Two Algorithms for Calibration of Precautionary Filters

Toni Jarimo
Umwelt und Risikovorsorge
ETH Zürich

July 8, 2002
# Contents

1 **Introduction** 2

2 **Description of Filters** 2
   2.1 Pösets, Single Filters and Filter Systems ............... 2
   2.2 Calibration of Filters ........................................ 5

3 **Mathematical Formulation of Filter Calibration Problem** 6
   3.1 Filter Calibration Problem as an Optimisation Problem ... 6
   3.2 Formulation of the Filter Calibration Problem as a LOCO-
       problem ............................................................ 7

4 **Methods for Solving the Filter Calibration Problem** 10
   4.1 Binary Linear Programming Model ............................ 10
       4.1.1 Short Introduction to Binary Linear Programming ... 10
       4.1.2 Variables ................................................... 11
       4.1.3 Constraints ................................................ 12
       4.1.4 Objective Function ....................................... 14
   4.2 The Cutting Algorithm .......................................... 15
   4.3 Improved Cutting Algorithm for Two-dimensional Filters ... 18

5 **Discussion** 20

6 **Summary and Conclusions** 21

**References** 23
1 Introduction

Managing the global risk factors caused by new (and existing) chemicals is a task of high importance for the well-fare of the whole world. Aiming to precautionary screening of new chemicals, a filter series approach to the problem has been introduced by Müller-Herold (2002). In Müller-Herold’s work, the object is to classify new chemicals by comparing some specific properties of the new chemicals with reference data. In this case, the reference data consists of already existing chemicals, for which the chemical properties are known. It is also predetermined, whether an existing chemical is historically regulated (red), or allowed (green). On the grounds of comparing the properties of new and existing chemicals, a new chemical is classified into one of the three grades: green (allowed), red (regulated) and yellow (needs further consideration).

The comparison of the chemical properties is performed by a series of (or possibly by only one) precautionary filters. The filters have been individually calibrated so, that of the reference data, in the ideal case, they let the predefined green chemicals pass, but stop the red chemicals. Descriptively, calibrating a filter means defining the mesh-width of a filter. Now as the new chemicals are filtered through the filter system, the chemicals that do not pass the system are classified as red, and the chemicals that may pass, are classified as green or yellow. For a detailed description of filter series approach and an illustrative example, see Müller-Herold et al. (2001), Müller-Herold (2002).

The calibration of the filters brings out a multicriteria optimisation problem. In order to build a sensible filter system, the filters have to be calibrated in a way, that of the predefined existing chemicals a maximum number of red chemicals are stopped and a maximum number of green chemicals are let pass.

In this paper, the filter calibration problem will be transformed into a Linear Objective Combinatorial Optimisation (LOCO) problem. From the general formulation of the LOCO problem will be derived a Binary Linear Programming (BLP) model, which in itself is solvable by known algorithms (see, e.g., Taha 1997). Also, for directly solving the filter calibration problem, we will introduce an algorithm, named the Cutting Algorithm.

2 Description of Filters

2.1 Posets, Single Filters and Filter Systems

The partially ordered set, poset \( P \), is a mathematical structure, which typically is formulated by two sets \( X \) and \( P \) (\( P := (X, P) \)). The set \( X \) consists of
the elements in question (e.g. chemicals), each having $D$ properties, namely $p_1, p_2, \ldots, p_D$. We denote the $i^{th}$ property of an element $a \in X$ by $p_i(a)$. Between two elements, say $a$ and $b$, of the set $X$, a comparison between each property is possible, i.e. for all $a, b \in X$ it can be said, whether $p_i(a)$ is greater than, equal or less than $p_i(b)$. The set $P$ contains all pairs of elements $(a, b)$, $a, b \in X$, for which $p_i(a) \leq p_i(b)$, $\forall i \in \{1, \ldots, D\}$. For a detailed description of posets, see, e.g., Graham et. al (1995).

The purpose of a precautionary filter is to classify the elements of a set $X \subset \mathcal{P}$. For example, in the case of precautionary screening of chemicals, the elements of $X$ are chemicals with specific chemical properties, e.g. spatial range, persistence etc. The poset formulation applied to environmental chemicals is introduced in the paper of Schucht (2002). The notation presented by Schucht will be used throughout this paper, whenever referring to posets.

The dimension of a filter is the number of the properties of an element in set $X$, that are taken into account in using the filter. Every property corresponds to one dimension in $D$-dimensional space, i.e. there are $D$ property axes.\footnote{The dimension of a filter should not be confused with the dimension of a poset. For information on the dimension of posets, see Trotter (1992).} For each axis two points are defined, namely $p_{i,lm}$ and $p_{i,mh}$, such that

$$p_{i,lm} \leq p_{i,mh}.$$  \hspace{1cm} (1)

These points divide each axis into three sections:

- **low**: $p_i \leq p_{i,lm}$ (lm=low-medium)
- **medium**: $p_{i,lm} < p_i \leq p_{i,mh}$ (mh=medium-high)
- **high**: $p_i > p_{i,mh}$

In $D$-dimensional space, these sections define $3^D$ classes, which can be coded by an ordered series of the sections. Geometrically these classes are rectangular hyper-prisms in the $D$-dimensional space. Figure 1 illustrates the classes of a two-dimensional filter. In the figure—and in the following—a class will be denoted by a series of discrete values of 1, 2 and 3, where 1 is a shorthand for “low”, 2 for “medium” and 3 for “high” value of a property. E.g. in the case of Figure 1, (2, 3) stands for “medium” value of property 1 and “high” value of property 2. Notation \{2, 3\} is used to describe unordered classification, i.e. either property 1 has a “medium” value and property 2 has a “high” value or property 1 has a “high” value and property 2 has a “medium” value.

An element with properties $p_1, \ldots, p_D$ is obviously placed in exactly one of the classes, thus the element can be classified through the code of the hyper-
Figure 1: Two-dimensional filter with nine classes and three output grades

prism. In the present work of Müller-Herold et al. (2001) and Schucht (2002), a precautionary filter classifies elements with three grades:

- green: “may pass”
- yellow: “needs further consideration”
- red: “will be stopped”.

Hence, the output of a filter classification is a grade of “green”, “yellow” or “red”. Grade “green” implies that the element may pass through the filter, “red” implies that the element will be stopped by the filter, and elements classified as “yellow” will be taken under further consideration. With a two-dimensional filter (as presented in Figure 1), grade “green” will be achieved in classes \{1,1\} and \{1,2\}, “yellow” in \{1,3\} and \{2,2\} and “red” in \{2,3\} and \{3,3\} (Müller-Herold 2002). At present, the classification scheme of higher dimensional filters is not yet properly defined, but will be discussed in this paper later on.

A filter series (or system) consists of successive filters, each taking care of some specific properties, which are related to different threat scenarios. For instance, in Müller-Herold’s (2002) work the “Pandora” filter is constructed
in respect of two intrinsic chemical properties, mobility and longevity, in
order to manage the threat scenario of global dispersion. The output of
a filter series has three grades, similarly to a single filter. The grades are
defined as follows:

green: element is classified as “green” in every filter of the series
red: element is classified as “red” in one or more filters of the
series
yellow: otherwise.

Hence, the output of a filter series is independent of the order of the single
filters, of which it is constructed. Also, every single filter in the series can—
and should—be individually calibrated for best performance. This allows the
removal and addition of single filters from and into the series, without having
to re-calibrate all the filters, but only the new filter entering the series.

2.2 Calibration of Filters

Filter calibration conceptually means fixing the values of \( p_{i,lm} \) and \( p_{i,mh} \) for
each property \( i \in \{1, 2, \ldots, D\} \) so, that the output of the filter has best
possible correspondence with reference data. Our reference data is a set \( X \)
of elements with properties \( p_1, \ldots, p_D \). Note, that with these properties, the
elements of \( X \) form a poset \( P = (X, P) \).

In addition, all elements of \( X \) are, e.g. on historical grounds, divided into
“green” and “red” elements. Let \( H = (X, H) \) be a poset, which has exactly
one property, denoted by \( h \), which is defined for all the elements in \( X \) as
follows:

\[
h(a) := \begin{cases} 
1, & \text{if } a \text{ is “red”} \\
0, & \text{if } a \text{ is “green”}
\end{cases} \quad \forall \ a \in X. 
\tag{2}
\]

Thus, \( X = G \cup R \) and \( G \cap R = \emptyset \), where \( G := \{ a \in X \mid h(a) = 0 \} \) and
\( R := \{ a \in X \mid h(a) = 1 \} \).

However, with fixed values of \( p_{i,lm} \)’s and \( p_{i,mh} \)’s, also the filter gives a classification of “green” or “red” (or “yellow”). Thus, there exists four basic types
of post-filter outcomes for the elements of the poset, that are being used to
calibrate the filter:

a. green in reality, green by filter classification
b. red in reality, green by filter classification
c. green in reality, red by filter classification
d. red in reality, red by filter classification.

From now on, by reference classification we mean the real classification of
an element of \( X \), and by filter classification we mean the classification, that
an element of \( X \) has by the filter with a specified calibration of the filter (i.e. fixed \( p_{i,lm} \)'s and \( p_{i,mh} \)'s). The \( p_{i,lm} \)'s and \( p_{i,mh} \)'s should be fixed aiming to the following goals:

1. the number in c should be minimal
2. the number in d should be maximal
3. the number in a should be maximal
4. the number in b should be minimal.

The order above is used as the prioritisation of the goals, in order to have a well-defined optimum for the \( p_{i,lm} \)'s and \( p_{i,mh} \)'s. The prioritisation follows from the fact, that elements classified as red cannot be retrieved later on. On the other hand, the elements wrongly classified as green (b) can still be stopped by another filter, thus giving the highest priority for minimising elements wrongly classified as red (c) has good grounds (see Müller-Herold et al. 2001, Schucht 2002).

3 Mathematical Formulation of Filter Calibration Problem

3.1 Filter Calibration Problem as an Optimisation Problem

We first formulate the filter calibration problem as a conventional optimisation problem, i.e. we define

- variables
- constraints
- objective function.

The variables of the problem are the boundaries \( p_{i,lm} \in \mathbb{R} \) and \( p_{i,mh} \in \mathbb{R} \) for all \( i \in \{1,2,\ldots,D\} \). These continuous variables define the sections low, medium and high for each of the properties. \( D \) describes the number of properties, i.e. the dimension of the filter being calibrated. From the nature of the variables, we have one constraint for every property, that is the inequality (1), discussed in Section 2.1: \( p_{i,lm} \leq p_{i,mh}, \forall i \in \{1,2,\ldots,D\} \).

For the objective function we give the following form:

\[
\text{max} \quad z = c_an_a - c_bn_b - c_cn_c + c_dn_d, \tag{3}
\]

where \( n_a, n_b, n_c, n_d \) represent the numbers of elements in a, b, c and d, respectively (see Section 2.2). It is noticeable that \( n_a, n_b, n_c, n_d \) are functions
of \( p_{i,lm} \) and \( p_{i,mh} \). Being a multicriteria optimisation problem—maximise \( n_a \) and \( n_d \), minimise \( n_b \) and \( n_c \)—some sort of prioritisation between the goals has to be defined. Hence, constants \( c_a, c_b, c_c, c_d \) are given as non-negative weights for classifications \( a, b, c \) and \( d \). E.g., if \( c_a = 2 \cdot c_b \), then, in the objective function, every two elements classified as \( b \) correspond to one element classified as \( a \), thus giving a higher priority for elements in \( a \). For more methods of multicriteria optimisation, see, e.g., Statnikov and Matusov (1995), Stadler (1988).

In order to implement the prioritisation discussed in Section 2.2 we introduce the following values for the coefficients \( c_a, c_b, c_c \) and \( c_d \):

\[
\begin{align*}
    c_c &= N \cdot c_d \\
    c_d &= N \cdot c_a \\
    c_a &= N \cdot c_b \\
\end{align*}
\]

(4)

where \( c_b \) can be chosen freely, e.g. \( c_b = 1 \). The use of above coefficients implies a strict prioritisation. Considering the objective function (3) this means, that it is always better to have, e.g., one by reference “red” element having filter classification “red” than any amount of by reference “green” elements having filter classification “green”.

Summarising the filter calibration problem, we have defined the following optimisation problem (P):

\[
\begin{align*}
    \max \quad & z(p_{1,lm}, \ldots, p_{D,lm}, p_{1,mh}, \ldots, p_{D,mh}) = c_a n_a - c_b n_b - c_c n_c + c_d n_d \\
    \text{so that} \quad & p_{i,lm} - p_{i,mh} \leq 0, \quad \forall i \in \{1,2,\ldots,D\} \\
    & p_{i,lm}, p_{i,mh} \in \mathbb{R}, \quad \forall i \in \{1,2,\ldots,D\} \\
    & c_a, c_b, c_c, c_d \geq 0, \quad \text{fixed}.
\end{align*}
\]

It is noteworthy that the variables of (P) are continuous, but the objective function is discrete, which leads to a more complicated optimisation task. In the following section (3.2) we will introduce a combinatorial approach on the filter calibration problem.

### 3.2 Formulation of the Filter Calibration Problem as a LOCO-problem

The LOCO-problem is a shorthand for Linear Objective Combinatorial Optimisation problem and is defined as follows (Nemhauser & Wolsey 1988):

**Definition 3.1 (LOCO-problem)** Let \( E \) be a set of elements, and for every element \( e \in E \) is given a value \( c_e \). Let \( \mathcal{F} \) be a family of subsets of \( E \), that
are characterised by a property $\Pi$, i.e. $\mathcal{F} := \{ F \subseteq E \mid F \text{ follows property } \Pi \}$.

The value of a subset $F \in \mathcal{F}$ is given by the sum of the values of its elements:

$$c(F) := \sum_{e \in F} c_e.$$

Searched is, in this sense, the best object $F^* \in \mathcal{F}$, i.e.

$$c(F^*) = \max \{ c(F) \mid F \in \mathcal{F} \}.$$

The advantage of a LOCO-problem is, that it can be reduced to a Binary Linear Programming (BLP) model (see Hoffman 1979, Graham 1995). For a definition of a BLP model, see, e.g., Taha (1997). In his book, Taha also introduces an efficient algorithm, called the additive algorithm, designed to solve a BLP model.

In order to formulate the filter calibration problem as a LOCO-problem, we first want to transform the continuous variables $p_{i,lm}$’s and $p_{i,mh}$’s into discrete elements of the set $E$ (see Definition 3.1). Let $X$ be the set of elements, which are used as the reference data for filter calibration. The number of elements in $X$ is the cardinality $|X| = N$. Let $P = \langle X, P \rangle$ be a poset with properties $p_1, \ldots, p_D$. Let $H = \langle X, H \rangle$ be a poset with the property $h$, defined in (2). Let $p_i$ be a vector, whose elements are the values of the $h$th property of all the $N$ elements in $X$, in increasing order, with respect to

$$p_i(j) \leq p_i(j + 1) \quad \forall j \in \{1, \ldots, N - 1\},$$

where $p_i(j)$ is the $j$th element of the vector $p_i$.

Now, without loss of generality, we assume that the boundaries $p_{i,lm}$ and $p_{i,mh}$ always lie either in an open interval (not necessarily in the same) between two successive elements of $p_i$, namely $(p_i(j), p_i(j + 1))$ for $j \in \{1, \ldots, N - 1\}$, or in an open interval of $(-\infty, p_i(1))$ or $(p_i(N), \infty)$, taking constraint (1) into account. Hence, there are $N + 1$ intervals in which the $p_{i,lm}$’s and $p_{i,mh}$’s can lie. In increasing order, the intervals are:

$$\{ (-\infty, p_i(1)), (p_i(1), p_i(2)), \ldots, (p_i(N), \infty) \}. \quad (6)$$

To represent the intervals, for each property $i$ we introduce two vectors, $x_{i,lm}$ and $x_{i,mh}$. Both vectors are binary and have length $N + 1$, i.e.

$$x_{i,lm}(k), x_{i,mh}(k) \in \{0, 1\} \quad \forall i \in \{1, \ldots, D\},
\quad k \in \{1, \ldots, N + 1\}. \quad (7)$$
Also, always only one element of a vector \( \mathbf{x} \) is 1 and all the other elements are 0:

\[
\sum_{k=1}^{N+1} x_{i,lm}(k) = \sum_{k=1}^{N+1} x_{i,mb}(k) = 1 \quad \forall \ i \in \{1, \ldots, D\},
\]

with \( x_{i,lm}(k) = 1 \) (resp. \( x_{i,mb} \)) indicating that the boundary \( p_{i,lm} \) lies in the\( \ k \)th interval.

The vectors \( \mathbf{x}_{i,lm} \) and \( \mathbf{x}_{i,mb} \) also have to follow the constraint (1), or equivalently

\[
\mathbf{x}_{i,lm}(m) = 1, \mathbf{x}_{i,mb}(n) = 1 \implies m \leq n \quad \forall \ i \in \{1, \ldots, D\}.
\]

We are now ready to formulate the filter calibration problem as a LOCO-problem. We have the set \( X = G \cup R \) consisting of two disjoint subsets \( G \) and \( R \), which contain the “green” and “red” reference elements. Also, all the elements in \( X \) have properties \( p_1, \ldots, p_D \). In addition, the filter, with fixed \( \mathbf{x}_{i,lm} \)'s and \( \mathbf{x}_{i,mb} \)'s, gives a classification of “green”, “yellow” or “red” for each element in \( X \). We denote these post-filter classifications by \( C_1, C_2 \) and \( C_3 \) for “green”, “yellow” and “red”, respectively. The task is to attach each element \( g \in G \) and \( r \in R \) to one of the classes \( C_1, C_2, C_3 \) so, that feasible vectors \( \mathbf{x}_{i,lm} \) and \( \mathbf{x}_{i,mb} \) exist for each property \( i \in \{1, \ldots, D\} \). The best solution is defined by the following goals and priorisation (see Section 2.2):

1. number of g’s attached to \( C_3 \) is minimised (c)
2. number of r’s attached to \( C_3 \) is maximised (d)
3. number of g’s attached to \( C_1 \) is maximised (a)
4. number of r’s attached to \( C_1 \) is minimised (b).

Let \( E \) denote the set of all possible matchings

\[
E := \{ (g_m, C_k), (r_n, C_k) \mid m = 1, \ldots, |G|, \ n = 1, \ldots, |R|,
\]

\[
k = 1, 2, 3 \}.
\]

The value \( c_e \) of an element \( e \in E \) is the value of the according coefficient from the objective function (3)

\[
c_e := \begin{cases} 
c_a, & \text{if } e = (g, C_1), \ g \in G \\
-c_b, & \text{if } e = (r, C_1), \ r \in R \\
-c_c, & \text{if } e = (g, C_3), \ g \in G \\
-c_d, & \text{if } e = (r, C_3), \ r \in R. \\
\end{cases}
\]
\( F \) is the family of feasible subsets of \( E \)

\[
F := \{ F = \{(g_1, C_k), \ldots, (g_{|G|}, C_k), (r_1, C_k), \ldots, (r_{|R|}, C_k)\} \mid \forall i \in \{1, \ldots, D\} \exists x_{i,lm}, x_{i,mh} \text{ s.t.} \ x_{i,lm}(m) = 1, x_{i,mh}(n) = 1 \implies m \leq n \}. \tag{12}
\]

The best object \( F^* \) is the one with the maximum sum of values

\[
c(F^*) = \max_{F \in F} \sum_{e \in F} c_e. \tag{13}
\]

With (10), (11), (12) and (13) the filter calibration problem has been formulated as a LOCO-problem. In the following Section 4, the LOCO-problem will be reduced into a BLP-model. Also an algorithmic approach is introduced to solve the filter calibration problem.

4 Methods for Solving the Filter Calibration Problem

4.1 Binary Linear Programming Model

4.1.1 Short Introduction to Binary Linear Programming

A binary linear program (BLP) is an optimisation problem with

- linear objective function
- linear constraints
- binary (zero-one) variables.

A BLP can be represented in its standard form:

\[
\begin{align*}
\max & \quad z = c^T x \\
\text{s.t.} & \quad Ax \leq b \\
& \quad x \in \{0, 1\}^n,
\end{align*}
\]

where \( x \) is an \((n \times 1)\)-vector, whose elements are the binary variables \( x_1, \ldots, x_n \), \( c \) is an \((n \times 1)\)-vector, which contains the coefficients of the objective function. \( A \) is an \((m \times n)\)-matrix, which defines the coefficients of the \( m \) constraints and \( b \) is an \((m \times 1)\)-vector, containing the right hand sides of the constraints. In the following a BLP model will be introduced to solve the filter calibration problem.
4.1.2 Variables

For the BLP—similarly to the mixed optimisation problem in Section 3.1—we will define the \textit{variables} (which now all are binary), the \textit{constraints} and the \textit{objective function}. From the LOCO formulation in Section 3.2, we take directly the vectors $\mathbf{x}_{i,\text{ln}}$ and $\mathbf{x}_{i,\text{mhn}}$, defined in (7), to represent the boundaries for each property $i = 1, \ldots, D$. Note that each element of the vectors $\mathbf{x}_{i,\text{ln}}$’s and $\mathbf{x}_{i,\text{mhn}}$’s is in fact a binary variable, denoted by $x_{i,ln}(k)$ (resp. $x_{i,mhn}$).

Furthermore, let $X = G \cup R \ (|X| = N)$ be the set, whose elements are the reference data used for the filter calibration. For each element $a_j$, $j = 1, \ldots, N$ of the set $X$ and for each property $p_i$, $i = 1, \ldots, D$ we take three binary variables, namely $l_{i,j}, m_{i,j}$ and $h_{i,j}$, which represent the classification of element $j$ according to property $i$:

\[
\begin{align*}
   l_{i,j} &= \begin{cases} 
   1, & \text{if } p_i(a_j) < p_{i,\text{ln}} \\
   0, & \text{otherwise} \end{cases} \\
   m_{i,j} &= \begin{cases} 
   1, & \text{if } p_{i,\text{ln}} < p_i(a_j) < p_{i,\text{mhn}} \\
   0, & \text{otherwise} \end{cases} \\
   h_{i,j} &= \begin{cases} 
   1, & \text{if } p_i(a_j) > p_{i,\text{mhn}} \\
   0, & \text{otherwise} \end{cases}
\end{align*}
\]

(14)

In addition, for each element $a \in X$ we define three binary variables to represent the post-filter classification:

\[
\begin{align*}
   g_j &= \begin{cases} 
   1, & \text{if } a_j \text{ is classified as “green”} \\
   0, & \text{otherwise} \end{cases} \\
   y_j &= \begin{cases} 
   1, & \text{if } a_j \text{ is classified as “yellow”} \\
   0, & \text{otherwise} \end{cases} \\
   r_j &= \begin{cases} 
   1, & \text{if } a_j \text{ is classified as “red”} \\
   0, & \text{otherwise} \end{cases}
\end{align*}
\]

(15)

In this BLP model it is assumed, that a filter classifies an element of the set $X$ as “red”, if at least one of its properties has a “high” value and at least one other property has a value of “medium” or “high”. Furthermore, “green” follows, if none of the properties has a “high” value and at least one property has a “low” value. Hence, definition (15) can be reformulated by using the variables $l$’s, $m$’s and $h$’s:
\[
g_j = \begin{cases} 
1, & \text{if } \sum_{i=1}^{D} h_{i,j} = 0 \text{ AND } \sum_{i=1}^{D} l_{i,j} \geq 1 \\
0, & \text{otherwise}
\end{cases} 
\]

\[
y_j = \begin{cases} 
1, & \text{if } g_j = 0 \text{ AND } r_j = 0 \\
0, & \text{otherwise}
\end{cases} 
\]

\[
r_j = \begin{cases} 
1, & \text{if } \sum_{i=1}^{D} h_{i,j} \geq 2 \text{ OR } \\
\sum_{i=1}^{D} m_{i,j} \geq 1 \text{ AND } \sum_{i=1}^{D} m_{i,j} \geq 1 \\
0, & \text{otherwise}
\end{cases} 
\]

\[a_j \in X \quad \forall j \in \{1, \ldots, N\}.\]

Table 1 summarises the variables we have defined for the BLP.

<table>
<thead>
<tr>
<th>Type of variable</th>
<th># of variables</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_{i,ln}(k))</td>
<td>(D \cdot (N + 1))</td>
<td>low-med boundary for property (i)</td>
</tr>
<tr>
<td>(x_{i,mh}(k))</td>
<td>(D \cdot (N + 1))</td>
<td>med-high boundary for property (i)</td>
</tr>
<tr>
<td>(l_{i,j}, m_{i,j}, h_{i,j})</td>
<td>(3 \cdot D \cdot N)</td>
<td>class. of element (j) according to prop. (i)</td>
</tr>
<tr>
<td>(g_j, y_j, r_j)</td>
<td>(3 \cdot N)</td>
<td>post-filter classification of element (j)</td>
</tr>
</tbody>
</table>

\[5DN + 2D + 3N\] total

4.1.3 Constraints

We first introduce constants that are being used to formulate the constraints for the BLP. Let \(p_{i,j} \in \{1, 2, \ldots, N\}\) denote the ordinal number of \(p_i(a_j), a_j \in X\) in the increasing order of \(p_i\)'s in dimension \(i\). Let \(s\) be a vector of length \((N + 1)\) and let the elements of \(s\) be:

\[
s = \left( \begin{array}{cccc}
\frac{1}{2} & \frac{3}{2} & \ldots & \frac{2N+1}{2}
\end{array} \right).
\]

Hence, the scalar multiplication of, e.g., \(s^T x_{i,ln}\) gives the average of \(p_{i,j}\) and \(p_{i,j+1}\), between which the low-medium boundary of dimension \(i\) lies. For the BLP, we now formulate the constraints (1) as:

\[
s^T x_{i,ln} \leq s^T x_{i,mh}, \quad \forall i \in \{1, \ldots, D\}.\]  \hspace{2cm} (17)
The variables $l$'s, $m$'s and $h$'s are defined through the constraints (18)-(22).

$$l_{i,j} + m_{i,j} + h_{i,j} = 1 \quad (18)$$

$$l_{i,j} \leq \frac{s^Tx_{i,lm}}{p_{i,j}} \quad (19)$$

$$l_{i,j} \geq \frac{s^Tx_{i,lm} - p_{i,j}}{N} \quad (20)$$

$$h_{i,j} \leq 1 + \frac{p_{i,j} - s^Tx_{i,mh}}{N} \quad (21)$$

$$h_{i,j} \geq 1 - \frac{s^Tx_{i,mh}}{p_{i,j}} \quad (22)$$

$l_{i,j}, m_{i,j}, h_{i,j} \in \{0, 1\}$ $\forall i \in \{1, \ldots, D\}, j \in \{1, \ldots, N\}$.

Constraint (18) follows directly from definition (14). Constraint (19) takes care that $l_{i,j}$ goes to zero if $p_{i,j} > s^Tx_{i,lm}$, whereas (20) makes $l_{i,j}$ go to one if $p_{i,j} < s^Tx_{i,lm}$. Respectively, constraints (21) and (22) make $h_{i,j}$ go to zero if $p_{i,j} < s^Tx_{i,mh}$ and to one if $p_{i,j} > s^Tx_{i,mh}$. Obviously, by (18), $m_{i,j} = 0$ if $l_{i,j} = 1$ or $h_{i,j} = 1$, and $m_{i,j} = 1$ if $l_{i,j} = 0$ and $h_{i,j} = 0$.

The following constraints define variables $g$'s, $y$'s and $r$'s according to definition in (16).

$$g_j \leq \sum_{i=1}^{D} l_{i,j} \quad (23)$$

$$g_j \leq 1 - \frac{\sum_{i=1}^{D} h_{i,j}}{D} \quad (24)$$

$$g_j \geq \frac{\sum_{i=1}^{D} l_{i,j}}{D} - \sum_{i=1}^{D} h_{i,j} \quad (25)$$

Constraints (23) and (24) make $g_j$ go to zero if none of the properties has a "low" value or if any of the properties has a "high" value. Constraint (25) makes $g_j$ go to one if at least one of the properties has a "low" value and none of the properties has a "high" value. Respectively, the constraints (26)-(29) for $r$'s are being built to implement the conditions represented in (16):

$$r_j \leq \sum_{i=1}^{D} h_{i,j} \quad (26)$$

$$r_j \leq \frac{\sum_{i=1}^{D} h_{i,j} + \sum_{i=1}^{D} m_{i,j}}{2} \quad (27)$$

13
\[ r_j \geq \frac{\sum_{i=1}^{D} h_{i,j} - 1}{D} \]  
\[ r_j \geq \frac{\sum_{i=1}^{D} m_{i,j}}{D} - (1 - h_{i,j}), \quad \forall i = 1, \ldots, D \]  

Constraints (26) and (27) force \( r_j \) to zero, if none of the properties has a "high" value, or if no two of the properties have at least "medium" value. By (28), \( r_j \) is made to go to one, if two or more properties have a "high" value. For each dimension (property), there is one constraint of type (29), which makes sure, that \( r_j \) is one, if the property \( i \) has a "high" value and at least one other property has a "medium" value. Finally, the correct value of \( y_j \) is taken care of by constraint (30):

\[
g_j + y_j + r_j = 1 \\
g_j, \ y_j, \ r_j \in \{0, 1\}
\]

With the constraints (17)-(30) we have defined the feasible solution space for the BLP model. In Table 2 is calculated the total number of constraints.

<table>
<thead>
<tr>
<th>Constraint</th>
<th># of constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>(17)</td>
<td>( D )</td>
</tr>
<tr>
<td>(18)-(22)</td>
<td>( 5 \cdot D \cdot N )</td>
</tr>
<tr>
<td>(23)-(25)</td>
<td>( 3 \cdot N )</td>
</tr>
<tr>
<td>(26)-(29)</td>
<td>( 3 \cdot N + D \cdot N )</td>
</tr>
<tr>
<td>(30)</td>
<td>( N )</td>
</tr>
<tr>
<td>Total</td>
<td>( 6DN + D + 7N )</td>
</tr>
</tbody>
</table>

### 4.1.4 Objective Function

With the variables introduced in Section 4.1.2, we can now formulate the objective function as follows:

\[
\max \quad z = c_a \sum_{j \in G} g_j - c_b \sum_{j \in R} g_j - c_c \sum_{j \in G} r_j + c_d \sum_{j \in R} r_j,
\]

where \( c_a, c_b, c_c \) and \( c_d \) are the coefficients introduced in Section 3.1, and \( G \) and \( R \) are the sets of "green" and "red" reference elements of \( X \), respectively.
4.2 The Cutting Algorithm

To specifically solve the filter calibration problem for the set $X$ and posets $P$ and $H$, we introduce an algorithm called the Cutting Algorithm, which also uses the assumption, that the boundaries lie between two values of properties. With this assumption, there is a finite set of possible solutions, in fact there are

$$
\binom{N+1}{j}^D = \left( N \frac{1 + N + 1}{2} \right)^D = \left( \frac{N^2 + 2N}{2} \right)^D
$$

possible solutions for the filter calibration problem. This is due to the fact that for each dimension $i$ there are $N + 1$ intervals, where the $p_{i,lm}$ and $p_{i,mh}$ can lie. With the condition that $p_{i,lm} \leq p_{i,mh}$, there are $\sum_{j=1}^{N+1} j$ possibilities for each each dimension, thus the total number of possibilities is that in (32).

The idea of the Cutting Algorithm is to search through the solutions, but at the same time cut away solutions so, that the optimum can be found from a smaller number of possibilities and not all the solutions have to be investigated.

In the description of the algorithm, by $x_{i,lm} = k$ (resp. $x_{i,mh}$) we denote, that the boundary $p_{i,lm}$ lies in the $k$th interval (see Section 3.2). The numbers $n_a$, $n_b$, $n_c$, $n_d$ and the objective function value $z$ are the same as earlier (see Section 3.1). Comments in the algorithm are separated by `'(s' and ')''.

The algorithm consists of initial part, where the variables are initialised, and main part, where the optimal solution(s) are found. The variables are initialised so, that

$$
\forall i \in \{1, \ldots, D\} \quad x_{i,lm}^0 = 1 \text{ and } x_{i,mh}^0 = N + 1.
$$

With this calibration all the elements of $X$ have “medium” value for each property, hence having the filter classification of “yellow”. In the main part, the feasible values of the variables are systematically searched through, in order to find the solutions that give the optimal objective function value $z^*$. The execution of the algorithm is as follows:

**Initial Part**

**Step 1** In each dimension $i = 1, \ldots, D$, sort the elements of $X = G \cup R$ in increasing order by the values of the properties $p_i(a)$, $a_j \in X \forall j = 1, \ldots, N$. 
Step 2 (* Fix initial solution: *)
For $i = 1, \ldots, D$ Do
\[
x^0_{i,lm} = 1, \quad x^0_{i,mh} = N + 1
\]
End For
(* Initially all elements are in the yellow region: *)
\[
n_a^0 = 0, \quad n_b^0 = 0, \quad n_c^0 = 0, \quad n_d^0 = 0
\]
Step 3 (* The best solution so far is marked by '***' *)
\[
n_a^* = n_a^0, \quad n_b^* = n_b^0, \quad n_c^* = n_c^0, \quad n_d^* = n_d^0\]
\[z^* = 0\]
For $i = 1, \ldots, D$ Do
\[
x^*_i,lm = x^0_i,lm, \quad x^*_i,mh = x^0_i,mh
\]
End For
(* $S^*$ is the set of best solutions so far. * 
Initially $S^*$ contains one element: * )
\[
S^* = \{(x^*_{i,lm}, \ldots, x^*_{D,lm}, x^*_{i,mh}, \ldots, x^*_{D,mh})\}
\]
Step 4 (* Initialisation of variables: *)
For $i = 1, \ldots, D$ Do
\[
x_i,lm = x^0_i,lm, \quad x_i,mh = x^0_i,mh
\]
End For

Main Part

Step 1.1
For $x_{1,lm} = x^0_{1,lm}, \ldots, x_{1,mh}$ Do
Calculate $n_a, \quad n_b, \quad n_c, \quad n_d$
Calculate $z$
If $n_c = 0$ & $n_c^* = 0$ & $n_d < n_d^*$ Then
(* Optimal solution cannot be found here *)
Break
Else If $z > z^*$ Then
(* A new best solution, update variables: *)
\[
z^* = z
\]
\[
n_a^* = n_a, \quad n_b^* = n_b, \quad n_c^* = n_c, \quad n_d^* = n_d
\]
For $i = 1, \ldots, D$ Do
\[
x^*_i,lm = x_i,lm, \quad x^*_i,mh = x_i,mh
\]
End For
(* Now we have a unique best solution: *)
\[
S^* = \{(x^*_{1,lm}, \ldots, x^*_{D,lm}, x^*_{1,mh}, \ldots, x^*_{D,mh})\}
\]
Else If $z = z^*$ Then
(* A solution as good as the best solution so far: *)
Add $(x_{1,lm}, \ldots, x_{D,lm}, x_{1,mh}, \ldots, x_{D,mh})$ to $S^*$.
End If
End For
Goto Step 1.2

**Step 1.2**
For $x_{1,mh} = x_{1,mh}^0, \ldots, x_{1,lm}^0$ Do
    Calculate $n_a, n_b, n_c, n_d$
    If $n_c > 0$ Then
        (* Optimal solution cannot be found here *)
        Break
    End If
End For
Goto Step 1.1

**Step 2.1**
For $x_{2,lm} = x_{2,lm}^0, \ldots, x_{2,mh}^0$ Do
    Goto Step 1.2
End For
Goto Step 2.2

**Step 2.2**
For $x_{2,mh} = x_{2,mh}^0, \ldots, x_{2,lm}^0$ Do
    Goto Step 2.1
End For
Goto Step 3.1

;

**Step D.1**
For $x_{D,lm} = x_{D,lm}^0, \ldots, x_{D,mh}^0$ Do
    Goto Step $(D - 1).2$
End For
Goto Step $D.2$

**Step D.2**
For $x_{D,mh} = x_{D,mh}^0, \ldots, x_{D,lm}^0$ Do
    Goto Step $D.1$
End For
Finished: $S^*$ contains all the optimal solutions.

The time-complexity of the algorithm in $\mathcal{O}$-notation becomes $\mathcal{O}(N^{2D})$, following directly from the number of possible solutions (32). It is noteworthy, that the calculation of $n_a$, $n_b$, $n_c$ and $n_d$ is a constant-time operation. This is due to the fact, that every time the calculation is performed, the filter classification of only one element of $X$ has possibly changed from a previously
known situation. Hence, only the filter classification of this one element has to be reassured.

4.3 Improved Cutting Algorithm for Two-dimensional Filters

In the case of a two-dimensional filter, the Cutting Algorithm can be improved to become more efficient. This is, because the pairs \((p_{1,lm}, p_{2,mh})\) and \((p_{1,mh}, p_{2,lm})\), independently of each other, define the regions \((2,3) \cup (3,3), (1,1) \cup (1,2)\) and \((3,2) \cup (3,3), (1,1) \cup (2,1)\), respectively (see Figure 1). Hence, the pairs \((p_{1,lm}, p_{2,mh})\) and \((p_{1,mh}, p_{2,lm})\) can be optimised separately, reducing significantly the number of solutions, that the algorithm has to search through.

In the algorithm, first the pair \((p_{1,lm}, p_{2,mh})\) is optimised keeping \(p_{1,lm}\) and \(p_{2,mh}\) in their initial positions (Step 1). Then, for each pair \((p_{1,mh}, p_{2,lm})\), which has given the best result in Step 1, the pair \((p_{1,lm}, p_{2,mh})\) is optimised (Step 2). However, the optimisation of the pairs has to be performed also in the opposite order, i.e. first the pair \((p_{1,lm}, p_{2,mh})\) is optimised in Step 3, and then, in Step 4, the pair \((p_{1,mh}, p_{2,lm})\) is optimised for each best result of the Step 3. In Step 5 the results of the two optimisation sequences (\{Step 1, Step 2\} and \{Step 3, Step 4\}) are compared, and the lesser solutions are omitted. The algorithm is described in the following:

**Step 0** Same as Initial Part in the Cutting Algorithm in Section 4.2.

**Step 1** \([x_{1,lm} = x_{1,lm}^0, x_{2,mh} = x_{2,mh}^0]\\
For x_{2,lm} = x_{2,lm}^0, \ldots, x_{2,mh}^0 Do\\
For x_{1,mh} = x_{1,mh}^0, \ldots, x_{1,lm}^0 Do\\
  Calculate n_a, n_b, n_c, n_d\\
  Calculate z\\
  If n_c < 0 Then\\
    Break\\
  Else If \(z > z_1^*\) Then\\
    \(z_1^* = z\)\\
    \(x_{1,mh}^* = x_{1,mh}, x_{2,lm}^* = x_{2,lm}\)\\
    \(S_1^* = \{(x_{1,mh}^*, x_{2,lm}^*)\}\)\\
    \(S_{21} = \{(x_{1,lm}, x_{1,mh}, x_{2,lm}, x_{2,mh})\}\)\\
  Else If \(z = z_1^*\) Then\\
    Add \((x_{1,mh}, x_{2,lm})\) to \(S_1^*\)\\
    Add \((x_{1,lm}, x_{1,mh}, x_{2,lm}, x_{2,mh})\) to \(S_{21}\)\\
  End If\\
End For\\
End For
Step 2 [Perform this step for each \((x_{1, mh}^*, x_{2, lm}^*) \in S^*_1\)]

\[
x_{1, mh} = x_{1, mh}^*, \ x_{2, lm} = x_{2, lm}^*
\]

For \(x_{2, mh} = x_{2, mh}^0, \ldots, x_{2, lm}^0\) Do

For \(x_{1, lm} = x_{1, lm}^0, \ldots, x_{1, mh}^0\) Do

Calculate \(n_a, n_b, n_c, n_d\)

Calculate \(z\)

If \(n_c = 0 \& n_c^* = 0 \& n_d < n_d^*\) Then

Break

Else If \(z > z_2^*\) Then

\[
z_2^* = z
\]

\[
x_{1, lm}^* = x_{1, lm}, \ x_{2, mh}^* = x_{2, mh}
\]

\[
S_2^* = \{(x_{1, lm}^*, x_{2, mh}^*)\}
\]

\[
[S_{12}^* = \{(x_{1, lm}^*, x_{1, mh}^*, x_{2, lm}^*, x_{2, mh}^*)\}]
\]

Else If \(z = z_3^*\) Then

Add \((x_{1, lm}, x_{2, mh})\) to \(S_2^*\)

[Add \((x_{1, lm}, x_{1, mh}, x_{2, lm}, x_{2, mh})\) to \(S_{12}^*\)]

End If

End For

End For

Step 3 [Perform Step 2, excluding everything in square brackets [...] , and let \(x_{1, mh} = x_{1, mh}^0, \ x_{2, lm} = x_{2, lm}^0\)]

Step 4 [Set \(S_{21}^* = \emptyset\). Perform Step 1 for each \((x_{1, lm}^*, x_{2, mh}^*) \in S_2^*\), excluding all [...]’s]

Step 5

If \(z(S_{12}^*) = z(S_{21}^*)\) Then

\(S_{12}^*\) and \(S_{21}^*\) contain all the optimal solutions

Else If \(z(S_{12}^*) > z(S_{21}^*)\) Then

\(S_{12}^*\) contains all the optimal solutions

Else If \(z(S_{21}^*) > z(S_{12}^*)\) Then

\(S_{21}^*\) contains all the optimal solutions

End If

The worst-case time-complexity of this algorithm is \(O(N^3)\). This is due to the two nested For-loops in Steps 1 and 2, and the fact, that in Steps 2 and 4 these For-loops are performed for each solution in sets \(S_1^*\) and \(S_2^*\). In the worst case the cardinality of sets \(S_1^*\) and \(S_2^*\) is proportional to \(N\), thus giving the time-complexity of \(O(N^3)\). However, in practice there are rarely more than a few elements in sets \(S_1^*\) and \(S_2^*\), thus, in reality, the time-complexity can be considered to be \(O(N^2)\).
5 Discussion

The two solution introduced—the BLP model and the algorithmic approach—are somewhat of different nature. After building the BLP model for a specific filter calibration problem, an algorithm is still needed to find an optimal solution to the BLP problem. Normally, to solve a general discrete optimisation problem, an algorithm such as, e.g., the Branch and Bound Algorithm or the Gomory Cutting-Plane Algorithm is used (Taha 1997). Specifically for a BLP problem, in his book Taha (1997) introduces a variation of the Branch and Bound Algorithm, called the Additive Algorithm, which for a BLP problem performs more efficiently than the Branch and Bound or the Gomory Cutting-Plane algorithms.

The size of a BLP is normally given by its number of variables and its number of linear constraints (Bazaraa et al. 1990). As presented in Tables 1 and 2, the BLP model for filter calibration has $5DN + 2D + 3N$ variables and $6DN + D + 7N$ linear constraints, where $D$ is the dimension of the filter and $N$ is the number of elements used as reference data. A big advantage for the BLP model is, that the amount of variables and constraints is linearly dependent on the size of the input, i.e. $D$ and $N$). Also, of the variables only the $x_{i,m}$’s and $x_{i,mh}$’s are truly variables. All the other variables are exactly defined by these. Hence, the number of real variables is reduced to $2D(N + 1)$.

When solving a BLP problem, an algorithm normally finds one optimal solution. However, in some cases the optimum for the filter calibration problem is not unique, but there exists alternative optimal solutions. Finding all of the optimal solutions for the filter calibration problem apparently involves more calculation for the algorithm.

The algorithms presented in Sections 4.2 and 4.3, after finishing execution, have found all the optimal solutions to the filter calibration problem. The time-complexity of the Cutting Algorithm stays $O(N^{2D})$ independent of the number of optimal solutions, whereas for the Improved Cutting Algorithm the time-complexity would be $O(N^{2M})$, where $M$ is the amount of optimal solutions. Obviously, $M$ is linearly dependent on $N$, which makes the worst-case time-complexity of the Improved Cutting Algorithm $O(N^3)$. In a unique optimal solution case, or if it is sufficient to find only one optimal solution, the time-complexity of the Improved Cutting Algorithm is reduced to $O(N^2)$. Both algorithms are polynomial in the number $N$. The dimension of the filter, however, gives an exponential dependency if $D > 2$.

Both the BLP model and the algorithms give intervals as the optimal solution for $p_{i,m}$’s and $p_{i,mh}$’s (see Section 3.2). Several methods can be used to define exact values for $p_{i,m}$’s and $p_{i,mh}$’s inside a given optimal interval. Firstly, the average of the end-points of the interval can be taken. Secondly,
a so called median rule can be used. Thirdly, the \( p_{i,m} \) ’s can be set to the infimum of the interval, and the \( p_{i,mh} \) ’s to the supremum of the interval. These three methods clearly result in different calibrations with the properties characteristic of the method. Hence, in a case with alternative optimal solutions, first has to be decided which given solution (i.e. intervals) to take, and then the exact points of \( p_{i,m} \) ’s and \( p_{i,mh} \) ’s in the interval. Making the decision of the solution in the case of alternative optima, is left beyond the scope of this paper.

6 Summary and Conclusions

To carry out the calibration of a precautionary filter, we have come out with two different approaches. Firstly, a binary linear program (BLP) model is constructed. Secondly, an algorithm—the Cutting Algorithm—that solves the general filter calibration problem is introduced. For the calibration of a two-dimensional filter, the Cutting Algorithm is slightly modified, resulting in better efficiency.

The BLP model can be solved by known algorithms (see, e.g., Taha 1997), however, the size of the model can become a problem. In the BLP model, there are \( 5DN + 2D + 3N \) variables, where \( D \) is the dimension of the filter in question and \( N \) is the amount of reference data. Although the number of real variables is less, in fact \( 2D(N + 1) \), in the calibration of, e.g., a two-dimensional filter with 80 elements of reference data, there are still

\[
2 \cdot 2(80 + 1) = 324
\]

variables in the model. The number of constraints is \( 6DN + D + 7N \), leading to

\[
6 \cdot 2 \cdot 80 + 2 + 7 \cdot 80 = 1522
\]

constraints in our example. Obviously, the big amount of constraints reduces the number of possible solutions, making it easier for an algorithm to find the optimum.

Most likely, a more practical way to find the optimal filter calibration is a direct algorithmic approach. The Cutting Algorithm introduced in this paper, finds the optimal solution(s) in the number of calculations proportional to \( O(N^{2D}) \). The Cutting Algorithm can be characterised as a polynomial algorithm, since in practical applications, the dimension of the filter usually stays as low as \( D = 2 \) or \( D = 3 \). With \( D = 2 \), the algorithm can be improved to perform in \( O(N^3) \), as a worst-case performance. In practise, when the number of alternative optima is relatively small, the performance with \( D = 2 \) has normally been found to be \( O(N^2) \).
In general, more discussion about the alternative optima is needed. The different optimal solutions can be very far from each other, therefore finding some grounds for the decision of which solution ought to be put into practise is essential. If a big number of alternative optima occurs, the solutions achieved by the algorithms should be taken as a basis for further discussion on the final calibration. Also, in that a solution for the filter calibration problem—with the present optimality criteria—is intervals for the variables, the exact value in the interval should be determined by some means. In this paper, we have presented three methods for determining the final value for the boundaries:

1. average of the end-points of the interval
2. the so called median rule
3. taking the end-points of the interval.

Summarising the achievements of our work, we have presented a polynomial algorithm to solve the filter calibration problem, as well as a binary linear program to model the filter calibration problem. For further research, an important topic is to find a method by which the final solution, i.e. the exact values of the boundaries, can be fixed. It is natural, that this might lead to additional optimality criteria, in order to have a unique solution for the problem. Another topic would be improving the performance of the algorithm in cases of $D > 2$. 

22
References


