Multi-currency Influence Diagrams

Søren Holbech Nielsen
holbech@cs.aau.dk

Thomas D. Nielsen
tdn@cs.aau.dk

Finn Verner Jensen
fvj@cs.aau.dk

Abstract

When using the influence diagrams framework for solving a decision problem with several different quantitative utilities, the traditional approach has been to convert the utilities into one common currency. This conversion is carried out using a tacit transformation, under the assumption that the converted problem is equivalent to the original one. In this paper we present an extension of the influence diagram framework, which allows for these decision problems to be modelled in their original form. We present an algorithm that, given a conversion function between the currencies of the original utilities, discovers a characterisation of all other such functions, which induce the same optimal strategy. As this characterisation can potentially be very complex, we give methods to present it in an approximate way.

1 Introduction

Influence diagrams (IDs) were introduced by Howard and Matheson (1984) as a compact modelling language for decision problems with a single decision maker (DM). When a decision problem is represented using the ID framework, the specification rests on two principal components: A graphical structure for capturing the qualitative part of the domain, and quantitative information in the form of probabilities for representing uncertainty and utilities for representing preferences.

The separation of the qualitative and quantitative part of the ID is one of the appealing properties of IDs when considered as a modelling tool. First, it helps the modeller to focus on structure rather than calculations, and second, the structure emphasises the local relations, which govern the specification of the probabilities. Unfortunately, this separation property does not completely extend to the specification of the utility function: The utility function is usually specified through a collection of local utility functions, which appear as the additive components of the global utility function. This implies that all local utility functions should appear on the same scale. For instance, in a medical domain money and discomfort would need to be transformed onto a common scale. Unfortunately, it is usually difficult to elicit the parameters, which governs such a transformation (e.g. what is the monetary cost of one unit of discomfort?).[1] Moreover, the nature of this transformation has a direct impact on the solution of the ID, but the effect cannot be made transparent when the transformation is tacit, and this type of uncertainty, or ignorance, is not easily represented in the model.

In this paper we propose a framework, termed Multi-currency IDs (MCIDs), for representing decision problems with local utility functions of different currencies.[2] An MCID can be seen as an ID augmented with currency information for the local utility functions. We propose an algorithm that, based on an MCID representation of a decision problem and a solution corresponding to a given set of currency transformation parameters, provides a characterisation of all combinations of parameters, which would give rise to the same solution — a solution being a strategy composed of a policy for each decision. In addition to this algorithm we provide methods for presenting the result to a DM in a comprehensible manner.

A Motivating Example: *The ID in Figure 1 models a decision problem where a doctor is faced with a patient. The health,* $Health_1$*, of the patient is revealed only indirectly by the* $Symptoms$ *which the*

[1] (Skaaning, 2000) considers utilities that are defined as a linear combination of cost, insult and risk, for instance.

[2] Even though the word currencies is used here, we are not restricting ourselves to monetary currencies, but consider human lives, spare time etc. as currencies also.

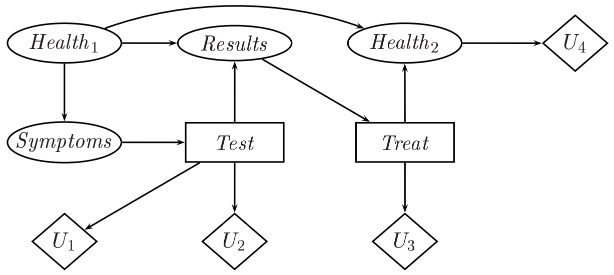

Figure 1: *Example of an ID.*

patient exhibits. Based on the symptoms, the doctor must decide whether to perform a $Test$. The test will produce a $Result$, which the doctor observes before deciding on a treatment (if any), $Treat$. The health of the patient, after a possible treatment has been administered, is represented by the variable $Health_2$. As medical supplies are expensive both the $Test$ and $Treat$ decisions are associated with a monetary cost represented by the utility nodes U_2 and U_3. Furthermore, if a test is performed, it might be associated with a degree of pain, risk of mortality, or other side effects on behalf of the patient. This is represented by the utility node U_1. The $Health_2$ variable also has a preferred state (corresponding to the patient being well), which is encoded by the utility node U_4.

Now, before the doctor calculates an optimal strategy for his decision problem, he needs to transform all utilities, $U_1, \ldots, U_4$, onto a common scale — say, dollars. This involves estimating the monetary equivalents of the patient being ill and of him being subjected to a painful test. However, different transformations might produce differing optimal strategies. Therefore it would be advantageous to know:

1. *What choice of conversion parameters has been the basis of the calculated optimal strategy?*
2. *What other conversion parameters would produce the same optimal strategy? If another stakeholder, such as the patient in this example, disagrees with the parameters, we could guarantee that even though there is disagreement on the exact choice of parameters, the identified set of parameters all render the same strategy optimal.*

★

2 Influence Diagrams

An ID is a directed acyclic graph consisting of *chance* ($\boldsymbol{V}_C$), *decision* ($\boldsymbol{V}_D$), and *utility nodes* ($\boldsymbol{V}_U$), with the constraints that a node has no children if and only if (iff) it is a utility node, and that there is a directed path encompassing all decision nodes in the diagram. A chance node (drawn as an ellipse) represents a discrete variable outside the DM's direct control, which we will refer to as a chance variable. A decision node (drawn as a rectangle) represents a discrete variable under the DM's direct control, which we will call a decision variable or simply a decision. A utility node (drawn as a diamond) represents a local utility function, and the set of local utility functions constitute the components in an additive factorisation of the global utility function (Tatman and Shachter, 1990). When the meaning is obvious from the context, we will use the terms node and variable interchangeably and will not distinguish between a variable or local utility function and the node representing it.

When speaking of IDs we denote by $\mathbf{pa}(V)$ the set of nodes which are parents of the node V, and by $\mathbf{sp}(V)$ we denote the set of *states* of variable V — states being outcomes for chance variables and decision options for decisions. For a set of variables, $\boldsymbol{S}$, we denote by $\mathbf{sp}(\boldsymbol{S})$ the configurations $\times_{V \in \boldsymbol{S}} \mathbf{sp}(V)$.

An arc in an ID represents either functional dependence, probabilistic dependence, or temporal precedence, depending on the type of node it goes into. In particular an arc emanating from a node, X, going into a decision node, D, is a temporal precedence arc and states that X is observed or decided upon before D is decided upon. A chance variable which is not a parent of any decision in the ID is either never observed or observed after the last decision. The temporal precedence arcs impose a partial temporal ordering, $\prec$, on $\boldsymbol{V}_C \cup \boldsymbol{V}_D$. Together with the requirement on a directed path through all decisions, this ordering induces a total temporal ordering on $\boldsymbol{V}_D$. For notational convenience, we will assume that the decisions are labelled $D_1, \ldots, D_n$, such that $i < j$ implies $D_i \prec D_j$. Furthermore, we will use the notation $\boldsymbol{C}_{i-1}$ to mean the set of chance variables observed immediately before deciding on D_i. By $\boldsymbol{C}_n$ we refer to the set of chance variables never observed (or observed after the last decision, D_n, has been decided upon). In summary we have

$$\boldsymbol{C}_0 \prec D_1 \prec \boldsymbol{C}_1 \prec \cdots \prec D_n \prec \boldsymbol{C}_n.$$

We define the *past* of decision D to be $\mathbf{past}(D) = \{V \in \boldsymbol{V}_C \cup \boldsymbol{V}_D \mid V \prec D\}$. We encode the quantitative aspects of the modelled decision problem as a set, $\boldsymbol{\Phi}$, of conditional probability distributions and a set, $\boldsymbol{\Psi}$, of local utility functions:

$$\boldsymbol{\Phi} = \{P(C|\mathbf{pa}(C)) \mid C \in \boldsymbol{V}_C\}, \text{ and}$$
$$\boldsymbol{\Psi} = \{U(\mathbf{pa}(U)) \mid U \in \boldsymbol{V}_U\}.$$

A pair, $(\boldsymbol{\Phi}, \boldsymbol{\Psi})$, is called a *realisation* for the ID. Here, and henceforth, we have used $f(V_1, \ldots, V_k)$ to denote a function of the type $f : \mathbf{sp}(V_1) \times \cdots \times \mathbf{sp}(V_k) \to \mathbb{R}$. Such a function is called a *potential*; we shall distinguish between *probability potentials*, denoted by ϕ's, and *utility potentials*, denoted by ψ's. Furthermore, the set of variables $\{V_1, \ldots, V_k\}$ will be referred to as the *domain* of f, denoted $\mathrm{dom}(f)$.

Given an ID and a decision, D, a function, $\delta_D : \mathbf{sp}(\mathbf{past}(D)) \to \mathbf{sp}(D)$, is called a *policy* for D. A collection of policies for each decision in an ID,

$$\boldsymbol{\Delta} = \{\delta_D : \mathbf{sp}(\mathbf{past}(D)) \to \mathbf{sp}(D) \mid D \in \boldsymbol{V}_D\},$$

is called a *strategy* for the ID. Given a policy, δ_D, for a decision, D, we define the *chance variable policy* (Cooper, 1988) for D, $P_{\delta_D}(D|\mathbf{past}(D))$, as $P_{\delta_D}(d|\boldsymbol{c}) = 1$ if $\delta_D(\boldsymbol{c}) = d$ and 0 otherwise. An *optimal strategy*, $\boldsymbol{\Delta}^*$, for an ID is a strategy that fulfills

$$\boldsymbol{\Delta}^* = \arg\max_{\boldsymbol{\Delta}} \sum_{\boldsymbol{V}_C \cup \boldsymbol{V}_D} \Big(\prod_{D \in \boldsymbol{V}_D} P_{\delta_D} \prod_{\phi \in \boldsymbol{\Phi}} \phi \sum_{\psi \in \boldsymbol{\Psi}} \psi \Big). \tag{1}$$

The individual policies in an optimal strategy are referred to as *optimal policies*. To denote that a policy for a decision, D, is a part of an optimal strategy, we write it as δ_D^*. The quantity that is maximised in Equation (1) is the *expected utility* of the decision problem given the strategy $\boldsymbol{\Delta}$, and it is denoted $\mathrm{eu}(\boldsymbol{\Delta})$.

When optimal policies are to be identified it is usually easier to work with a recursive expression for the maximum expected utility instead of Equation (1). Here $\boldsymbol{\Phi}_{(j)}$ denotes the set of potentials in $\boldsymbol{\Phi}$ which have a variable in $\boldsymbol{C}_j$ in their domain, and $\boldsymbol{\Psi}_{(j)}$ is the potentials in $\boldsymbol{\Psi}$ having a variable in $\boldsymbol{C}_j \cup \{D_j\}$ (or just $\boldsymbol{C}_0$ if $j = 0$) in their domain.:

$$\delta_{D_n}^* = \arg\max_{\delta_{D_n}} \sum_{\boldsymbol{C}_n} P_{\delta_{D_n}} \prod_{\phi \in \boldsymbol{\Phi}_{(n)}} \phi \sum_{\psi \in \boldsymbol{\Psi}_{(n)}} \psi, \tag{2}$$

and

$$\rho_{D_n} = \max_{\delta_{D_n}} \frac{\sum_{\boldsymbol{C}_n} P_{\delta_{D_n}} \prod_{\phi \in \boldsymbol{\Phi}_{(n)}} \phi \sum_{\psi \in \boldsymbol{\Psi}_{(n)}} \psi}{\sum_{\boldsymbol{C}_n} P_{\delta_{D_n}} \prod_{\phi \in \boldsymbol{\Phi}_{(n)}} \phi}, \tag{3}$$

for the last decision, D_n. For all other decisions, D_i, we have

$$\delta_{D_i}^* = \arg\max_{\delta_{D_i}} \sum_{\boldsymbol{C}_i} P_{\delta_{D_i}} \prod_{\phi \in \boldsymbol{\Phi}_{(i)}} \phi \Big(\sum_{\psi \in \boldsymbol{\Psi}_{(i)}} \psi + \rho_{D_{i+1}} \Big), \tag{4}$$

and

$$\rho_{D_i}^* = \max_{\delta_{D_i}} \frac{\displaystyle\sum_{\boldsymbol{C}_i} P_{\delta_{D_i}} \prod_{\phi \in \boldsymbol{\Phi}_{(i)}} \phi \Big(\sum_{\psi \in \boldsymbol{\Psi}_{(i)}} \psi + \rho_{D_{i+1}} \Big)}{\displaystyle\sum_{\boldsymbol{C}_i} P_{\delta_{D_i}} \prod_{\phi \in \boldsymbol{\Phi}_{(i)}} \phi}. \tag{5}$$

We may then write the maximum expected utility of the ID as

$$\mathrm{eu}(\boldsymbol{\Delta}^*) = \sum_{\boldsymbol{C}_0} \prod_{\phi \in \boldsymbol{\Phi}_{(0)}} \phi \Big(\sum_{\psi \in \boldsymbol{\Psi}_{(0)}} \psi + \rho_{D_1} \Big).$$

Not all variables in the past of a decision are necessarily relevant for that decision. Therefore, we call a variable, V, *required* for a decision, D, if there exists a realisation and a configuration, $\boldsymbol{c}$, over the variables in $\mathbf{past}(D) \setminus \{V\}$, such that $\delta_D^*(\boldsymbol{c}, v_i) \neq \delta_D^*(\boldsymbol{c}, v_j)$ for two states, v_i and v_j, in $\mathbf{sp}(V)$. The set of required variables for a decision, D, is denoted by $\mathbf{req}(D)$. We may then redefine a policy to be a function $\delta_D : \mathbf{sp}(\mathbf{req}(D)) \to \mathbf{sp}(D)$.

Given an ID and a realisation, an optimal strategy may be found through the use of any one of a number of algorithms including (Shachter, 1986; Shenoy, 1991; Madsen and Jensen, 1999), which all utilise the distributive and associative law on the expressions in Equations (2) to (5).

3 Multi-currency Influence Diagrams

From the summations of utility potentials in Equations (2) to (5), it is clear that these must be of the same type, i.e. defined over the same currency. We now introduce a framework capable of representing decision problems involving utilities of several currencies. We call models in this framework Multi-currency Influence Diagrams (MCIDs).

Basically, an MCID is just an ID where each of the utility nodes is annotated with the currency of the corresponding local utility function. Formally, all syntax and semantics of IDs described in Section 2 carry over to MCIDs, except for the requirement that the local utility functions must form an additive decomposition of the global utility function. By assuming some arbitrary, but fixed, order of the currencies in the MCID, $s_1, \ldots, s_m$, we may refer to the currency of a local utility, U, by a natural number, $i \in \{1, \ldots, m\}$.

In what follows we will refer to the number of different currencies of an MCID as the *dimension* of the MCID, and throughout assume this to be m. A realisation of an MCID, is therefore a tuple, $(\mathbf{\Phi}, \mathbf{\Psi}_1, \ldots, \mathbf{\Psi}_m)$, where $\mathbf{\Psi}_i$ is the set of local utility functions of currency i. We require that, for each set, $\mathbf{\Psi}_i$, the sum of its elements corresponds to a utility function, which encodes the same preference ordering as if the DM had disregarded all consequences measured in currencies different from i. Note that, it follows that an MCID of dimension 1 is an ID, hence, from this point on, we will assume m to be larger than 1.

A strategy for an MCID is the same as for an ID: A prescription for choices given previous observations and decisions. However, if we want to compute an optimal strategy for an MCID, we need a method for comparing amounts of one currency with amounts of another. This can be seen from the following example.

A Simple Example of an MCID: *Consider a simple two-dimensional MCID, over the currencies A and B, with only a single binary decision, D, and two utilities, U_1 and U_2, defined as*

$$U_1(D = d_1) = 1A, \quad U_1(D = d_2) = 5A$$
$$U_2(D = d_1) = 2B, \quad U_2(D = d_2) = 1B.$$

Both choices of D can be optimal choices depending on how much the DM values amounts of currency A relative to amounts of currency B. If $D = d_1$ should be an optimal strategy then

$$\text{eu}(D = d_1) \geq \text{eu}(D = d_2)$$
$$\Leftrightarrow 1A + 2B \geq 5A + 1B \Leftrightarrow \qquad -4A + B \geq 0,$$

whereas if $-4A + B \leq 0$ then $D = d_2$ is an optimal strategy. If we regard the currency name, A, as a real variable representing the DM's degree of appreciation of amounts of A, and similar for currency name B, then $-4 \cdot A + B$ corresponds to an amount of appreciation equivalent to -4 A's and one B. The set of all values for A and B, where $-4 \cdot A + B \geq 0$, then corresponds to the possible attitudes of the DM which would render d_1 the optimal choice of the decision problem modelled by the MCID. The state space of $A \times B$, viz. $\mathbb{R}^2$, is thus partitioned into two regions, each corresponding to an optimal strategy. ★

To sum up, given a strategy, $\mathbf{\Delta}$, we see that eu$(\mathbf{\Delta})$ is an element of $\mathbb{R}^m \to \mathbb{R}$ rather than a scalar value as is the case with strategies for IDs. The means we use for comparing currencies is called *currency mappings*:

Definition 1 *Let $\mathcal{I}$ be an MCID of dimension m and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)$, a point in $\mathbb{R}^m$, then $\boldsymbol{\alpha}$ is a currency mapping for $\mathcal{I}$.*

The semantics of a currency mapping, $\boldsymbol{\alpha}$, *reflecting a DM's preferences*, is that, for any two amounts, x_i and x_j, of currencies i and j, respectively, we have that $\alpha_i x_i$ equals $\alpha_j x_j$ iff the DM values x_i of currency i as much as x_j of currency j. It follows that we assume the preferential relationship between currencies to be a linear one. The objective of the DM then becomes to maximize the global utility given by $\sum_{i=1}^{m} \alpha_i \sum_{\psi_i \in \mathbf{\Psi}_i} \psi_i$. We will use Latin letters ($\boldsymbol{f}$, $\boldsymbol{q}$ etc.) to denote arbitrary points in $\mathbb{R}^m$, and Greek letters ($\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ etc.) when we want to emphasise that they are currency mappings. In general, we will use q_i to refer to the i'th coordinate of a point $\boldsymbol{q}$. In what follows we will furthermore use $\boldsymbol{f} \cdot \boldsymbol{q}$ to denote the scalar product $\sum_i f_i q_i$.

In this paper we will assume without loss of generality that each element of a currency mapping is positive, i.e. a currency mapping is an element of $\mathbb{R}^{+m}$

rather than $\mathbb{R}^m$, where $\mathbb{R}^+$ denotes the set of strictly positive reals. This assumption implies that everybody should be able to agree on whether amounts of each currency, i, is beneficial to be had or not, and that no-one would contest the relevance of amounts of any currency. If a currency, i, is disadvantageous to be had (meaning that α_i should be negative) we expect the modelled decision problem to have negative utilities specified for positive amounts of i, as is usually done when costs are specified in IDs. All results in the paper, except a method for keeping space requirements down during computation, can easily be extended to scenarios where this assumption does not hold. The notation is more complicated, though.

If for a strategy, $\mathbf{\Delta}$, we have that $\boldsymbol{\alpha}\cdot\mathrm{eu}(\mathbf{\Delta})$ is greater than or equal to $\boldsymbol{\alpha}\cdot\mathrm{eu}(\mathbf{\Delta}')$ for all other strategies $\mathbf{\Delta}'$, we say that $\mathbf{\Delta}$ is optimal given the currency mapping $\boldsymbol{\alpha}$. We will denote an optimal strategy for an MCID given a currency mapping, $\boldsymbol{\alpha}$, as $\mathbf{\Delta}^*_{\boldsymbol{\alpha}}$. As several strategies might give rise to the same expected utility, the set of all optimal strategies corresponding to $\boldsymbol{\alpha}$ is denoted as $\overline{\mathbf{\Delta}^*_{\boldsymbol{\alpha}}}$.

4 Support Analysis of Currency Mappings

Clearly, if we are given a currency mapping, $\boldsymbol{\alpha}$, in addition to an MCID we can solve it by means of simply converting it into an ID through multiplying each local utility function of currency i by α_i. This simple solution method allows for optimal strategies to be computed for any $\boldsymbol{\alpha}$, and the distinction between MCIDs and currency mappings emphasises the assumptions leading to the results. However, we would not be closer to answering the second question in the motivating example. In order to do this, we must render the effect of $\boldsymbol{\alpha}$ on the optimal strategy transparent. We would then be able to reason about the universality of the applicability of that optimal strategy. We can obtain this transparency by postponing the conversion of utilities until it is needed, so that we can analyse which currency mappings give rise to the same optimal strategy. We start by introducing some auxiliary concepts.

4.1 Preliminaries

Given an MCID and a strategy, $\mathbf{\Delta}$, we will call the set

$$\mathbf{su}(\mathbf{\Delta}) = \{\boldsymbol{\beta} \in \mathbb{R}^{+m} \mid \mathbf{\Delta} \in \overline{\mathbf{\Delta}^*_{\boldsymbol{\beta}}}\}$$

the *support* of $\mathbf{\Delta}$. Intuitively, $\mathbf{su}(\mathbf{\Delta})$ is the set of currency mappings for which $\mathbf{\Delta}$ is an optimal strategy. We will refer to the process of calculating the support of an optimal strategy as performing support analysis of the MCID. In the simple example above we actually found the support of both strategies $D = d_1$ and $D = d_2$, which turned out to be the two partitions of $\mathbb{R}^{+2}$ defined by $-4A + B = 0$. Later it will become apparent that any such support can be described as an intersection of partitions of $\mathbb{R}^m$; each partition described by a linear inequality.

As mentioned in the beginning of this section, we will postpone the conversion of utility potentials until needed. Hence, we introduce a new potential which can represent utility functions of several currencies:[3] A *multi-currency utility potential* (MCUP) of dimension m over the variables in a set, $\boldsymbol{S}$, is a function, $\theta : \mathbf{sp}(\boldsymbol{S}) \to \mathbb{R}^m$, attributing to each configuration, $\boldsymbol{c}$, over the variables in $\boldsymbol{S}$ a measure of utility, $(\theta(\boldsymbol{c}))_i$, of each currency, i. For a DM whose preferences are reflected by the currency mapping $\boldsymbol{\beta}$, $\boldsymbol{\beta}\cdot\theta(\boldsymbol{S})$ is a utility potential that for any configuration, $\boldsymbol{c}$, over variables in $\boldsymbol{S}$ yields the value $\beta_1(\theta(\boldsymbol{c}))_1 + \cdots + \beta_m(\theta(\boldsymbol{c}))_m$, which is equivalent to amounts $(\theta(\boldsymbol{c}))_1, \ldots, (\theta(\boldsymbol{c}))_m$ of currencies $1, \ldots, m$, respectively. When adding MCUPs or multiplying probability potentials onto them, we simply treat each dimension of the MCUP as a regular utility potential, and perform the operation on each dimension separately.

4.2 Support Analysis

We are now ready to give a procedure for performing support analysis of an MCID. We assume the existence of an optimal strategy, $\mathbf{\Delta}^*_{\boldsymbol{\alpha}}$, determined by some initial currency mapping, $\boldsymbol{\alpha}$, as well as a realisation, $(\mathbf{\Phi}, \mathbf{\Psi}_1, \ldots, \mathbf{\Psi}_m)$, and we look for the support of $\mathbf{\Delta}^*_{\boldsymbol{\alpha}}$ for this realisation. The method

[3] Such potentials are not part of the MCID framework as such, but rather data structures used by the method for support analysis that we present.

is inspired by that of Lazy evaluation presented in (Madsen and Jensen, 1999) and basically traces the steps of this method while recording the requirements on $\boldsymbol{\alpha}$ for $\boldsymbol{\Delta}^*_{\boldsymbol{\alpha}}$ to be an optimal strategy. The method consists of an initialisation phase and an identification phase. The initialisation phase consists of two steps: First, two empty sets are generated, Ξ and $\boldsymbol{\Theta}$, where Ξ will hold inequalities defining $\mathbf{su}(\boldsymbol{\Delta}^*_{\boldsymbol{\alpha}})$, and $\boldsymbol{\Theta}$ is a container for the MCUPs that are used in the identification phase. Second, for each currency, i, each potential, ψ, in $\boldsymbol{\Psi}_i$ is converted to an MCUP, θ, such that $\theta_k = \psi$ if $k = i$ and 0 otherwise, where 0 denotes the function yielding the zero value for all input. The identification phase follows the Lazy evaluation method, except for the steps normally carried out when a variable is eliminated (see Algorithm 1). The major difference between these steps and the corresponding steps in Lazy evaluation is that we do not perform a maximisation to uncover an optimal strategy. Instead we look for a set, Ξ, of linear inequalities (Step 4) that need to be fulfilled if $\boldsymbol{\Delta}^*_{\boldsymbol{\alpha}}$ is to be optimal. We will refer to the inequalities in Ξ as *constraints*, since they constrain the support set.

Algorithm 1: *The elimination steps of the identification phase. The strategy, $\boldsymbol{\Delta}^*_{\boldsymbol{\alpha}}$, is assumed to be given apriori.*

1. Let V be the variable to be eliminated, and let $\boldsymbol{\Phi}_V$ denote the set of probability potentials in $\boldsymbol{\Phi}$ with V in their domain, and $\boldsymbol{\Theta}_V$ denote the set of MCUPs in $\boldsymbol{\Theta}$ with V in their domain.

2. Let

$$\phi_V = \prod_{\phi \in \boldsymbol{\Phi}_V} \phi, \quad \text{and} \quad \theta_V = \sum_{\theta \in \boldsymbol{\Theta}_V} \theta.$$

3. If V is a chance variable, then set

$$\boldsymbol{\Phi} \leftarrow (\boldsymbol{\Phi} \setminus \boldsymbol{\Phi}_V) \cup \left\{ \sum_V \phi_V \right\},$$

and

$$\boldsymbol{\Theta} \leftarrow (\boldsymbol{\Theta} \setminus \boldsymbol{\Theta}_V) \cup \left\{ \frac{\sum_V (\phi_V \theta_V)}{\sum_V \phi_V} \right\}.$$

4. If V is a decision variable, then set

$$\boldsymbol{\Phi} \leftarrow (\boldsymbol{\Phi} \setminus \boldsymbol{\Phi}_V) \cup \{\phi(V = v) \mid \phi \in \boldsymbol{\Phi}_V\},$$

where v is some arbitrary state of V,[4] and

$$\boldsymbol{\Theta} \leftarrow (\boldsymbol{\Theta} \setminus \boldsymbol{\Theta}_V) \cup \{\theta_V(\delta^*_V)\},$$

where δ^*_V is the policy for V in $\boldsymbol{\Delta}^*_{\boldsymbol{\alpha}}$. For each configuration, $\boldsymbol{c}$, over the variables in $\mathbf{req}(V)$ and each state $v \neq \delta^*_V(\boldsymbol{c})$ of V, set

$$\Xi \leftarrow \Xi \cup \{\boldsymbol{f}_{\boldsymbol{c},v} \cdot \boldsymbol{\gamma} \geq 0\}, \tag{6}$$

where $\boldsymbol{f}_{\boldsymbol{c},v}$ denotes $\theta(\boldsymbol{c}, \delta^*_V(\boldsymbol{c})) - \theta(\boldsymbol{c}, v)$.

Alternatively, the Lazy evaluation algorithm itself can easily be interleaved with the method presented here by inserting the following step prior to Step 4 in Algorithm 1:

* For each configuration, $\boldsymbol{c}$, over the variables in $\mathbf{req}(V)$ set

$$\delta^*_V(\boldsymbol{c}) = \arg\max_{v \in \mathbf{sp}(V)} \boldsymbol{\alpha} \cdot \theta_V(\boldsymbol{c}, v).$$

Although the method, as presented here, follows the structure of the Lazy evaluation method, it can easily be adapted to follow the structure of any other solution method that is based on the expressions in Equations (2) to (5). We conjecture that any such adaptation would identify the support set as long as the constraints are identified and stored.

Proposition 2 *Let $\boldsymbol{\beta}$ be in $\mathbb{R}^{+m}$, $\boldsymbol{\Delta}$ a strategy, and Ξ the result of running the method described above on $\boldsymbol{\Delta}$. Then $\boldsymbol{\beta}$ is an element of $\mathbf{su}(\boldsymbol{\Delta})$ iff $\boldsymbol{\beta}$ satisfies all inequalities in Ξ.*

4.3 Finding a minimal support set

The procedure described above finds the support of a strategy for a given currency mapping, and stores it as a set of constraints, Ξ. The cardinality of Ξ is given by

$$|\Xi| = \sum_{D \in \boldsymbol{V}_D} |\mathbf{sp}(\mathbf{req}(D))|(|\mathbf{sp}(D)| - 1),$$

[4] Note that any potential, ϕ, in $\boldsymbol{\Phi}_V$ is constant over V.

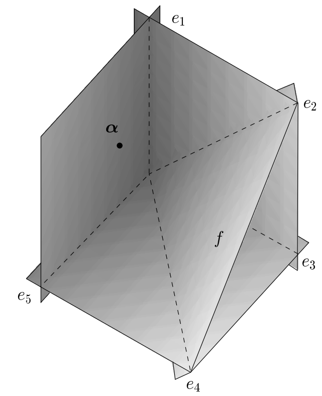

Figure 2: *A graphical depiction of four constraints in Ξ for a three-dimensional MCID.*

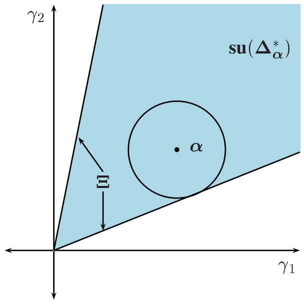

Figure 3: *Representing Ξ for a two-dimensional MCID by the largest ball centered at $\boldsymbol{\alpha}$.*

and storing it requires an amount of memory equal to $m|\Xi|$. This size can be problematic for larger decision problems — both in terms of memory requirements and in terms of representing the resulting Ξ to a human DM in a comprehensible manner.

We have devised a method for keeping the size of Ξ minimal during the analysis. By minimal we mean that Ξ does not include a constraint, $f : \boldsymbol{f} \cdot \boldsymbol{\gamma} \geq 0$, such that the set $\Xi \setminus \{f\}$ defines the same volume as Ξ. Conversely, if such a constraint, f, is in Ξ, we call it *superfluous*. Traditionally, such a task would be carried out by repeated applications of linear programming, but this can be done more efficiently when the number of dimensions, m, is significantly smaller than the number of decisions in the MCID. This seems to be the case in most examples of multi-currency decision problems in the literature, see e.g. (Boman et al., 1999; Skaaning, 2000). Unfortunately space restrictions prevent us from describing the method in detail, so only a brief informal outline will be given.

The approach rests on the fact that all constraints in Ξ define hyperplanes passing through the origin, and that the support, defined by them, is therefore a pyramid extending from the origin (see Figure 2). Hence, we can represent the area defined by Ξ as a set of edges instead of the set of constraints Ξ. In the figure there was a pyramid defined by the edges e_1, e_3, and e_5. The addition of a new constraint, f, defines a new pyramid whose edges are e_1, e_2, e_4, and e_5. Since f is not superfluous the addition of it thus renders the part defined by e_2, e_3, and e_4 invalid. This is recorded in the set of edges, describing the same area as Ξ, by dropping the edge e_3 and adding e_2 and e_4. By storing the pyramid as a list of edges sorted according to their distance to $\boldsymbol{\alpha}$, and calculating the minimal distance from $\boldsymbol{\alpha}$ to the hyperplane defined by f, we can quickly determine which of the edges can be rendered invalid by the addition of f. Once a constraint no longer participates in defining any edges, it is removed from Ξ.

4.4 Presenting the Support to a Human DM

Given that Ξ has been identified, we provide two compact but approximative representation techniques, which are useful for presenting the support to a DM.[5]

An immediate approach to representing Ξ in a compact manner is to define an m-dimensional ball, centered at $\boldsymbol{\alpha}$ with radius equal to the minimum Euclidean distance from $\boldsymbol{\alpha}$ to any one of the hyperplanes defined by the constraints in Ξ (see Figure 3). The approach also allows for a highly compact representation of Ξ during computation: Only a single scalar value (the radius of the ball) needs to be stored. Whenever a constraint closer to $\boldsymbol{\alpha}$ is identified, we replace the old radius with the distance from $\boldsymbol{\alpha}$ to this new constraint. Unfortunately this representation can be a rather crude approximation, as seen in Figure 3.

Another, more accurate, representation technique is to present Ξ by the largest ball that will fit into the

[5]These techniques do not presuppose that Ξ is minimal.

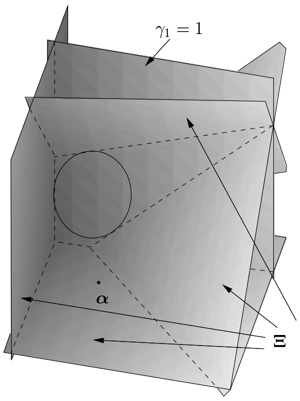

Figure 4: *Representing Ξ by the largest ball inscribed in the intersection between the constraints in Ξ and a hyperplane defined by $\gamma_i = 1$.*

support defined by it. That is, abandon $\boldsymbol{\alpha}$ as a fixed center of the representation. As Ξ describes an infinite pyramid, no such largest ball exists, so we propose to make a cut through the computed pyramid Ξ, by forcing one of the γ_is to be 1, and then representing the resulting intersection by the largest ball which can fit into it (see Figure 4). The first step of this approach corresponds to identifying a base currency, i, that all other currencies are compared to.

For the second step to be successfully completed it is necessary that the intersection of the pyramid and the cut is a bounded volume. This is the case if the hyperplane defined by $\gamma_i = 1$ intersects all hyperplanes defined by the constraints in Ξ. This is equivalent to having that there exists no f in Ξ, such that the hyperplane defined by f is parallel to the hyperplane defined by $\gamma_i = 1$. As all hyperplanes pass through the origin, it is sufficient to choose an i where the constraint $\gamma_i \geq 0$ is not in Ξ. Alternatively, additional linear constraints on parameters will have to be put into Ξ.

Once a base currency, i, has been chosen, each constraint, $f_1\gamma_1 + \cdots + f_m\gamma_m \geq 0$, in Ξ is replaced with the constraint $f_1\gamma_1 + \cdots + f_{i-1}\gamma_{i-1} + f_i + f_{i+1}\gamma_{i+1} + \cdots + f_m\gamma_m \geq 0$. The resulting set of constraints we denote $\Xi^{\gamma_i=1}$. To find the largest ball enclosed in the volume of $\mathbb{R}^{m-1}$ defined by the constraints in $\Xi^{\gamma_i=1}$ we solve the linear program

$$f^1(\boldsymbol{y}), \cdots, f^k(\boldsymbol{y}),$$
$$z \leq \text{dist}(\boldsymbol{y}, \boldsymbol{H}(f^1)), \cdots, z \leq \text{dist}(\boldsymbol{y}, \boldsymbol{H}(f^k)),$$

where $\boldsymbol{y}$ is in $\mathbb{R}^{m-1}$, $\{f^1, \ldots, f^k\} = \Xi^{\gamma_i=1}$, and $\text{dist}(\boldsymbol{y}, \boldsymbol{H}(f^i))$ denotes the Euclidean distance from $\boldsymbol{y}$ to the hyperplane $\boldsymbol{H}(f^i)$. The function that is to be optimised is z, and upon resolution the ball centered at $\boldsymbol{y}$ having radius z is the largest ball inscribed in the part of the support that corresponds to γ_i being 1.

Acknowledgments

We would like to thank Martin Raussen of the Department of Mathematical Sciences at Aalborg University for useful discussions and two anonymous reviewers for helpful comments and suggestions.

References

Magnus Boman, Paul Davidsson, and Håkan L. Younes. 1999. Artificial decision making under uncertainty in intelligent buildings. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 65–70.

Gregory F. Cooper. 1988. A method for using belief networks as influence diagrams. In *Proceedings of the Fourth Conference on Uncertainty in Artificial Intelligence*, pages 55–63.

Ronald A. Howard and James E. Matheson. 1984. Influence diagrams. *Readings on the Principles and Applications of Decision Analysis*, pages 720–763.

Anders L. Madsen and Finn V. Jensen. 1999. Lazy evaluation of symmetric Bayesian decision problems. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 382–390.

Ross D. Shachter. 1986. Evaluating influence diagrams. *Operations Research*, 34:871–882.

Prakash P. Shenoy. 1991. Valuation-based systems for Bayesian decision analysis. *Operations Research*, 40:463–484.

Claus Skaaning. 2000. A knowledge acquisition tool for Bayesian-network troubleshooters. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 549–557.

Joseph A. Tatman and Ross D. Shachter. 1990. Dynamic programming and influence diagrams. *IEEE Transactions on Systems Management and Cybernetics*, 20:265–279.