CUBO A Mathematical Journal
Vol.13, N$^{\underline{o}}$01, (103–123). March 2011

Engineering Design under Imprecise Probabilities: Computational Complexity

Vladik Kreinovich
Department of Computer Science,
University of Texas at El Paso,
500 W. University,
El Paso, Texas 79968, USA.
email: `vladik@utep.edu`

ABSTRACT

In engineering design problems, we want to make sure that a certain quantity c of the designed system lies within given bounds – or at least that the probability of this quantity to be outside these bounds does not exceed a given threshold. We may have several such requirements – thus the requirement can be formulated as bounds $[\underline{F}_c(x), \overline{F}_c(x)]$ on the cumulative distribution function $F_c(x)$ of the quantity c; such bounds are known as a *p-box.*

The value of the desired quantity c depends on the design parameters a and the parameters b characterizing the environment: $c = f(a, b)$. To achieve the design goal, we need to find the design parameters a for which the distribution $F_c(x)$ for $c = f(a, b)$ is within the given bounds for all possible values of the environmental variables b. The problem of computing such a is called *backcalculation.* For b, we also have ranges with different probabilities – i.e., also a p-box. Thus, we have backcalculation problem for p-boxes.

For p-boxes, there exist efficient algorithms for finding a design a that satisfies the given constraints. The next natural question is to find a design that satisfies additional constraints: on the cost, on the efficiency, etc. In this paper, we prove that that in general, the problem of finding such a design is computationally difficult (NP-hard). We show that this problem is NP-hard already in the simplest possible linearized case, when the dependence $c = f(a, b)$ is linear. We also provide an example when an efficient algorithm is possible.

RESUMEN

En los problemas de diseño de ingeniería, en los que quiere asegurarse de que una determinada cantidad c del sistema se encuentra dentro de los límites dados – o por lo menos que la probabilidad de esa cantidad fuera de estos límites no superen un determinado umbral. Es posible que haya varios requisitos – la exigencia puede formularse como límites $[\underline{F}_c(x), \overline{F}_c(x)]$ en la función de distribución acumulada $F_c(x)$ de la cantidad de c, esos límites son conocidos como *p-caja*.

El valor de la cantidad deseada c depende de los parámetros de diseño a y los parámetros b caracterizar el medio ambiente: $c = f(a, b)$. Para lograr el objetivo de diseño, tenemos que encontrar los parámetros de diseño a para que la distribución de $F_c(x)$ para $c = f(a, b)$ esté dentro de los límites dados por todos los valores posibles de las variables ambientales b. El problema de la informática se llama *a retrocálculo*. Por b, también tienen rangos con diferentes probabilidades, es decir, también una p-box. Por lo tanto, tenemos un problema de retrocálculo para p-cajas.

Para p-cajas, existen algoritmos eficientes para encontrar un diseño a que satisface las restricciones dadas. La pregunta lógica es encontrar un diseño que satisfaga restricciones adicionales: el coste, la eficiencia, etc. En este trabajo, demostramos, en general, el problema de encontrar un diseño que es computacionalmente difícil (NP-hard). Se demuestra que este problema es NP-hard ya en el caso lineal más simple posible, cuando la dependencia $c = f(a, b)$ es lineal. También ofrecemos un ejemplo, cuando un algoritmo eficiente es posible.

Keywords: Engineering design, imprecise probability, computational complexity, p-boxes, NP-hard

AMS Subject Classification: 65K10, 65J99, 49M15, 49J53, 47J20, 47H04, 90C30, 90C33.

1 Engineering Design Problems and the Notion of Backcalculation: Deterministic Case

One of the main objective of engineering design is to guarantee that the value of a certain quantity (or several quantities) c is within a given range $[\underline{c}, \overline{c}]$. For example, when we design a car engine, we must make sure:

- that its power is at least as much as needed for the loaded car to climb the steepest mountain roads,
- that the concentration of undesirable substances in the exhaust does not exceed the required threshold, etc.

The value of the quantity c usually depends on the parameters a describing the design and on the parameters b of the environment: $c = f(a, b)$. For example, the concentration of a substance in a car exhaust depends:

- on the parameter(s) that describe the design of the car exhaust filters, and
- on the concentration of the chemicals in the original fuel.

We need to select a design a in such a way that

$$c = f(a, b) \in [\underline{c}, \overline{c}]$$

for all possible values of the environmental parameter(s) b.

In this paper, we consider the simplest case when:

- the design of each system is characterized by a single parameter a, and
- the environment is also characterized by a single parameter b.

We will show that already in this simple case, the design problem is, in general, computationally difficult (NP-hard). We also show that in some cases, a feasible algorithm is possible. (Some of our results first appeared in a conference paper [5].)

To be able to find a design that satisfies the given constraint on c for all possible values of the environmental parameter b, we need to know which values of b are possible, i.e., we need to know the range $[\underline{b}, \overline{b}]$ of possible values of b. Thus, we arrive at the following problem:

- we know the desired range $[\underline{c}, \overline{c}]$;
- we know the dependence $c = f(a, b)$;
- we know the range $[\underline{b}, \overline{b}]$ of possible values of b;
- we want to describe the set of all values of a for which $f(a, b) \in [\underline{c}, \overline{c}]$ for all $b \in [\underline{b}, \overline{b}]$.

This design-related problem is sometimes called a *backcalculation* problem, to emphasize its difference from the *forward calculation* problem, when

- we are given a design a and
- we want to estimate the value of the desired characteristic $c = f(a, b)$.

2 Linearized Problem

In many engineering situations, the intervals of possible values of a and b are reasonably narrow: $a \approx \widetilde{a}$ and $b \approx \widetilde{b}$ for some $\widetilde{a}$ and $\widetilde{b}$. In such situations, we can expand the dependence $c = f(a, b)$ in Taylor series and ignore terms which are quadratic and higher order in terms of $\Delta a \stackrel{\text{def}}{=} a - \widetilde{a}$ and $\Delta b \stackrel{\text{def}}{=} b - \widetilde{b}$. As a result, we get a simple linear dependence

$$c \approx c_0 + k_a \cdot a + k_b \cdot b. \tag{2.1}$$

We can simplify this expression even further if we take into account that the numerical value of each of the quantities a and b depends on the choice of the starting point and on the choice of a measuring unit. If we change the starting point and the measuring unit, then the new numerical value can be obtained from the original one by an appropriate linear transformation. For example, if we know the temperature t_C in Celsius, then we can compute the temperature t_F in the Fahrenheit scale as $t_F = 32 + 1.8 \cdot t_C$. We can use this possibility to simplify the above expression for c. Specifically, we can change the starting points and the measuring units in such a way that:

- the new numerical value for a is described by the linear expression $c_0 + k_a \cdot a$, and
- the new numerical value for b is described by the linear expression $k_b \cdot b$.

In these new scales, the dependence of c on a and b takes the simplest form $c = a + b$.

We will show that the design problem becomes computationally difficult (NP-hard) already for this simplest case.

3 From Guaranteed Bounds to p-Boxes

Ideally, it is desirable to provide a 100% guarantee that the quantity c never exceeds the threshold $\overline{c}$. In practice, however, too many unpredictable factors affect the performance of a system and thus, such a guarantee is not realistically possible. What we can realistically guarantee is that the probability of exceeding c is small enough. In other words, we set some threshold $\varepsilon_c > 0$ and we require that $\text{Prob}(c \le \overline{c}) \ge 1 - \varepsilon_c$.

In addition to this requirement, we can also require that the excess of c over $\overline{c}$ be not too large. This can be done, e.g., by requiring that for some value $c_1 > \overline{c}$, the probability $\text{Prob}(c \le c_1)$ is bounded from below by the value $1 - \varepsilon_1$ for some smaller $\varepsilon_1 < \varepsilon$. We can several such requirements for different values c_i and ε_i.

Similarly, instead of the idealized exact inequality $c \ge \underline{c}$, in practice, we can only require that $\text{Prob}(c \ge \underline{c}) \le \delta$ for some small probability $\delta > 0$.

From the mathematical viewpoint, all such constraints are lower or upper bounds on the values of the cumulative distribution function

$$F_c(x) \stackrel{\text{def}}{=} \text{Prob}(c \le x).$$

By combining the bounds corresponding to all the constraints, we can thus conclude that the cdf $F_c(x)$ must satisfy, for every x, the inequalities

$$\underline{F}_c(x) \le F_c(x) \le \overline{F}_c(x), \tag{3.1}$$

where $\underline{F}_c(x)$ is the largest of all the lower bounds on $F_c(x)$ and $\overline{F}_c(x)$ is the smallest of all the upper bounds on $F_c(x)$.

In other words, for every x, the corresponding value $F_c(x)$ must belong to the interval $[\underline{F}_c(x), \overline{F}_c(x)]$. This x-dependent interval is known as a *probability box*, or a *p-box*, for short; see, e.g., [2].

Similarly, for the environmental parameter b, we rarely know guaranteed bounds $\underline{b}$ and $\overline{b}$. At best, we know that for a given bound $\overline{b}$, the probability of exceeding this bound is small, i.e., that $\text{Prob}(b \le \overline{b}) \ge 1 - \varepsilon_b$ for some small ε_b. So here too, instead of a single bound, in effect, we have a p-box $[\underline{F}_b(x), \overline{F}_b(x)]$.

4 Towards Formulating the Design (Backcalculation) Problem for p-Boxes

In the deterministic approach to design, we assume that we can manufacture an object with the exact value a of the corresponding parameter – or at least the value which is guaranteed to be within the given bounds $[\underline{a}, \overline{a}]$.

In manufacturing, however, it is not practically possible to always guarantee that the value a is within the given interval. At best, we can guarantee that, e.g., the probability of $a \le \overline{a}$ is greater than or equal to $1 - \varepsilon_a$ for some small value ε_a. In other words, the design restriction on a can also be formulated in terms of p-boxes. Thus, we arrive at the following problem.

5 Backcalculation Problem for p-Boxes

We are given:

- the desired p-box $[\underline{F}_c(x), \overline{F}_c(x)]$ for c;
- the dependence $c = f(a, b)$; and
- the p-box $[\underline{F}_b(x), \overline{F}_b(x)]$ describing b.

Our objective is to find a p-box $[\underline{F}_a(x), \overline{F}_a(x)]$ for which:

- for every probability distribution
$$F_a(x) \in [\underline{F}_a(x), \overline{F}_a(x)],$$
- for every probability distribution $F_b(x) \in [\underline{F}_b(x), \overline{F}_b(x)]$, and
- for all possible correlations between a and b,

the distribution of $c = f(a, b)$ is within the given p-box $[\underline{F}_c(x), \overline{F}_c(x)]$.

6 Reminder: Forward Calculation for p-Boxes

In order to analyze the *back*calculation problem for p-boxes, let us first describe how the corresponding *forward* calculation problem is solved. Let us assume that we know:

- the p-box $[\underline{F}_a(x), \overline{F}_a(x)]$ for a; and
- the p-box $[\underline{F}_b(x), \overline{F}_b(x)]$ describing b.

The objective of forward calculation is to find the range $[\underline{F}_c(x), \overline{F}_c(x)]$ of possible values of $F_c(x)$ for $c = f(a, b)$ for all possible distributions $F_a(x) \in [\underline{F}_a(x), \overline{F}_a(x)]$ and $F_b(x) \in [\underline{F}_b(x), \overline{F}_b(x)]$ and all possible correlations between a and b.

It turns out that these calculations are best done in terms not of the original cdfs and p-boxes, but rather in terms of their inverses – quantile functions. For a cdf $F_a(x)$, quantiles $a_0, \ldots, a_n$ are described as values for which $F_a(a_i) = \dfrac{i}{n}$. Since the cdf is monotonic, the quantiles are also monotonic: if $i < j$, then $a_i \le a_j$.

When instead of the exact cdf, we only know a p-box $[\underline{F}_a(x), \overline{F}_a(x)]$, then instead of the exact quantiles a_i we only know interval bounds $[\underline{a}_i, \overline{a}_i]$ for these quantiles:

- the lower bounds $\underline{a}_i$ are quantiles of the function $\overline{F}_c(x)$, and
- the upper bounds $\overline{a}_i$ are quantiles of the function $\underline{F}_c(x)$.

These bounds are also monotonic: if $i < j$, then $\underline{a}_i \le \underline{a}_j$ and $\overline{a}_i \le \overline{a}_j$.

For the function $c = f(a, b) = a + b$, forward calculation can be easily described in terms of the quantile bounds: once we know the bounds $[\underline{a}_i, \overline{a}_i]$ and $[\underline{b}_i, \overline{b}_i]$ corresponding to a and b, we can compute the quantile bounds for c as follows:

$$\underline{c}_i = \max_j(\underline{a}_j + \underline{b}_{i-j}); \tag{6.1}$$

$$\overline{c}_i = \min_j(\overline{a}_{j-i} + \overline{b}_{n-j}). \tag{6.2}$$

These formulas were first described in [8].

7 Formulation of the Problem

In terms of quantile bounds, the backcalculation problem takes the following form:

- we know the quantile intervals $[\underline{b}_i, \overline{b}_i]$ corresponding to the environmental variable b;
- we are given the intervals $[\widetilde{\underline{c}}_i, \widetilde{\overline{c}}_i]$ that should contain the quantiles for $c = a + b$;
- we must find the bounds $\underline{a}_i$ and $\overline{a}_i$ for which, for the values $\underline{c}_i$ and $\overline{c}_i$ are determined by the formulas (6.1) and (6.2), we have $[\underline{c}_i, \overline{c}_i] \subseteq [\widetilde{\underline{c}}_i, \widetilde{\overline{c}}_i]$.

8 In Effect, We Have Two Different Problems: Finding $\underline{a}_i$ and Finding $\overline{a}_i$

An important consequence of the formulas (6.1) and (6.2) is that:

- the lower bounds $\underline{c}_i$ for c are determined only by the lower bounds $\underline{a}_i$ and $\underline{b}_i$ for a and b, and
- the upper bounds $\overline{c}_i$ for c are determined only by the upper bounds $\overline{a}_i$ and $\overline{b}_i$ for a and b.

Thus, in effect, we can formulate the problem of finding the values $\underline{a}_i$ and the problem of finding the values $\overline{a}_i$ as two separate yet similar problems.

Without losing generality, in the following text, we will only consider the following problem of finding $\underline{a}_i$:

- we know the values $\underline{b}_i$;
- we are given the values $\widetilde{\underline{c}}_i$;
- we must find the values $\underline{a}_0 \le \ldots \le \underline{a}_n$ for which

$$\widetilde{\underline{c}}_i \le \max_j (\underline{a}_j + \underline{b}_{i-j}). \tag{8.1}$$

9 A Simple Example

Before we start describing how to solve the general problem, let us first describe a simple example.

In the general p-box case, to describe the uncertainty of a variable x, we use $n + 1$ quantile intervals $[\underline{x}_i, \overline{x}_i]$ for $i = 0, 1, \ldots, n$.

P-box uncertainty is a generalization of the case of interval uncertainty, in which the uncertainty in each variable x is characterized by a single interval $[\underline{x}, \overline{x}]$. This case corresponds to $n = 0$.

In this case, the inequality for $\underline{a}_0$ takes the form $\widetilde{\underline{c}}_0 \le \underline{b}_0 + \underline{a}_0$ or, equivalently, $\widetilde{\underline{c}}_0 - \underline{b}_0 \le \underline{a}_0$ – the same form as for interval uncertainty.

We are looking for the simplest possible example which is different from interval uncertainty. After a single interval, the next simplest example is when we use two intervals, i.e., when $n = 1$.

In this case, we need to find two values $\underline{a}_0 \le \underline{a}_1$ that satisfy the following inequalities:

$$\widetilde{\underline{c}}_0 \le \underline{b}_0 + \underline{a}_0; \tag{9.1}$$

$$\widetilde{\underline{c}}_1 \le \max(\underline{b}_0 + \underline{a}_1, \underline{b}_1 + \underline{a}_0). \tag{9.2}$$

Depending on which term in the max expression is the largest, we have two cases: $\underline{b}_0 + \underline{a}_1 \le \underline{b}_1 + \underline{a}_0$ and $\underline{b}_0 + \underline{a}_1 \ge \underline{b}_1 + \underline{a}_0$.

In the first case, the following inequalities must be satisfied:

$$\begin{gathered} \widetilde{\underline{c}}_0 \le \underline{b}_0 + \underline{a}_0; \\ \underline{b}_0 + \underline{a}_1 \le \underline{b}_1 + \underline{a}_0; \\ \widetilde{\underline{c}}_1 \le \underline{b}_1 + \underline{a}_0. \end{gathered} \tag{9.3}$$

The first of these three inequalities from (9.3) is equivalent to

$$\widetilde{\underline{c}}_0 - \underline{b}_0 \le \underline{a}_0. \tag{9.4}$$

Similarly, the third inequality from (9.3) is equivalent to

$$\widetilde{\underline{c}}_1 - \underline{b}_1 \le \underline{a}_0. \tag{9.5}$$

Thus, these two inequalities are equivalent to

$$\max(\widetilde{\underline{c}}_0 - \underline{b}_0, \widetilde{\underline{c}}_1 - \underline{b}_1) \le \underline{a}_0. \tag{9.6}$$

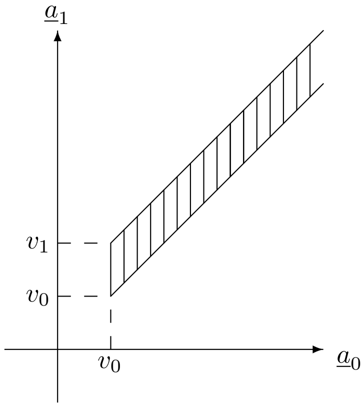
The second inequality from (9.3) is equivalent to

$$\underline{a}_1 - \underline{a}_0 \le \underline{b}_1 - \underline{b}_0. \tag{9.7}$$

Thus, in the first case, the values $\underline{a}_0$ and $\underline{a}_1$ must satisfy the following two inequalities:

$$\begin{gathered} \underline{a}_0 \ge \max(\widetilde{\underline{c}}_0 - \underline{b}_0, \widetilde{\underline{c}}_1 - \underline{b}_1); \\ 0 \le \underline{a}_1 - \underline{a}_0 \le \underline{b}_1 - \underline{b}_0. \end{gathered} \tag{9.8}$$

Graphically, the values that satisfy these inequalities fill the following area:



where $v_0 \stackrel{\text{def}}{=} \max(\widetilde{\underline{c}}_0 - \underline{b}_0, \widetilde{\underline{c}}_1 - \underline{b}_1)$ and $v_1 \stackrel{\text{def}}{=} v_0 + \underline{b}_1 - \underline{b}_0$.

In the second case, the following inequalities must be satisfied:

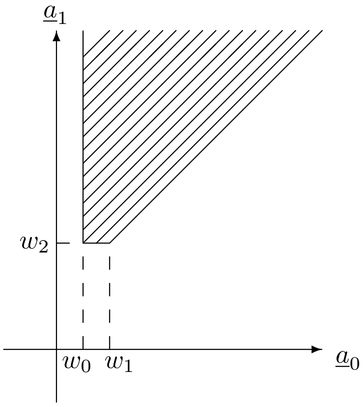
$$\begin{gathered}\widetilde{\underline{c}}_0 \le \underline{b}_0 + \underline{a}_0;\\ \\ \underline{b}_0 + \underline{a}_1 \ge \underline{b}_1 + \underline{a}_0;\\ \\ \widetilde{\underline{c}}_1 \le \underline{b}_0 + \underline{a}_1.\end{gathered} \tag{9.9}$$

By moving the unknowns to one side and all the other terms to the other side, we conclude that we must satisfy the following inequalities:

$$\begin{gathered}\underline{a}_0 \ge \widetilde{\underline{c}}_0 - \underline{b}_0;\\ \\ \underline{a}_1 - \underline{a}_0 \ge \underline{b}_1 - \underline{b}_0;\\ \\ \underline{a}_1 \ge \widetilde{\underline{c}}_1 - \widetilde{\underline{c}}_0.\end{gathered} \tag{9.10}$$

(Since $\underline{b}_1 - \underline{b}_0 \ge 0$, the second inequality automatically implies that $\underline{a}_1 \ge \underline{a}_0$.)

Graphically, we have the following representation:



where $w_0 \stackrel{\text{def}}{=} \underline{\widetilde{c}}_0 - \underline{b}_0$, $w_1 \stackrel{\text{def}}{=} w_0 + (\underline{b}_1 - \underline{b}_0)$, and $w_2 \stackrel{\text{def}}{=} \underline{\widetilde{c}}_1 - \underline{\widetilde{c}}_0$.

Overall, the set of all possible design values is the union of these two sets. Let us illustrate this union on a simple numerical example when $\underline{b}_0 = \underline{\widetilde{c}}_0 = 0$, $\underline{b}_1 = 1$, and $\underline{\widetilde{c}}_1 = 2$. In this case, the first case corresponds to the following inequalities

$$\underline{a}_0 \geq 1;$$

$$0 \leq \underline{a}_1 - \underline{a}_0 \leq 1. \tag{9.11}$$

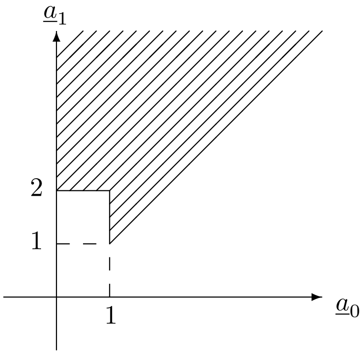
and the second case leads to the following inequalities:

$$\underline{a}_0 \geq 0;$$

$$\underline{a}_1 - \underline{a}_0 \geq 1; \tag{9.12}$$

$$\underline{a}_1 \geq 2.$$

Thus, the union of the two corresponding sets has the following form:



10 A Designed System Usually Consists of Several Subsystems

A designed system usually consists of several subsystems. So, instead of selecting a *single* p-box for a single design parameter a, we need to design p-boxes corresponding to *all* these subsystems.

Let S denote the number of these subsystems, and let $\underline{a}_i^{(s)}$, $\underline{b}_i^{(s)}$, and $\widetilde{\underline{c}}_i^{(s)}$ denote quantile bounds corresponding to the s-th subsystem, $s = 1, 2, \ldots, S$.

Thus, we arrive at the following problem:

- we know the values $\underline{b}_i^{(s)}$;
- we are given the values $\widetilde{\underline{c}}_i^{(s)}$;
- we must find, for each $s = 1, \ldots, S$, the values
$$\underline{a}_0^{(s)} \leq \ldots \leq \underline{a}_n^{(s)}$$
for which
$$\widetilde{\underline{c}}_i^{(s)} \leq \max_j (\underline{a}_j^{(s)} + \underline{b}_{i-j}^{(s)}). \tag{10.1}$$

11 There Exist Effective Algorithms for Backcalculation

For p-boxes, there are efficient algorithms for solving the backcalculation problem; see, e.g., [1, 3, 4, 7].

12 Need for Additional Cost Constraints

In general, as we can see, the backcalculation problem has many possible solutions. Some design solutions require less efforts, some require more efforts. It is therefore desirable not just to find *a solution*, but rather to find a solution which satisfies given constraints on the manufacturing efforts such as cost, energy expenses, etc.

The values $\underline{a}_i^{(s)}$ are the lower bounds on the design parameters. The smaller the lower bounds, the easier it is to maintain them. Thus, the cost of maintaining a lower bound increases with the value $\underline{a}_i^{(s)}$.

In this paper, we show that the problem is NP-hard even for the simplest case when the corresponding effort is simply proportional to the value $\underline{a}_i^{(s)}$, and thus, the overall effort of maintaining all the characteristics $\underline{a}_i^{(s)}$ is equal to the weighted linear combination
$$E = \sum_{s=1}^{S} \sum_{i=0}^{n} w_i^{(s)} \cdot \underline{a}_i^{(s)}. \tag{12.1}$$

The corresponding constraint is that this effort should not exceed a given value e.

As we have mentioned, we may have several (C) constraints corresponding to different type of effort – cost, energy consumption, etc. Thus, in general, the constrained backcalculation problem takes the following form.

13 Formulation of the Problem in Precise Mathematical Terms

We are given:

- positive integers n, S, and C;
- the values $\underline{b}_i^{(s)}$ corresponding to different $s = 1, \ldots, S$ and $i = 0, \ldots, n$;
- the values $\widetilde{\underline{c}}_i^{(s)}$ corresponding to different $s = 1, \ldots, S$ and $i = 0, \ldots, n$;
- the values e_c corresponding to different $c = 1, \ldots, C$, and
- the values $w_{c,i}^{(s)}$ corresponding to different $s = 1, \ldots, S$, $c = 1, \ldots, C$, and $i = 0, \ldots, n$.

We must find for each $s = 1, \ldots, S$, the values

$$\underline{a}_0^{(s)} \leq \ldots \leq \underline{a}_n^{(s)}$$

for which the following two sets of inequalities are satisfied:

$$\widetilde{\underline{c}}_i^{(s)} \leq \max_j (\underline{a}_j^{(s)} + \underline{b}_{i-j}^{(s)}); \tag{13.1}$$

$$\sum_{s=1}^{S} \sum_{i=0}^{n} w_{c,i}^{(s)} \cdot \underline{a}_i^{(s)} \leq e_c. \tag{13.2}$$

14 Our Main Result

Our main result is that the above problem is NP-hard.

15 Proof

Main idea. Formally, NP-hard means that an arbitrary problem from a certain class NP can be reduced to this problem; see, e.g., [6]. Thus, to prove that a problem is NP-hard, it is sufficient to prove that a known NP-hard problem can be reduced to it. Indeed,

- by definition of NP-hardness, every problem with the class NP can be reduced to the known NP-hard problem;
- since this known problem can be reduced to our problem,
- we can therefore conclude that every problem form the class NP can be reduced to our problem;
- in other words, we can conclude that our problem is NP-hard.

In our proof, as such a known NP-hard problem, we take the *knapsack* problem; see, e.g., [6]. In this problem, we are given a set of S objects, for each of which we know its volume $v_s > 0$ and its price $p_s > 0$. We also know the total volume V of a knapsack and the threshold price P. Within the restriction on the volume, we must select some of the S objects in such a way that the total price of all the selected objects is at least P.

To describe this problem in precise terms, for each object i, we define a new variable x_s such that $x_s = 1$ if the s-th object is taken and $x_s = 0$ if the s-th object is not taken. In terms of these new variables, the overall volume of all the selected objects is equal to $\sum\limits_{s=1}^{S} v_s \cdot x_s$, and the overall price of all the selected objects is equal to $\sum\limits_{s=1}^{S} p_s \cdot x_s$. Thus, the knapsack problem takes the following form: find the values $x_s \in \{0, 1\}$ for which

$$\sum_{s=1}^{S} v_s \cdot x_s \le V \tag{15.1}$$

and

$$\sum_{s=1}^{S} p_s \cdot x_s \ge P. \tag{15.2}$$

We will prove that this problem can be reduced to the above backcalculation problem, i.e., that for each instance $v_1, \ldots, v_S, p_1, \ldots, p_S, V, P$ of the knapsack problem there is an instance of the backcalculation problem whose solution can effectively lead to the solution of the original knapsack problem.

Towards reduction: selection of p-boxes. In our reduction, we will use the same pair of p-boxes b, c for all S subsystems, the only difference will be in the weights.

Let us denote the common value of $\underline{b}_i^{(s)}$ for all $s = 1, \ldots, S$ by b_i and the common value of $\underline{\widetilde{c}}_i^{(s)}$ by c_i. In these notations, the constrains on the unknowns $\underline{a}_i^{(s)}$ take a simplified form

$$c_i \le \max_j (\underline{a}_j^{(s)} + b_{i-j}). \tag{15.3}$$

In our reduction, we take $n = 1$. For $n = 1$, for every s, the inequalities (15.3) lead to the following constraints on the corresponding two unknown $\underline{a}_0^{(s)} \le \underline{a}_1^{(s)}$:

$$c_0 \le \underline{a}_0^{(s)} + b_0; \tag{15.4}$$

$$c_1 \le \max(\underline{a}_0^{(s)} + b_1, \underline{a}_1^{(s)} + b_0). \tag{15.5}$$

Specifically, we take $b_0 = c_0 = 0$, $b_1 = 1$, and $c_1 = 2$. For these values, the above inequalities take the following form:

$$\underline{a}_0^{(s)} \ge 0; \tag{15.6}$$

$$2 \le \max(\underline{a}_0^{(s)} + 1, \underline{a}_1^{(s)}). \tag{15.7}$$

The largest of the two values is greater than or equal to 2 if and only if (at least) one of these two values is greater than or equal to 2. Thus, the second constraint (15.7) means that:

- either $2 \le \underline{a}_0^{(s)} + 1$ and thus $\underline{a}_0^{(s)} \ge 1$
- or $\underline{a}_1^{(s)} \ge 2$.

Analysis of the selected p-boxes. Let us show that out of all possible solutions $\underline{a}_0^{(s)} \le \underline{a}_1^{(s)}$ satisfying these two inequalities, only two solutions satisfy the additional constraint $\underline{a}_0^{(s)} + \underline{a}_1^{(s)} \le 2$:

- $\underline{a}_0^{(s)} = \underline{a}_1^{(s)} = 1$ and
- $\underline{a}_0^{(s)} = 0$ and $\underline{a}_1^{(s)} = 2$.

Indeed, we know that for each solution, we have:

- either $\underline{a}_0^{(s)} \ge 1$
- or $\underline{a}_1^{(s)} \ge 2$.

In the first case $\underline{a}_0^{(s)} \ge 1$, we have

$$\underline{a}_0^{(s)} + \underline{a}_1^{(s)} = 2\underline{a}_0^{(s)} + (\underline{a}_1^{(s)} - \underline{a}_0^{(s)}). \tag{15.8}$$

The first term in the right-hand side is ≥ 2, the second term is always non-negative – since $\underline{a}_1^{(s)} \ge \underline{a}^{(s)}$. Thus, the only possibility for the right-hand side sum to be ≤ 2 is when the value $2\underline{a}_0^{(s)}$ is exactly 2, and the difference $\underline{a}_1^{(s)} - \underline{a}_0^{(s)}$ is exactly 0. In this case, we have $\underline{a}_0^{(s)} = \underline{a}_1^{(s)} = 1$.

In the second case $\underline{a}_1^{(s)} \ge 2$, due to $\underline{a}_0^{(s)} \ge 0$ (condition (15.6)), the only possibility for the sum $\underline{a}_0^{(s)} + \underline{a}_1^{(s)}$ to be ≤ 2 is when $\underline{a}_0^{(s)} = 0$ and $\underline{a}_1^{(s)} = 2$.

Reduction and the final part of the proof. We will reduce the given instance of the knapsack problem to the following system with 3 constraints:

- In the first constraint, we take $w_{1,i}^{(s)} = 1$ for all s and i, and we take $e_1 = 2 \cdot S$.
- In the second constraint, we take $w_{2,0}^{(s)} = v_s$, $w_{2,1}^{(s)} = 0$, and $e_2 = V$.

- In the third constraint, we take $w_{2,0}^{(s)} = 0$, $w_{2,1}^{(s)} = p_s$, and

$$e_3 = P - 2 \cdot \sum_{s=1}^{S} p_s. \tag{15.9}$$

Let us first analyze the first constraint

$$\sum_{s=1}^{S} (\underline{a}_0^{(s)} + \underline{a}_1^{(s)}) \leq 2 \cdot S. \tag{15.10}$$

As we have shown in the previous section, each simple sum $\underline{a}_0^{(s)} + \underline{a}_1^{(s)}$ is at least 2, and it is only equal to 2 when either $\underline{a}^{(s)} = 0$ or $\underline{a}_0^{(s)} = 1$. Thus, the only possibility for the sum $\sum\limits_{s=1}^{S} (\underline{a}_0^{(s)} + \underline{a}_1^{(s)})$ of S such simple sums to not exceed $2S$ is when each of these sums is equal to exactly 2, i.e., when for every s,

- either either $\underline{a}^{(s)} = 0$ or $\underline{a}_0^{(s)} = 1$, and
- $\underline{a}_1^{(s)} = 2 - \underline{a}_0^{(s)}$.

For these values $\underline{a}_i^{(s)}$, the second constraint takes the form

$$\sum_{s=1}^{S} v_s \cdot \underline{a}_0^{(s)} \leq V, \tag{15.11}$$

and the third constraint takes the form

$$\sum_{s=1}^{S} p_s \cdot \underline{a}_1^{(s)} \leq 2 \cdot \sum_{s=1}^{S} p_s - P. \tag{15.12}$$

Substituting the expression $\underline{a}_1^{(s)} = 2 - \underline{a}_0^{(s)}$ into this inequality, we get

$$\sum_{s=1}^{S} p_s \cdot (2 - \underline{a}_0^{(s)}) \leq 2 \cdot \sum_{s=1}^{S} p_s - P, \tag{15.13}$$

i.e., equivalently,

$$2 \sum_{s=1}^{S} p_s - \sum_{s=1}^{S} p_s \cdot \underline{a}_0^{(s)} \leq 2 \sum_{s=1}^{S} p_s - P, \tag{15.14}$$

which, in its turn, is equivalent to

$$\sum_{s=1}^{S} p_s \cdot \underline{a}_0^{(s)} \geq P, \tag{15.15}$$

Thus, for every solution to the constrained backcalculation problem, the values $x_s \stackrel{\text{def}}{=} \underline{a}_0^{(s)}$ form a solution to the knapsack problem: each of them is equal to 0 or 1, and they satisfy the corresponding inequalities (15.11) and (15.15).

Vice versa, one can easily check that if the values x_s form a solution to the knapsack problem, then the corresponding values $\underline{a}_0^{(s)} = x_s$ and $\underline{a}_1^{(s)} = 2 - x_s$ form a solution to the constrained backcalculation problem.

The reduction is proven, so the backcalculation problem is indeed, in general, NP-hard.

16 Situation in Which a Feasible Algorithm Is Possible

Let us describe an example of an optimization problem in which a feasible algorithm is possible.

Optimal backcalculation: towards the formulation of the problem in precise terms. In general, there are many combinations of the values $\underline{a}_i$ that satisfy the desired constraints. Which combination should we choose?

In the design case, a natural requirement is to select the values $\underline{a}_i$ which will be the easiest to implement. To describe this idea in precise terms, we must analyze how easy is it to implement different values of $\underline{a}_i$.

In general, the value $\underline{a}_i$ is the value for which $\overline{F}(\underline{a}_i) = \dfrac{i}{n}$, i.e., we which we must guarantee that

$$\text{Prob}(a \le \underline{a}_i) = F(\underline{a}_i) \le \overline{F}(\underline{a}_i) = \frac{i}{n}. \tag{16.1}$$

In particular:

- once we select the value $\underline{a}_0$, we must guarantee that the probability $\text{Prob}(a \le \underline{a}_0)$ of the actual value a being below $\underline{a}_0$ is 0:
$$\text{Prob}(a \le \underline{a}_0) = 0;$$
- once we select $\underline{a}_1$, we must guarantee that the probability $\text{Prob}(a \le \underline{a}_1)$ of the actual value a being below $\underline{a}_1$ does not exceed $1/n$:
$$\text{Prob}(a \le \underline{a}_1) \le 1/n;$$
- once we select $\underline{a}_2$, we must guarantee that the probability $\text{Prob}(a \le \underline{a}_2)$ of the actual value a being below $\underline{a}_2$ does not exceed $2/n$:
$$\text{Prob}(a \le \underline{a}_2) \le 2/n;$$
- etc.

When we motivated the need to take into account partial information about the probabilities, we have mentioned that the most difficult task is to guarantee that the actual a *never* gets below a threshold. The corresponding restriction is related to the value $\underline{a}_0$: we must guarantee that $a \geq \underline{a}_0$. The smaller this value $\underline{a}_0$, the weaker this constraint and thus, the easiest to satisfy. Thus, it is reasonable to select the smallest possible value $\underline{a}_0$.

Once this value is selected and the corresponding bound is guaranteed, we must guarantee that the inequalities $a \geq \underline{a}_i$ be guaranteed with the probabilities $\geq 1 - i/n$. The closer this probability to 1, the more stringent is the corresponding requirement, and thus, the more difficult the corresponding task. So, after we have selected $\underline{a}_0$, the most difficult of the remaining tasks is to select the value $\underline{a}_1$, the value for which the guaranteed probability of violating the restriction $a \geq \underline{a}_1$ is the smallest (probability $= 1/n$). Thus, to make the design as easy to implement as possible, we should make this restriction the least difficult to implement – i.e., we should select the value $\underline{a}_1$ as small as possible.

Once we have fixed $\underline{a}_0$ and $\underline{a}_1$, the most difficult of the remaining tasks is to to guarantee that $a \geq \underline{a}_2$ with probability $\geq 1 - (2/n)$. Thus, we must select the corresponding threshold $\underline{a}_2$ to be as small as possible.

As a result, we arrive at the following formulation of the backcalculation problem.

Formulation of the optimal backcalculation problem in precise mathematical terms. Out of all the tuples $\underline{a}_0 \leq \ldots \leq \underline{a}_n$ that satisfy the inequalities (8.1),

- we first select all the tuples for which the value $\underline{a}_0$ is the smallest possible;
- out of the selected tuples, we select all the tuples for which the value $\underline{a}_1$ is the smallest possible;
- then, out of the newly selected tuples, we select those for which the value $\underline{a}_2$ is the smallest possible;
- etc.

In mathematical terms, we can say that a tuple $\underline{a} = (\underline{a}_0, \ldots, \underline{a}_n)$ is *better* than a tuple $\underline{a}' = (\underline{a}'_0, \ldots, \underline{a}'_n)$ if one of the following conditions hold:

- either $\underline{a}_0 < \underline{a}'_0$;
- or $\underline{a}_0 = \underline{a}'_0$ and $\underline{a}_1 < \underline{a}'_1$;
- or $\underline{a}_0 = \underline{a}'_0$, $\underline{a}_1 = \underline{a}'_1$, and $\underline{a}_2 < \underline{a}'_2$;
- $\ldots$
- or $\underline{a}_0 = \underline{a}'_0$, $\ldots$, $\underline{a}_{i-1} = \underline{a}'_{i-1}$, and $\underline{a}_i < \underline{a}'_i$;
- $\ldots$

- or $\underline{a}_0 = \underline{a}'_0$, …, $\underline{a}_{n-1} = \underline{a}'_{n-1}$, and $\underline{a}_n < \underline{a}'_n$.

In computer science, this relation is known as a *lexicographic* (alphabetic) order, since this is exactly how words are placed in a dictionary or in a lexicon: a word $w = \ell_0\ell_1\ldots$ consisting of the letters ℓ_0, ℓ_1, …, is placed before a word $w' = \ell'_0\ell'_1\ldots$ consisting of the letters ℓ'_0, ℓ'_1, … if one of the following conditions hold:

- either the letter ℓ_0 precedes the letter ℓ'_0 (e.g., ∅ apple goes before ∅ zebra);
- or $\ell_0 = \ell'_0$ and the next letter ℓ_1 precedes the corresponding letter ℓ'_1 (e.g., ∅ abuse goes before ∅ alpha);
- …
- or $\ell_0 = \ell'_0$, …, $\ell_{i-1} = \ell'_{i-1}$, and the next letter ℓ_i precedes the corresponding letter ℓ'_i;
- …

In these terms, we must select the tuple which is the smallest in the lexicographic order.

Towards a solution to the optimal backcalculation problem. To find the optimal solution, let us start with the value $\underline{a}_0$. For $i = 0$, the condition (8.1) becomes $\underline{a}_0 + \underline{b}_0 \le \widetilde{\underline{c}}_0$, i.e., equivalently, $\underline{a}_0 \ge \widetilde{\underline{c}}_0 - \underline{b}_0$. The smallest real number that satisfies this inequality is the value $\underline{a}_0 = \widetilde{\underline{c}}_0 - \underline{b}_0$. Thus, we should take

$$\underline{a}_0 = \widetilde{\underline{c}}_0 - \underline{b}_0. \tag{16.2}$$

Let us now assume that we have already selected the values $\underline{a}_0, \ldots, \underline{a}_{i-1}$, and that we are now selecting the value $\underline{a}_i$. The i-th condition (8.1) has the form

$$\widetilde{\underline{c}}_i \le \max_j(\underline{b}_{i-j} + \underline{a}_j) = \max\left[\max_{j\le i-1}(\underline{b}_{i-j} + \underline{a}_j), \underline{b}_0 + \underline{a}_i\right]. \tag{16.3}$$

If

$$\widetilde{\underline{c}}_i \le \max_{j\le i-1}(\underline{b}_{i-j} + \underline{a}_j), \tag{16.4}$$

then the condition (16.3) is already satisfied. In this case, the only restriction on $\underline{a}_i$ is that $\underline{a}_i \ge \underline{a}_{i-1}$; thus, the smallest possible value of $\underline{a}_i$ is $\underline{a}_i = \underline{a}_{i-1}$.

If the inequality (16.4) is not satisfied, then to satisfy (16.3), we must satisfy the inequality $\widetilde{\underline{c}}_i \le \underline{b}_0 + \underline{a}_i$, i.e., equivalently, $\underline{a}_i \ge \widetilde{\underline{c}}_i - \underline{b}_0$. Thus, the smallest possible value here is $\underline{a}_i = \widetilde{\underline{c}}_i - \underline{b}_0$. So, we arrive at the following algorithm:

Algorithm for optimal backcalculation. ([1, 3, 4, 7]) We know:

- the interval quantiles $[\underline{b}_i, \overline{b}_i]$ for the environmental variable b; and
- the interval quantiles $[\underline{\widetilde{c}}_i, \overline{\widetilde{c}}_i]$ for c.

We want to find the lexicographically optimal interval quantiles $[\underline{a}_i, \overline{a}_i]$ for a for which c satisfies the given constraints.

In this algorithm, we compute the values $\underline{a}_0, \underline{a}_1, \ldots$ one by one.

- First, we compute $\underline{a}_0 = \underline{\widetilde{c}}_0 - \underline{b}_0$.
- Once the values $\underline{a}_0, \ldots, \underline{a}_{i-1}$ are computed, we check whether
$$\underline{\widetilde{c}}_i \le \max_{j \le i-1} (\underline{b}_{i-j} + \underline{a}_j). \tag{16.5}$$
If this inequality is satisfied, we take $\underline{a}_i = \underline{a}_{i-1}$; otherwise, we take $\underline{a}_i = \underline{\widetilde{c}}_i - \underline{b}_0$.

Comment. In the derivation of the algorithm, we, in fact, proved that the result $\underline{a}$ of applying this algorithm always satisfies the inequalities (8.1): indeed, we have selected each value $\underline{a}_i$ in such a way that the i-th inequality (8.1) is satisfied.

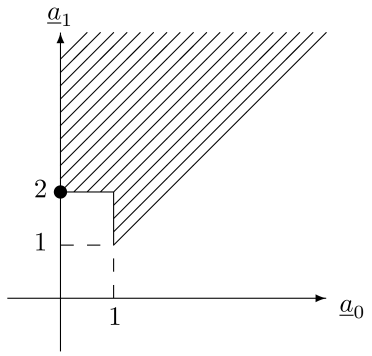
We have also shown that no tuple satisfying (8.1) can be (lexicographically) better than the result $\underline{a}$ of using this algorithm – and thus, this result $\underline{a}$ is indeed optimal.

Algorithm illustrated on the above simple example. In the above example when $\underline{b}_0 = \underline{\widetilde{c}}_0 = 0$, $\underline{b}_1 = 1$, and $\underline{\widetilde{c}}_1 = 2$, we do the following:

- First, we compute $\underline{a}_0 = \underline{\widetilde{c}}_0 - \underline{b}_0 = 0 - 0 = 0$.
- Next, we check whether
$$\underline{\widetilde{c}}_1 \le \max_{j \le 0} (\underline{b}_{i-j} + \underline{a}_j) = \underline{b}_1 + \underline{a}_0. \tag{16.6}$$
Here, we have $2 = \underline{\widetilde{c}}_1 \not\le \underline{b}_1 + \underline{a}_0 = 1 + 0 = 1$, so we take $\underline{a}_1 = \underline{\widetilde{c}}_1 - \underline{b}_0 = 2 - 0 = 2$.

One can easily check that the resulting tuple $(\underline{a}_0, \underline{a}_1) = (0, 2)$ is indeed lexicographically optimal:

- it has the smallest possible value of $\underline{a}_0 = 0$, and
- out of all the tuples with this value of $\underline{a}_0$, it has the smallest possible value of $\underline{a}_1$.



Acknowledgments

This work was supported in part by NSF grant HRD-0734825 and by Grant 1 T36 GM078000-01 from the National Institutes of Health.

Received: June 2009. Revised: November 2009.

References

[1] S. Ferson, *Using approximate deconvolution to estimate cleanup targets in probabilistic risk analyses*, In: P. T. Kostecki, E. J. Calabrese, and M. Bonazountas (eds.), Hydrocarbon Contaminated Soils, Amherst Scientific Publishers, Amherst, Massachusetts, 1995, pp. 245–254.

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

[3] S. Ferson, 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.

[4] S. Ferson and T. F. Long, *Deconvolution can reduce uncertainty in risk analyses*, In: M. Newman and C. Strojan (eds.), Risk Assessment: Measurement and Logic, Ann Arbor Press, Ann Arebor, Michigan, 1997.

[5] V. Kreinovich, *Expert knowledge is needed for design under uncertainty: for p-boxes, backcalculation is, in general, NP-hard*, Proceedings of the 2009 Annual Conference of the North American Fuzzy Information Processing Society NAFIPS'09, Cincinnati, Ohio, June 14–17, 2009.

[6] C. H. Papadimitriou, *Computational Complexity*, Addison Wesley, 1993.

[7] W. T. Tucker and S. Ferson, *Setting cleanup targets in a probabilistic assessment*, In: S. Mishra (ed.), Groundwater Quality Modeling and Management under Uncertainty, American Society of Civil Engineers, Reston, Virginia, 2003.

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