative Distribution Functions: A Natural Tool for Describing Probability Distributions

In many practical problems, we want to satisfy certain constraints, i.e., we want to make sure that a certain quantity x does not exceed a give threshold x_0 (or, vice versa, exceed a certain threshold x_0).

For example, when we design a car,

- we want to make sure that the level of pollution x generated by the car does not exceed the allowed level x_0, and
- we also want to make sure that the power x of the car's engine is at least as large as what is needed to move this car with a required speed.

Usually, in practice, the value of the corresponding quantity x depends on many factors. For example, the power of the car's engine depends on the quality of the gasoline which may slightly differ from one gas station to the other. In such situations, at best, we can determine the *probabilities* of different values of x.

Since the quantity x is now random, it is impossible to 100% guarantee that the value of x is always below or always under the given threshold. The only thing that we can, in principle, guarantee, is that the probability of not satisfying a given constraint is sufficiently low. Thus, to gauge the quality of a given design, we must find, for each value x_0, the probability $F(x_0) = \text{Prob}(x \le x_0)$ that the quantity x does satisfy the constraint – or, equivalently, the probability $\text{Prob}(x > x_0) = 1 - F(x_0)$ that x does not satisfy the constraint.

The values of $F(x_0)$ for different x_0 form a *cumulative distribution function* (cdf, for short). Thus, cdf is indeed a natural tool for describing probability distributions.

2 p-Boxes: A Natural Tool for Describing Partial Information about Probabilities

For many real-life quantities x, we do not have the complete information about their probability distributions.

*Corresponding author. Email: olgak@utep.edu (O. Kosheleva).

For such quantities, for each value x_0, instead of the *exact* value of the cumulative distribution function $F(x_0) = \text{Prob}(x \le x_0)$, we only know the *interval* $[\underline{F}(x_0), \overline{F}(x_0)]$ of possible values of $F(x_0)$. The intervals corresponding to different x_0 form a *probability box*, or *p-box*; see, e.g., [3, 4].

3 Processing p-Boxes

Some quantities of interest z are described in terms of others: e.g., $z = f(x, y)$. For example, for the car, the pollution level is determined by several characteristics of the engine and of the filters in the car's exhaust. In such situations, it is reasonable to use the information about the quantities x and y to determine information about $z = f(x, y)$.

In particular, since, as we have mentioned, cdf is a very important piece of information about a physical quantity, we must be able to find the cdf for z based on the cdfs for x and y. Taking into account that we usually have only partial information about the probabilities, we can reformulate this question is a more practically useful terms:

- given p-boxes for x and y,
- find the resulting p-box for $z = f(x, y)$.

4 The Result Depends on Whether the Variables are Independent or Not

The p-box formed by all possible distributions for $z = f(x, y)$ depends on what we know about the correlation between x and y:

- In some practical situations, we know that x and y are statistically independent.
- In other practical situations, we have no information about the correlation between x and y:
  - it is possible that x and y are independent;
  - it is possible that x and y are positively correlated; and
  - it is also possible that x and y are negatively correlated.

It is also possible that we have partial information about the correlation between x and y.

5 Addition: An Important Practical Case of Data Processing

In many practical situations, possible changes in x and y are small: there are some nominal values $\widetilde{x}$ and $\widetilde{y}$ and the actual values of the quantities x and y are close to these nominal values. In other words, the differences $\Delta x \stackrel{\text{def}}{=} x - \widetilde{x}$ and $\Delta y \stackrel{\text{def}}{=} y - \widetilde{y}$ are small and therefore, we can expand the expression $f(x, y) = f(\widetilde{x} + \Delta x, \widetilde{y} + \Delta y)$ in Taylor series in Δx and Δy and keep only linear terms in this expansion:

$$z = f(\widetilde{x} + \Delta x, \widetilde{y} + \Delta y) \approx c_0 + c_x \cdot \Delta x + c_y \cdot \Delta y, \tag{1}$$

where

$$c_0 \stackrel{\text{def}}{=} f(\widetilde{x}, \widetilde{y}), \quad c_x \stackrel{\text{def}}{=} \frac{\partial f}{\partial x}, \quad c_y \stackrel{\text{def}}{=} \frac{\partial f}{\partial y}. \tag{2}$$

It is reasonably easy to transform the cdf (or a p-box) for x into a cdf for $\Delta x = x - \widetilde{x}$, and it is easy to transform this cdf (or p-box) for Δx into a cdf for $x' = c_x \cdot \Delta x$. Similarly, from the known p-box for Δy, we can compute the p-boxes for $\Delta y = y - \widetilde{y}$ and $y' = c_y \cdot \Delta y$.

Thus, the main non-trivial step of computing the p-box for z is computing the p-box for the sum $x' + y'$ of the two quantities $x' = c_x \cdot \Delta x$ and $y' = c_y \cdot \Delta y$ for each of which we know the p-boxes. Thus, addition is, indeed, a very important practical case of data processing. Addition is what we will study in this paper.

Comment. There is an additional reason why addition is important. Often, the actual relation $z = f(x, y)$ has the form of a product $z = x \cdot y$. This case can also be reduced to addition $Z = X + Y$ if instead of the original quantities x, y, and z, we consider their logarithms

$$Z \stackrel{\text{def}}{=} \ln(z); \quad X \stackrel{\text{def}}{=} \ln(x); \quad Y \stackrel{\text{def}}{=} \ln(y). \tag{3}$$

6 Formulation of the Resulting Computational Problem

In view of the above, we will consider the following problem:

- we know the p-boxes $[\underline{F}_x(x), \overline{F}_x(x)]$ and $[\underline{F}_y(y), \overline{F}_y(y)]$ for x and y;
- we want to find the p-box $[\underline{F}_z(z), \overline{F}_z(z)]$ corresponding to $z = x + y$.

We consider two version of this problem:

- when we know that x and y are independent, and
- when we have no information about the dependence between x and y.

7 The Solution to the Problem is Known

For each of the two versions, the solution to the above problem is known; see, e.g., [3, 4, 6].

For the independent case, the answer can be best described in terms of the probability density functions $\underline{\rho}_x(x)$, $\overline{\rho}_x(x)$, $\underline{\rho}_y(y)$, $\overline{\rho}_y(y)$, $\underline{\rho}_z(z)$, and $\overline{\rho}_z(z)$ that correspond to the cdfs $\underline{F}_x(x)$, $\overline{F}_x(x)$, $\underline{F}_y(y)$, $\overline{F}_y(y)$, $\underline{F}_z(z)$, and $\overline{F}_z(z)$:

$$\underline{\rho}_z(z) = \int \underline{\rho}_x(x) \cdot \underline{\rho}_y(y - x)\, dx; \tag{4}$$

$$\overline{\rho}_z(z) = \int \overline{\rho}_x(x) \cdot \overline{\rho}_y(y - x)\, dx. \tag{5}$$

In computational terms, we only know a finite number of values of the pdfs, with some step δx, so we have to use the discretized versions of these formulas:

$$\underline{\rho}_z(z) = \sum_x \underline{\rho}_x(x) \cdot \underline{\rho}_y(y - x) \cdot \delta x; \tag{6}$$

$$\overline{\rho}_z(z) = \sum_x \overline{\rho}_x(x) \cdot \overline{\rho}_y(y - x) \cdot \delta x. \tag{7}$$

For the possibly dependent case, the answer can be best described in terms of the *quantiles*. For a given integer n, the i-th quantile r_i is the value for which $F(r_i) = i/n$. When instead of the exact cdf $F(x)$, we have a p-box $[\underline{F}(x), \overline{F}(x)]$, different values $F(x) \in [\underline{F}(x), \overline{F}(x)]$ lead, in general, to different values of the quantiles. The smallest possible value $\underline{r}_i$ of the i-th quantile is attained for the largest possible value $\overline{F}(x)$ of the cdf, and, vices versa, the largest possible value $\overline{r}_i$ of the i-th quantile is attained for the smallest possible value $\underline{F}(x)$ of the cdf. Thus, a p-box $[\underline{F}(x), \overline{F}(x)]$ can be described by $n + 1$ quantile intervals $[\underline{r}_i, \overline{r}_i]$, where $\overline{F}(\underline{r}_i) = \underline{F}(\overline{r}_i) = i/n$.

Once we know the quantile intervals $[\underline{x}_i, \overline{x}_i]$ and $[\underline{y}_i, \overline{y}_i]$ corresponding to x and y, the quantile intervals $[\underline{z}_i, \overline{z}_i]$ for $x + y$ can be obtained by using the following formulas [3, 4, 6]:

$$\underline{z}_k = \max_i(\underline{x}_i + \underline{y}_{k-i}); \tag{8}$$

$$\overline{z}_k = \min_i(\overline{x}_i + \overline{y}_{k-n+i}). \tag{9}$$

It should be mentioned that once we know the formula for $\underline{z}_k$ (i.e., for $\overline{F}(x)$), the formula for $\overline{z}_k$ (i.e., for $\underline{F}(x)$) can be obtained from the fact that

$$F_{-x}(x_0) = \text{Prob}(-x \le x_0) = \text{Prob}(x \ge -x_0) = 1 - \text{Prob}(x \le -x_0) = 1 - F_x(-x_0), \tag{10}$$

and therefore,

$$\underline{F}_{-x} = 1 - \overline{F}_x(-x_0). \tag{11}$$

In view of this reduction, it is sufficient to be able to efficiently compute the values $\underline{z}_k$.

8 Computational Complexity of the Above Formulas – When They are Applied Directly

Let us first consider the case when we simply use the formulas (6), (7), (8), and (9) to directly compute the p-box for $z = x + y$. Let us denote by n the number of known values of x- and y-characteristics.

In this case, each of these formulas requires that for each of n values of the corresponding z-characteristic, we need to process n values of x- and y-characteristics. For example, to compute each value $\underline{z}_k$, we need to process n values $\underline{x}_i$ and $\underline{y}_{k-i}$. Thus, overall, we need $O(n^2)$ computational steps.

9 Known Fact: In the Independent Case, Computations Can be Made Faster

It is known that for independent variables x and y, we can perform computations of the expressions (6) and (7) faster.

Indeed, from the mathematical viewpoint, the expressions (6) and (7) are *convolutions*. It is known (see, e.g., [1, 2]) that convolutions can be computed faster than in the time $O(n^2)$ is we use Fourier transforms

$$\hat{f}(\omega) \stackrel{\text{def}}{=} \frac{1}{\sqrt{2\pi}} \cdot \int e^{-\mathrm{i}\cdot\omega\cdot x} \cdot f(x)\, dx \tag{12}$$

of the corresponding functions. Specifically:

- the Fourier transform $\hat{h}(\omega)$ of the convolution

$$h(z) = \int f(x) \cdot g(z-x)\, dx \tag{13}$$

  of two functions $f(x)$ and $g(x)$ is equal to the product of their Fourier transforms: $\hat{h}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)$, and

- both the Fourier transform itself and the inverse Fourier transform (that transforms the Fourier transform $\hat{f}(\omega)$ back into the original functions $f(x)$) can be computed in time $O(n\cdot\log(n))$; the corresponding algorithms are called *Fast Fourier Transform* (FFT, for short).

For the probability density function $f(x) = \rho(x)$, the Fourier transform (12) has a direct statistical meaning: it is the expected value $E[e^{-\mathrm{i}\cdot\omega\cdot x}]$ of the quantity $e^{-\mathrm{i}\cdot\omega\cdot x}$. This expected value $\chi_x(\omega)$ is known as a *characteristic function* of the probability distribution, and it is known that the characteristic function of the sum $z = x+y$ of two independent random variables is equal to the product of the corresponding characteristic functions: $\chi_z(\omega) = \chi_x(\omega) \cdot \chi_y(\omega)$.

Thus, the pdf $\rho_z(z)$ can be computed as follows:

- first, we apply FFT to the pdfs $\rho_x(x)$ and $\rho_y(y)$ and compute the corresponding characteristic functions $\chi_x(\omega)$ and $\chi_y(\omega)$;
- then, we compute the characteristic function $\chi_z(\omega)$ as

$$\chi_z(\omega) = \chi_x(\omega) \cdot \chi_y(\omega); \tag{14}$$

- finally, we apply the inverse FFT to the function $\chi_z(\omega)$ and get the desired pdf $\rho_z(z)$.

Since both FFT and inverse FFT require time $O(n \cdot \log(n))$, and the point-wise multiplication $\chi_z(\omega) = \chi_x(\omega) \cdot \chi_y(\omega)$ requires as many computational steps as there are values, i.e., n, overall, we need

$$O(n \cdot \log(n)) + O(n) + O(n \cdot \log(n)) = O(n \cdot \log(n)) \tag{15}$$

computational steps – which, for large n, is much smaller than n^2 steps for the direct application of the convolution formula.

10 New Idea: For the Possibly Dependent Case, Computations Can Also be Performed Faster

To compute the formula (8) faster, let us reduce it to computing the convolution; this reduction is similar to the one described in [5] for a different computational problem – related to processing fuzzy data.

The formula (8) is similar to the convolution formula, but it has two differences:

- first, in the formula (8), we add the values $\underline{x}_i$ and $\underline{y}_{k-i}$, while in the convolution formula, we multiply the corresponding values;
- second, in the formula (8), we take the maximum of the values corresponding to different i, while in the convolution formula, we add these values.

The first difference is the easiest to handle: to reduce addition to multiplication, we can use the known fact that $e^{a+b} = e^a \cdot e^b$. Thus, if we take

$$x_i \stackrel{\text{def}}{=} e^{\underline{x}_i}, \quad y_i \stackrel{\text{def}}{=} e^{\underline{y}_i}, \quad z_i \stackrel{\text{def}}{=} e^{\underline{z}_i}, \tag{16}$$

then the formula (8) takes the multiplicative form

$$z_k = \max(x_i \cdot y_{k-i}). \tag{17}$$

In this representation, the values $x_i = e^{\underline{x}_i}$ and $y_i = e^{\underline{y}_i}$ are always positive.

To reduce maximum to the sum, we can take into account that for positive values a_i, we have

$$\max(a_1, \ldots, a_n) = \lim_{p \to \infty} (a_1^p + \ldots + a_n^p)^{1/p} \tag{18}$$

and thus, for sufficiently large p, we have

$$\max(a_1, \ldots, a_n) \approx (a_1^p + \ldots + a_n^p)^{1/p}. \tag{19}$$

Applying this approximate equality to the right-hand side of the formula (17), we conclude that

$$z_k \approx \left(\sum_i x_i^p \cdot y_{k-i}^p \right)^{1/p}, \tag{20}$$

i.e., equivalently, that

$$z_k^p \approx \sum_i x_i^p \cdot y_{k-i}^p. \tag{21}$$

So, if we take

$$X_i \stackrel{\text{def}}{=} x_i^p = e^{p \cdot \underline{x}_i}, \quad Y_i \stackrel{\text{def}}{=} y_i^p = e^{p \cdot \underline{y}_i}, \quad Z_i \stackrel{\text{def}}{=} z_i = e^{p \cdot \underline{z}_i}, \tag{22}$$

we conclude that

$$Z_k \approx \sum_i X_i \cdot Y_{k-i}, \tag{23}$$

i.e., that the values Z_k are equal to the convolution of the values X_i and Y_j.

We already know how to compute the convolution fast. Once we know $Z_k = e^{p \cdot \underline{z}_k}$, we can reconstruct $\underline{z}_k$ as $\underline{z}_k = \dfrac{1}{p} \cdot \ln(Z_k)$. Thus, we arrive at the following fast algorithm for computing the sum of the corresponding p-boxes for the possibly dependent case.

11 Resulting Fast Algorithm for Computing the Sum $Z = X + Y$ of p-Boxes

Without losing generality, we will only describe how to compute the lower endpoints $\underline{z}_k$ of the quantile intervals from the corresponding endpoints $\underline{x}_i$ and $\underline{y}_i$. For these computations, we need to select a large integer p. Then:

- first, we compute the values
$$X_i = x_i^p = e^{p \cdot \underline{x}_i}, \quad Y_i \stackrel{\text{def}}{=} y_i^p = e^{p \cdot \underline{y}_i}; \tag{24}$$
- then, we apply FFT to the values X_i and to the values Y_j, resulting in $\hat{X}(\omega)$ and $\hat{Y}(\omega)$;
- we compute $\hat{Z}(\omega) = \hat{X}(\omega) \cdot \hat{Y}(\omega)$;
- we apply the inverse FFT to the function $\hat{Z}(\omega)$ and get the Z_i;
- finally, we compute $\underline{z}_k = \dfrac{1}{p} \cdot \ln(Z_k)$.

Here, both FFT and inverse FFT require time $O(n \cdot \log(n))$, and point-wise operations like multiplication or raising to the power p require as many computational steps as there are values (i.e., n). Thus, overall, we need
$$O(n \cdot \log(n)) + O(n) + O(n \cdot \log(n)) = O(n \cdot \log(n)) \tag{25}$$
computational steps – which, for large n, is much smaller than n^2 steps for the direct application of the formula (8).

Acknowledgments

The authors are thankful to Sa-aat Niwitpong and Vladik Kreinovich for their help.

References

[1] Bracewell, R., *The Fourier Transform and Its Applications*, McGraw-Hill, 1986.

[2] Cormen, T.H., C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, Massachisetts, 2001.

[3] Ferson, S., *RAMAS Risk Calc 4.0*. CRC Press, Boca Raton, Florida, 2002.

[4] Ferson, S., V. Kreinovich, and W.T. Tucker, Untangling equations involving uncertainty: deconvolutions, updates, and backcalculations, *Proceedings of the NSF Workshop on Reliable Engineering Computing*, Savannah, Georgia, September 15–17, 2004.

[5] Kosheleva, O., S.D. Cabrera, G.A. Gibson, and M. Koshelev, Fast implementations of fuzzy arithmetic operations using fast fourier transform (FFT), *Fuzzy Sets and Systems*, vol.91, no.2, pp.269–277, 1997.

[6] Williamson, R.C., and T. Downs, Probabilistic arithmetic I: Numerical methods for calculating convolutions and dependency bounds, *International Journal of Approximate Reasoning*, vol.4, pp.89–158, 1990.