Multi-currency Influence Diagrams

Søren Holbech Nielsen, Thomas D. Nielsen, and Finn V. Jensen

Aalborg University, Aalborg, Denmark

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. The extension allows for these decision problems to be modelled in their original form. We present an algorithm that, given a linear 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 [4] 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 locality principle 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. For decision problems where several parties are affected by the decision making process, the utility functions for the individual stakeholders would also have to be transformed into a single utility function before reasoning can take place. Unfortunately, it is usually difficult to elicit the parameters, which governs such transformations (e.g. what is the monetary cost of one unit of discomfort?).[1] Moreover, the nature of such a transformation has a direct impact on the solution of the ID, but this effect cannot be made transparent when the transformation is tacit. Furthermore, 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 an optimal strategy composed of a policy for each decision. The result of such an analysis thus provides an indication of how robust the optimal strategy is in terms of the tacit conversion parameters. Regular sensitivity analysis (see e.g. [3] and [8]), on the other hand, provides robustness measures in terms of isolated deviances in one or more of the actual utility values in the ID, with no regard to how the uncertainties in these parameters are related. By encoding each value for which regular sensitivity analysis is to be performed by its own currency, the analysis could be performed by the method we propose here, and it can therefore be seen as a generalization of regular sensitivity analysis.

As the result of the analysis may be quite complex, we provide, in addition to this algorithm, methods for presenting the result to a DM in a comprehensible manner.

Example 1 (A Motivating Example). The ID in Fig. 1 models a decision problem where a doctor is faced with a patient. The health *Health*$_1$ of the patient at the time of the initial consultation is revealed only indirectly by the symptoms *Symptoms* exhibited by the patient. Based on the symptoms, the doctor must decide whether to perform a test *Test*. The test will produce a result *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.

[1] [12] 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. Also, in decision problems with several stakeholders, each currency can be seen as the utility for one specific stakeholder.

Now, before the doctor calculates an optimal strategy for his decision prob-

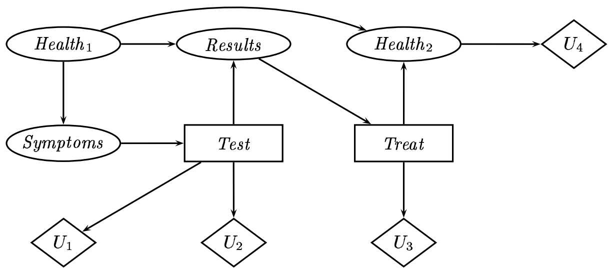

Fig. 1. *Example of an ID*

lem, he needs to transform U_1, U_2, U_3, and 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.

None of these questions can be answered from the ID alone.

As an example of how decision problems involving several stakeholders can be interpreted as a multi-currency problem, the ID introduced above can be seen as describing a conflict between the interests of the patient and the hospital (and/or the patient's medical insurance company) mediated by the doctor: By letting the patient specify utility values corresponding to utility nodes U_1 and U_4, and the hospital the ones for U_2 and U_3, we can investigate how the decisions of the doctor relate to these two stakeholders; he may be indulgent to please the patient more than the hospital or vice versa.

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 two 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 *chance variable*, which is a discrete variable outside the DM's direct control. A decision node (drawn as a rectangle) represents a *decision variable* (or simply a *decision*), which is a discrete variable under the DM's direct control. 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 [13]. When the meaning is obvious from the context, we use the terms node and variable interchangeably and do 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 immediately before D is decided upon. *No-forgetting* is assumed so that variables observed or decided upon immediately before previous decisions are remembered at subsequent decisions. 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 assume that the decisions are labelled $D_1, \ldots, D_n$, such that $i < j$ implies $D_i \prec D_j$. Furthermore, we 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

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

and no-forgetting amounts to the assumption that for any D_k all observations of variables in $\cup_{i<k} \boldsymbol{C}_i$ and decisions taken for $D_1, \ldots, D_{k-1}$ are remembered when the DM decides on D_k. 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:

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

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 distinguish between *probability potentials*, denoted by ϕ's, and *utility potentials*, denoted by ψ's. Furthermore, the set of variables $\{V_1, \ldots, V_k\}$ is referred to as the *domain* of f, denoted

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*[2] 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

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

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 (1) is the *expected utility* of the decision problem given the strategy $\boldsymbol{\Delta}$, and it is denoted eu$(\boldsymbol{\Delta})$. Not all variables in the past of a decision D are necessarily relevant for D. Therefore, we call a variable V *required* for 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 we denote by $\mathbf{req}(D)$.[3] We may then redefine a policy to be a function $\delta_D : \mathbf{sp}(\mathbf{req}(D)) \to \mathbf{sp}(D)$.
When optimal policies are to be identified, it is usually easier to work with a recursive expression for the maximum expected utility instead of (1). An example is the *variable elimination* algorithm [5]: Define

$$\boldsymbol{\Phi}_{(n)} = \{\phi \in \boldsymbol{\Phi} \mid \mathrm{dom}(\phi) \cap (\boldsymbol{C}_i \cup \{D_i\}) \neq \varnothing\},$$

and similarly for $\boldsymbol{\Psi}_{(n)}$. Then

$$\delta^*_{D_n}(\boldsymbol{c}) = \arg\max_{d \in \mathbf{sp}(D_n)} \sum_{\boldsymbol{e} \in \mathbf{sp}(\boldsymbol{C}_n)} \prod_{\phi \in \boldsymbol{\Phi}_{(n)}} \phi(\boldsymbol{c}, d, \boldsymbol{e}) \sum_{\psi \in \boldsymbol{\Psi}_{(n)}} \psi(\boldsymbol{c}, d, \boldsymbol{e}) , \quad (2)$$

and we set

$$\phi_n(\boldsymbol{c}) = \sum_{\boldsymbol{e} \in \mathbf{sp}(\boldsymbol{C}_n)} \prod_{\phi \in \boldsymbol{\Phi}_{(n)}} \phi(\boldsymbol{c}, \delta^*_{D_n}(\boldsymbol{c}), \boldsymbol{e}) , \quad (3)$$

and

$$\psi_n(\boldsymbol{c}) = \left(\sum_{\boldsymbol{e} \in \mathbf{sp}(\boldsymbol{C}_n)} \prod_{\phi \in \boldsymbol{\Phi}_{(n)}} \phi(\boldsymbol{c}, \delta^*_{D_n}(\boldsymbol{c}), \boldsymbol{e}) \sum_{\psi \in \boldsymbol{\Psi}_{(n)}} \psi(\boldsymbol{c}, \delta^*_{D_n}(\boldsymbol{c}), \boldsymbol{e}) \right) / \phi_n(\boldsymbol{c}) , \quad (4)$$

[3] [9] provides an operational method for determining $\mathbf{req}(D)$ for any decision D in an ID.

for all configurations $\boldsymbol{c}$ over $\mathbf{req}(D_n)$. For all $i < n$ we recursively define

$$\boldsymbol{\Phi}_{(i)} = \{\phi \in \boldsymbol{\Phi} \cup \{\phi_{i+1}, \dots, \phi_n\} \mid \mathrm{dom}(\phi) \cap (\boldsymbol{C}_i \cup \{D_i\}) \neq \varnothing\} \setminus \cup_{j>i} \boldsymbol{\Phi}_{(j)} \quad (5)$$

and similarly for $\boldsymbol{\Psi}_{(i)}$ with $\psi_{i+1}, \dots, \psi_n$. We then get

$$\delta^*_{D_i}(\boldsymbol{c}) = \arg\max_{d \in \mathbf{sp}(D_i)} \sum_{\boldsymbol{e} \in \mathbf{sp}(\boldsymbol{C}_i)} \prod_{\phi \in \boldsymbol{\Phi}_{(i)}} \phi(\boldsymbol{c}, d, \boldsymbol{e}) \sum_{\psi \in \boldsymbol{\Psi}_{(i)}} \psi(\boldsymbol{c}, d, \boldsymbol{e}) \; , \quad (6)$$

and set

$$\phi_i(\boldsymbol{c}) = \sum_{\boldsymbol{e} \in \mathbf{sp}(\boldsymbol{C}_i)} \prod_{\phi \in \boldsymbol{\Phi}_{(i)}} \phi(\boldsymbol{c}, \delta^*_{D_i}(\boldsymbol{c}), \boldsymbol{e}) \; , \quad (7)$$

and

$$\psi_i(\boldsymbol{c}) = \left(\sum_{\boldsymbol{e} \in \mathbf{sp}(\boldsymbol{C}_i)} \prod_{\phi \in \boldsymbol{\Phi}_{(i)}} \phi(\boldsymbol{c}, d, \boldsymbol{e}) \Big(\sum_{\psi \in \boldsymbol{\Psi}_{(i)}} \psi(\boldsymbol{c}, d, \boldsymbol{e}) \Big) \right) / \phi_i(\boldsymbol{c}) \; , \quad (8)$$

for all configurations $\boldsymbol{c}$ over $\mathbf{req}(D_i)$. We may then write the maximum expected utility of the ID as

$$\mathrm{eu}(\boldsymbol{\Delta}^*) = \sum_{\boldsymbol{e} \in \mathbf{sp}(\boldsymbol{C}_0)} \prod_{\phi \in \boldsymbol{\Phi}_{(0)}} \phi(\boldsymbol{e}) \sum_{\psi \in \boldsymbol{\Psi}_{(0)}} \psi(\boldsymbol{e}) \; .$$

Given an ID and a realisation, an optimal strategy may be found through the use of any one of a number of algorithms including [5], [7], [10], and [11], which all utilise the distributive and associative law on the expressions in (2) to (8).

3 Multi-currency Influence Diagrams

From the summations of utility potentials in (2) to (8), it is clear that these must be of the same type, i.e. defined over the same currency; in the ID framework this is tackled by transforming the different currencies into one currency during construction of the ID. We now introduce a framework capable of handling 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, the syntax and semantics of IDs described in Sec. 2 carry over to MCIDs, except for the requirement that the local utility functions must be an additive decomposition of the global utility function. By assuming some arbitrary, but fixed, order of the currencies in the MCID $s_1, \dots, s_m$ we may refer to the currency of a local utility function by a natural number $i \in \{1, \dots, m\}$.
In what follows we refer to the number of different currencies of an MCID

as the *dimension* of the MCID, and throughout we assume this to be m. A realisation of an MCID is therefore a tuple $(\boldsymbol{\Phi}, \boldsymbol{\Psi}_1, \ldots, \boldsymbol{\Psi}_m)$, where $\boldsymbol{\Psi}_i$ is the set of local utility functions of currency i. We require that, for each $\boldsymbol{\Psi}_i$, its elements form an additive decomposition of a utility function, which encodes the same preference ordering as if the DM had disregarded all consequences measured in currencies different from i, and that there is some linear combination of these utility functions describing the decision makers preferences in full. Note that it follows that an MCID of dimension 1 is an ID, and hence from this point on we assume m to be larger than 1.
A strategy for an MCID is the same as for an ID: A prescription of choices given the required variables in the past. 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.

Example 2 (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 local utilities U_1 and U_2 defined as $U_1(D = d_1) = 1A$, $U_1(D = d_2) = 5A$, $U_2(D = d_1) = 2B$, and $U_2(D = d_2) = 1B$.
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 $\mathrm{eu}(D = d_1) \geq \mathrm{eu}(D = d_2)$, which is equivalent to

$$1A + 2B \geq 5A + 1B \;\Leftrightarrow\; -4A + B \geq 0 \;,$$

If we regard the currency name A as a real variable, representing the DM's degree of appreciation of amounts of A, and similarly 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 rendering 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, we see that the payoff of following a specific strategy is an element of $\mathbb{R}^m$ (e.g. $(5, 1)$ in Ex. 2) rather than a scalar value as is the case with strategies for IDs. The means we use for comparing amounts of different currencies are 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 (CM) for $\mathcal{I}$.

The semantics of a CM $\boldsymbol{\alpha}$, *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. We also say that the DM *adheres* to $\boldsymbol{\alpha}$. This way α_i becomes a measure of appreciation for the DM of one unit of currency i. In a multi-stakeholder setting, α_i becomes a weight of importance attributed to satisfying the i'th stakeholder compared to the other stakeholders. In any case, the objective of the DM is to maximize the *expected global utility* of a strategy $\boldsymbol{\Delta}$ given by

$$\boldsymbol{\alpha} \cdot \mathrm{eu}(\boldsymbol{\Delta}) = \sum_{i=1}^{m} \alpha_i \sum_{\boldsymbol{c} \in \mathbf{sp}(\boldsymbol{V}_C \cup \boldsymbol{V}_D)} \Big(\prod_{D \in \boldsymbol{V}_D} P_{\delta_D}(\boldsymbol{c}) \prod_{\phi \in \boldsymbol{\Phi}} \phi(\boldsymbol{c}) \sum_{\psi_i \in \boldsymbol{\Psi}_i} \psi_i(\boldsymbol{c}) \Big) . \quad (9)$$

It follows that we assume the preferential relationship between currencies is a linear one. Thus, in Ex. 2, a DM adhering to the CM $(3, 2)$, would have an expected global utility of $3 \cdot 1 + 2 \cdot 2 = 7$ for choosing d_1 and $3 \cdot 5 + 2 \cdot 1 = 17$ for choosing d_2. In order to maximize the expected global utility he should therefore choose d_2.
We use bold face Latin letters ($\boldsymbol{f}$, $\boldsymbol{q}$, etc.) to denote arbitrary points in $\mathbb{R}^m$, and Greek letters ($\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$) to denote points when we want to emphasise that they are CMs. In general, we use q_i to refer to the i'th coordinate of a point $\boldsymbol{q}$. We shall not distinguish between a point $\boldsymbol{q}$ and the corresponding vector going from the origin to $\boldsymbol{q}$. In what follows we furthermore use $\boldsymbol{f} \cdot \boldsymbol{q}$ to denote the scalar product $\sum_i f_i q_i$.
In this paper we assume, without loss of generality, that each element of a currency mapping is positive, i.e. a CM 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.
If for a strategy, $\boldsymbol{\Delta}$, we have that $\boldsymbol{\alpha}\cdot\mathrm{eu}(\boldsymbol{\Delta})$ is greater than or equal to $\boldsymbol{\alpha}\cdot\mathrm{eu}(\boldsymbol{\Delta}')$ for all other strategies $\boldsymbol{\Delta}'$, it means that a DM adhering to $\boldsymbol{\alpha}$ appreciates the expected utility of following $\boldsymbol{\Delta}$ at least as much as that of following any other strategy, and we consequently say that $\boldsymbol{\Delta}$ is optimal given the CM $\boldsymbol{\alpha}$. As can be seen from (9), and the distributive law of the scalar product, the optimal strategy for $\boldsymbol{\alpha}$ is equivalent to the one obtained by solving an ID resulting from multiplying the individual utilities in the MCID with $\boldsymbol{\alpha}$ beforehand. We denote an optimal strategy for an MCID given a CM $\boldsymbol{\alpha}$ as $\boldsymbol{\Delta}^*_{\boldsymbol{\alpha}}$. As several strategies might give rise to the same expected utility, the set of all optimal strategies corresponding to $\boldsymbol{\alpha}$ is denoted as $\overline{\boldsymbol{\Delta}^*_{\boldsymbol{\alpha}}}$.

4 Support Analysis of MCIDs

Clearly, if we are given a CM $\boldsymbol{\alpha}$ in addition to an MCID we can solve it by means of simply converting the MCID into an ID, through multiplying each local utility function of currency i by α_i, and then solving the resulting ID. This simple solution method allows for optimal strategies to be computed for any $\boldsymbol{\alpha}$, and the distinction between MCIDs and CMs emphasises the assumptions leading to the results. However, we would not be closer to answering the second question in the motivating example, viz. what other currency mappings would lead to the same optimal strategy? 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 obtain this transparency by postponing the conversion of utilities until it is needed, so that we can analyse the requirements these conversions bestow upon those CMs giving rise to the same optimal strategy.

4.1 Preliminaries

Given an MCID and a strategy $\boldsymbol{\Delta}$, we call the set $\mathbf{su}(\boldsymbol{\Delta}) = \{\boldsymbol{\beta} \in \mathbb{R}^m_+ \mid \boldsymbol{\Delta} \in \overline{\boldsymbol{\Delta}^*_{\boldsymbol{\beta}}}\}$ the *support* of $\boldsymbol{\Delta}$. Intuitively, $\mathbf{su}(\boldsymbol{\Delta})$ is the set of CMs for which $\boldsymbol{\Delta}$ is an optimal strategy. We refer to the process of calculating the support of an optimal strategy as performing support analysis of the MCID. In Ex. 2 we actually found the support of both strategies $D = d_1$ and $D = d_2$ (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 postpone the conversion of utility potentials until needed. Hence, we introduce a new type of potential, which can represent utility functions of several currencies:[4] 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. In Ex. 2 we could have exchanged the two utility functions U_1 and U_2 with the two MCUPs θ_1 and θ_2, where $\theta_1(d_1) = (1, 0)$, $\theta_1(d_2) = (5, 0)$, $\theta_2(d_1) = (0, 2)$, $\theta_2(d_2) = (0, 1)$. For a DM whose preferences are reflected by the CM $\boldsymbol{\beta}$, $\boldsymbol{\beta} \cdot \theta(\boldsymbol{S})$ 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 to the DM is equivalent to the 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. For instance, in Ex. 2 we have that $\theta_1 + \theta_2$ is the MCUP θ_+ where $\theta_+(d_1) = (1, 2)$ and $\theta_+(d_2) = (5, 1)$, which to a DM adhering to $\boldsymbol{\beta} = (3, 2)$ would be equivalent to a utility potential ψ, where $\psi(d_1) = 7$ and $\psi(d_2) = 17$.

4.2 Support Analysis

We are now ready to give a procedure for performing support analysis of an MCID. We describe the method first, and give an example of its application

[4] Such potentials are not part of the MCID framework as such, but rather data structures used by the proposed method for doing support analysis.

afterwards. We assume the existence of an optimal strategy $\boldsymbol{\Delta}^*_{\boldsymbol{\alpha}}$ determined by some initial currency mapping $\boldsymbol{\alpha}$ as well as a realisation $(\boldsymbol{\Phi}, \boldsymbol{\Psi}_1, \ldots, \boldsymbol{\Psi}_m)$, and we look for the support of $\boldsymbol{\Delta}^*_{\boldsymbol{\alpha}}$ for this realisation. The method is inspired by that of Lazy evaluation presented in [7] 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 $\boldsymbol{\Xi}$ and $\boldsymbol{\Theta}$ are generated, where $\boldsymbol{\Xi}$ will hold inequalities defining $\mathbf{su}(\boldsymbol{\Delta}^*_{\boldsymbol{\alpha}})$, and $\boldsymbol{\Theta}$ is a container for MCUPs used in the identification phase. Second, for each currency i each potential ψ in $\boldsymbol{\Psi}_i$ is converted to a MCUP θ and put into $\boldsymbol{\Theta}$, 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 of linear inequalities (Step 4) that need to be fulfilled if $\boldsymbol{\Delta}^*_{\boldsymbol{\alpha}}$ is to be optimal. We refer to the inequalities in $\boldsymbol{\Xi}$ 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[5], and
$$\boldsymbol{\Theta} \leftarrow (\boldsymbol{\Theta} \setminus \boldsymbol{\Theta}_V) \cup \{\theta_V(\delta^*_V)\} ,$$

[5] Note that any potential in $\boldsymbol{\Phi}_V$ must be constant over V

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

$$\Xi \leftarrow \Xi \cup \{\boldsymbol{f}_{\mathbf{c},v} \cdot \gamma \geq 0\} , \tag{10}$$

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

Before proceeding, we illustrate the workings of the algorithm by an example:

Example 3. We specify the ID from Ex. 1 as an MCID. We assume that the state space of the variables are as follows:

$$\begin{aligned}
&\mathbf{sp}(\mathit{Health}_1(H_1)) = \{\mathit{bad}(b_1), \mathit{healthy}(h_1)\} ,\\
&\mathbf{sp}(\mathit{Symptoms}(S)) = \{\mathit{symptoms}(sy), \mathit{none}(no)\} ,\\
&\mathbf{sp}(\mathit{Test}(Te)) = \{\mathit{test}(te), \mathit{no\ test}(nte)\} ,\\
&\mathbf{sp}(\mathit{Results}(R)) = \{\mathit{positive}(po), \mathit{negative}(ne), \mathit{no\ result}(nr)\} ,\\
&\mathbf{sp}(\mathit{Treat}(Tr)) = \{\mathit{treat}(tr), \mathit{no\ treatment}(ntr)\} , \text{ and}\\
&\mathbf{sp}(\mathit{Health}_2(H_2)) = \{\mathit{bad}(b_2), \mathit{healthy}(h_2)\} .
\end{aligned}$$

The three currencies we work with are, and are ordered as *comfort*(c), *dollars*(\$), and *health*(h). The realization that we work with is defined by the following parameters:

$$\begin{aligned}
&P(b_1) = 0.5 , \quad P(sy|b_1) = 0.95 , \quad P(sy|h_1) = 0.1 , \quad P(nr|nte, \cdot) = 1 ,\\
&P(nr|te, \cdot) = 0 , \quad P(po|te, b_1) = 0.99 , \quad P(po|te, h_1) = 0.01 ,\\
&P(b_2|b_1, tr) = 0.001 , \quad P(b_2|b_1, ntr) = 0.999 , \quad P(b_2|h_1, tr) = 0 ,\\
&P(b_2|h_1, ntr) = 0.0001 , \quad U_1(te, nte) = (-1c, 0) , \quad U_2(te, nte) = (-\$1000, 0) ,\\
&U_3(tr, ntr) = (-\$10000, 0) , \text{ and } \quad U_4(b_2, h_2) = (-1h, 0) .
\end{aligned}$$

We assume that the doctor is adhering to a CM $\boldsymbol{\alpha} = (10, 1, 100000)$ meaning that he regards discomfort on behalf of the patient as bad as the loss of \$10 and the death of the patient as bad as a loss of \$100000. The CM corresponds to a strategy prescribing *treatment* either if a *test* was conducted, and a *positive* test result was gotten, or *no test* was conducted and there was *symptoms*, and *no treatment* otherwise.
In the initialization part of the algorithm, we construct an empty set Ξ and convert the four utility potentials into a set of four MCUPs

$$\begin{aligned}
\boldsymbol{\Theta} = \{&\theta_1(te, nte) = ([-1, 0, 0], [0, 0, 0]) , \ \theta_2(te, nte) = ([0, -1000, 0], [0, 0, 0]) ,\\
&\theta_3(tr, ntr) = ([0, -10000, 0], [0, 0, 0]) , \ \theta_4(b_2, h_2) = ([0, 0, -1], [0, 0, 0])\} .
\end{aligned}$$

Next we perform variable elimination in an order that respects the $\prec$-ordering of the MCID: First *Health*$_2$, then *Health*$_1$, *Treat*, *Results*, *Test*, and lastly *Symptoms*. Marginalizing out *Health*$_2$, according to Steps 2 and 3 results in dropping $P(H_2|H_1,Tr)$ from the set of probability potentials and replacing θ_4 with a new MCUP:

$$\theta_{H_1,Tr} = \frac{\sum_{H_2} P(H_2|H_1,Tr)\cdot\theta_4}{\sum_{H_2} P(H_2|H_1,Tr)} ,$$

in which $\theta_{H_1,Tr}(b_1,tr) = [0,0,-10^{-3}]$, $\theta_{H_1,Tr}(b_1,ntr) = [0,0,-0.999]$, $\theta_{H_1,Tr}(h_1,tr) = [0,0,0]$, and $\theta_{H_1,Tr}(h_1,ntr) = [0,0,-10^{-4}]$. Further marginalizing out *Health*$_1$ (again according to Steps 2 and 3) replaces all probability potentials with

$$\phi_{R,S,Te} = \sum_{H_1} P(H_1)\cdot P(S|H_1)\cdot P(R|H_1,Te)$$

and $\theta_{H_1,Tr}$ with

$$\theta_{R,S,Te,Tr} = \frac{\sum_{H_1} P(H_1)\cdot P(S|H_1)\cdot P(R|H_1,Te)\cdot\theta_{H_1,Tr}}{\phi_{S,R,Te}} .$$

This last potential is shown in Table 1.
When marginalizing *Treat*, we construct the MCUP $\theta^{+}_{R,S,Te,Tr} = \theta_3+\theta_{R,S,Te,Tr}$ (shown in Table 2), according to Step 2, and by marginalizing *Treat* out of this, according to Step 4 and $\boldsymbol{\Delta}^{*}_{\boldsymbol{\alpha}}$, we end up with a new MCUP $\theta_{R,S,Te}$ (not shown), but also add 12 constraints to $\boldsymbol{\Xi}$ – one for each configuration over *Results*, *Symptoms*, and *Test*. For instance, for the configuration (po, sy, te), we construct the constraint ($\boldsymbol{\gamma} = (\gamma_c, \gamma_\$, \gamma_h)$)

$$\begin{aligned}((0,-10000,-9.99\cdot 10^{-4}) - (0,0,-9.98\cdot 10^{-1}))\boldsymbol{\gamma} &\geq 0\\ -10000\gamma_\$ + 9.97\cdot 10^{-1}\gamma_h &\geq 0 ,\end{aligned}$$

which is satisfied by $\boldsymbol{\alpha}$, as can easily be verified. After the other 11 constraints have been calculated the algorithm continues with elimination of the remaining variables.

Table 1. $\theta_{R,S,Te,Tr}$

		te		nte	
		tr	ntr	tr	ntr
po	sy	$[0,0,-9.99\cdot 10^{-4}]$	$[0,0,-9.98\cdot 10^{-1}]$	$[0,0,0]$	$[0,0,0]$
	no	$[0,0,-8.46\cdot 10^{-4}]$	$[0,0,-8.45\cdot 10^{-1}]$	$[0,0,0]$	$[0,0,0]$
ne	sy	$[0,0,-8.76\cdot 10^{-4}]$	$[0,0,-8.76\cdot 10^{-2}]$	$[0,0,0]$	$[0,0,0]$
	no	$[0,0,-5.61\cdot 10^{-7}]$	$[0,0,-6.60\cdot 10^{-4}]$	$[0,0,0]$	$[0,0,0]$
nr	sy	$[0,0,0]$	$[0,0,0]$	$[0,0,-9.05\cdot 10^{-4}]$	$[0,0,-9.04\cdot 10^{-1}]$
	no	$[0,0,0]$	$[0,0,0]$	$[0,0,-5.26\cdot 10^{-5}]$	$[0,0,-5.27\cdot 10^{-2}]$

Table 2. $\theta^{+}_{R,S,Te,Tr}$

		te		nte	
		tr	ntr	tr	ntr
po	sy	$[0, -10^4, -9.99 \cdot 10^{-4}]$	$[0, 0, -9.98 \cdot 10^{-1}]$	$[0, -10^4, 0]$	$[0, 0, 0]$
	no	$[0, -10^4, -8.46 \cdot 10^{-4}]$	$[0, 0, -8.45 \cdot 10^{-1}]$	$[0, -10^4, 0]$	$[0, 0, 0]$
ne	sy	$[0, -10^4, -8.76 \cdot 10^{-4}]$	$[0, 0, -8.76 \cdot 10^{-2}]$	$[0, -10^4, 0]$	$[0, 0, 0]$
	no	$[0, -10^4, -5.61 \cdot 10^{-7}]$	$[0, 0, -6.60 \cdot 10^{-4}]$	$[0, -10^4, 0]$	$[0, 0, 0]$
nr	sy	$[0, -10^4, 0]$	$[0, 0, 0]$	$[0, -10^4, -9.05 \cdot 10^{-4}]$	$[0, 0, -9.04 \cdot 10^{-1}]$
	no	$[0, -10^4, 0]$	$[0, 0, 0]$	$[0, -10^4, -5.26 \cdot 10^{-5}]$	$[0, 0, -5.27 \cdot 10^{-2}]$

As presented here, the algorithm presupposes that $\boldsymbol{\Delta}^*_{\boldsymbol{\alpha}}$ has been computed beforehand. Alternatively, the Lazy evaluation algorithm itself can easily be interleaved 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 (2) to (8). We conjecture that any such adaptation would identify the support set as long as the constraints are identified and stored. With or without this modification, though, we have the following important result:

Theorem 1. *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 $\boldsymbol{su}(\boldsymbol{\Delta})$ iff $\boldsymbol{\beta}$ satisfies all inequalities in Ξ.*

Proof. The "if" part of the theorem is obvious. We therefore only show the "only if" part, viz. that if $\boldsymbol{\beta}$ fails to satisfy at least one constraint in Ξ, then $\boldsymbol{\Delta}$ cannot be an optimal strategy for a DM adhering to $\boldsymbol{\beta}$, and hence that $\boldsymbol{\beta}$ is not an element of $\mathbf{su}(\boldsymbol{\Delta})$.
Without loss of generality, assume that $f \in \Xi$ is the first constraint not satisfied by $\boldsymbol{\beta}$ that is identified by the algorithm. This happens during elimination of some decision D_i in Step 4 of the algorithm, with some MCUP θ_{D_i} having been calculated during Step 2. For some configuration $\boldsymbol{c}$ over $\mathrm{dom}(\theta_{D_i}) \setminus \{D_i\}$ and state $v \neq \delta_{D_i}$ we must have that f is

$$(\theta_{D_i}(\boldsymbol{c}, \delta_{D_i}(\boldsymbol{c})) - \theta_{D_i}(\boldsymbol{c}, v)) \cdot \boldsymbol{\gamma} \geq 0 \ .$$

Since $\boldsymbol{\beta}$ fails to satisfy this constraint, it follows that

$$(\theta_{D_i}(\boldsymbol{c}, \delta_{D_i}(\boldsymbol{c})) - \theta_{D_i}(\boldsymbol{c}, v)) \cdot \boldsymbol{\beta} \not\geq 0 \ ,$$

which is equivalent to

$$\theta_{D_i}(\boldsymbol{c}, \delta_{D_i}(\boldsymbol{c})) \cdot \boldsymbol{\beta} < \theta_{D_i}(\boldsymbol{c}, v) \cdot \boldsymbol{\beta} \ .$$

We construct the strategy $\boldsymbol{\Delta}'$ obtained from $\boldsymbol{\Delta}$ by letting $\delta'_{D_i}(\boldsymbol{c}) = v$ and leaving all other policies intact. We then have that $\mathrm{eu}(\boldsymbol{\Delta}') \cdot \boldsymbol{\beta}$ is strictly greater than $\mathrm{eu}(\boldsymbol{\Delta}) \cdot \boldsymbol{\beta}$, and hence that $\boldsymbol{\Delta}$ cannot be an optimal strategy for a DM adhering to $\boldsymbol{\beta}$.

Glancing over the constraints identified in Ex. 3 it is clear that each constraint $f : a\gamma_1 + b\gamma_2 + c\gamma_3 \geq 0$ defines a hyperplane in $\mathbb{R}^3$ given by the equation $a\gamma_1 + b\gamma_2 + c\gamma_3 = 0$. As the zero vector $\mathbf{0} = (0, 0, 0)$ is a satisfying solution to each equation, it follows that all these hyperplanes must pass through the origin of $\mathbb{R}^3$, and hence that the points satisfying all constraints must lie in a bottomless pyramid extending from the origin, as illustrated in Fig. 2(a). A corresponding visualization for a two-dimensional MCID is shown in Fig. 2(b). That the support of a strategy extends indefinitely from the origin as a pyramid, also makes sense from a purely semantical point of view: If a CM is scaled by multiplying each entry by some positive constant, the relative difference in appreciation between amounts of the individual currencies, for a DM adhering to this CM, stays the same. Hence, if a CM renders some strategy optimal, that strategy should also be optimal for any positively scaled version of this CM. From a more formal point of view, we also have that, for any CM $\boldsymbol{\alpha}$ and two strategies $\boldsymbol{\Delta}_i$ and $\boldsymbol{\Delta}_j$, if $\boldsymbol{\alpha} \cdot \mathrm{eu}(\boldsymbol{\Delta}_i) \geq \boldsymbol{\alpha} \cdot \mathrm{eu}(\boldsymbol{\Delta}_j)$ then it must necessarily be the case that $c\boldsymbol{\alpha} \cdot \mathrm{eu}(\boldsymbol{\Delta}_i) \geq c\boldsymbol{\alpha} \cdot \mathrm{eu}(\boldsymbol{\Delta}_j)$ for any $c > 0$, and hence that the pyramidal forms of support areas are what we should expect.

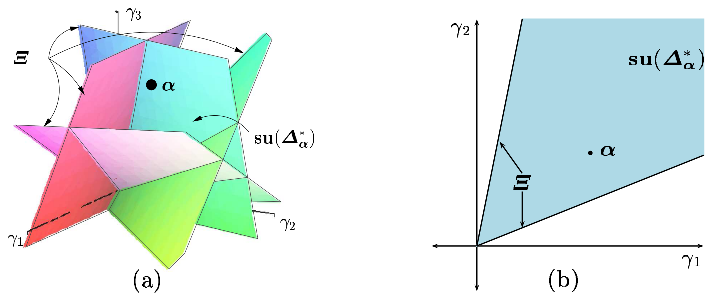
γ_3
Ξ
$\boldsymbol{\alpha}$
$\mathrm{su}(\boldsymbol{\Delta}^*_{\boldsymbol{\alpha}})$
γ_1
γ_2
(a)
γ_2
$\mathrm{su}(\boldsymbol{\Delta}^*_{\boldsymbol{\alpha}})$
$\boldsymbol{\alpha}$
Ξ
γ_1
(b)

Fig. 2. *Supports for a three-dimensional (a) and a two-dimensional MCID (b)*

5 Finding a Minimal Support Set

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

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

and storing it requires an amount of memory proportional to $m \cdot |\Xi|$. 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, 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. [1] and [12].
The approach is purely geometric and rests on the already mentioned fact that all constraints in Ξ define a pyramid extending from the origin of $\mathbb{R}^m$ (see Figs. 2(a) and 3(a)), and a black box view of the support analysis as a simple constraint generating process. For sake of clarity, we first describe the method informally in the intuitively understood three dimensional setting, and then give a formal presentation of the more general m-dimensional setting.
As the support of a strategy is a pyramid extending from the origin, we can represent the volume defined by Ξ as a set of lines extending from the origin, each corresponding to the intersection of two hyperplanes defined by constraints in Ξ (see Fig. 3(a), which illustrates the support defined by the initial set of constraints $\gamma_1 \geq 0$, $\gamma_2 \geq 0$, and $\gamma_3 \geq 0$, as well as an additional constraint f). We refer to such lines as *edges* of the pyramid, and we denote the set of them as $\boldsymbol{E}_\Xi$. When a new constraint g is to be added to Ξ (see Fig. 3(b)) it may be superfluous. With the alternative representation this question can be restated as whether the points on any of the edges in $\boldsymbol{E}_\Xi$ fail to satisfy g. If this is the case, g is not superfluous. In Fig. 3(b) we have that there was originally a pyramid defined by the edges e_1, e_2, e_3, and e_4, but as all points (but the origin) on e_1 and e_2 fail to satisfy the new constraint g, it is not superfluous. Therefore it is added to Ξ, which now defines a new pyramid whose edges are e_3, e_4, e_5, and e_6. Thus $\boldsymbol{E}_\Xi$ is updated by dropping e_1 and e_2 and adding e_5 and e_6. This means that the original constraint $\gamma_1 \geq 0$ no longer participates in defining any of the edges in $\boldsymbol{E}_\Xi$, and *it* is therefore superfluous now. Consequently, $\gamma_1 \geq 0$ must be dropped from Ξ to keep it minimal (see Fig. 3(c)). That is, in this alternative representation, we can

also detect when old constraint are rendered superfluous by new constraints. There is one main hurdle to be overcome by an implementation: Whenever

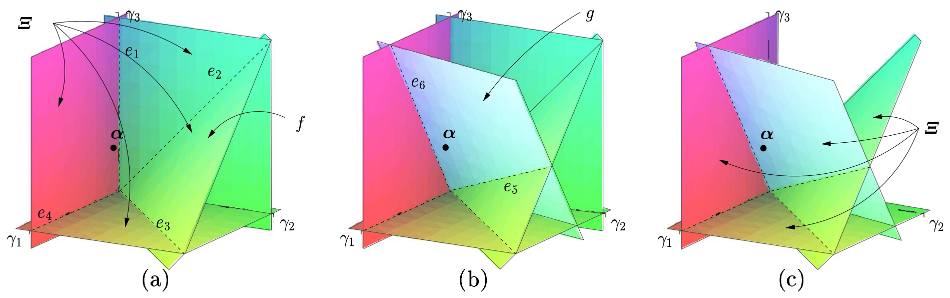

Fig. 3. *Keeping Ξ minimal when a new constraint g is added. Edges are shown as dashed lines*

a new constraint h is identified, the edges in $\boldsymbol{E}_{\Xi}$ need to be checked for points not satisfying h. By storing the pyramid as a list of edges sorted according to their angular distance to $\boldsymbol{\alpha}$ (see Fig. 4(a)), and calculating the minimal angular distance from $\boldsymbol{\alpha}$ to the hyperplane defined by h, we can quickly determine those edges that can contain points not satisfying h, i.e. those with an angular distance greater than the distance to the hyperplane defined by h. If no such edges exist (as in Fig. 4(b)), h is clearly superfluous. If such edges do exist, they have to be checked against h one at the time, and only if all of them satisfy h, it must be superfluous. However, as the set of constraints grow, the girth of the pyramid becomes smaller, and we would therefore expect that more new constraints fail the initial check of being closer to $\boldsymbol{\alpha}$ than any of the edges in Ξ, and that this further check by enumeration can be avoided.

To describe the approach precisely, we formalize and generalize the discussion above: First, if $f(\boldsymbol{\gamma}) = \boldsymbol{f} \cdot \boldsymbol{\gamma} \geq 0$ is a constraint in Ξ and $\boldsymbol{\beta}$ is some point in $\mathbb{R}^m$, then we use $f(\boldsymbol{\beta})$ to state that $\boldsymbol{f} \cdot \boldsymbol{\beta} \geq 0$ and $\neg f(\boldsymbol{\beta})$ to state that $\boldsymbol{f} \cdot \boldsymbol{\beta} < 0$. Furthermore, we denote the hyperplane $\{\boldsymbol{\beta} \in \mathbb{R}^m \mid \boldsymbol{f} \cdot \boldsymbol{\beta} = 0\}$ as $\boldsymbol{H}(f)$, and talk of Ξ as consisting of hyperplanes when it introduces no ambiguity. A constraint f in Ξ is then defined to be *superfluous* in Ξ if there exists no point $\boldsymbol{\beta}$ in $\mathbb{R}^m$, such that $\neg f(\boldsymbol{\beta})$ and $g(\boldsymbol{\beta})$ for all $g \neq f$ in Ξ. If no constraint in a set of constraints Ξ is superfluous, we say that Ξ is *minimal*. The *angular distance* from $\boldsymbol{\beta}$ to $\boldsymbol{\alpha}$ is given by

$$\angle\boldsymbol{\beta}\boldsymbol{\alpha} = \cos^{-1} \frac{\boldsymbol{\beta} \cdot \boldsymbol{\alpha}}{||\boldsymbol{\beta}|| \cdot ||\boldsymbol{\alpha}||} .$$

Here and henceforth $||\boldsymbol{q}||$ denotes the Euclidean length of the vector $||\boldsymbol{q}||$. Moreover, we say that the set of constraints Ξ constitutes a *pyramid*, if for

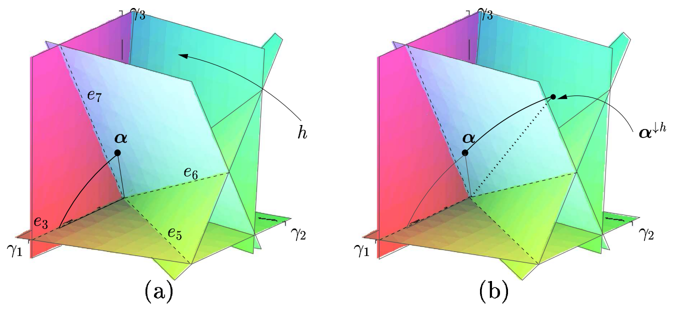

Fig. 4. *Rejecting a constraint, h, because of its angular distance to $\boldsymbol{\alpha}$*

all points $\boldsymbol{\beta}$ in $\mathbb{R}^m$, where the angular distance to $\boldsymbol{\alpha}$ is greater than 90°, there exists at least one constraint f in $\varXi$, such that $\neg f(\boldsymbol{\beta})$. The previously introduced assumption of only considering CMs in $\mathbb{R}^m_+$, we restate as an inclusion of the m constraints $\gamma_i \geq 0$ in $\varXi$ from the outset of the support analysis. This ensures that $\varXi$ is a pyramid at the beginning of the support analysis. The triangle inequality ensures that the points in a pyramid having the largest angular distance to $\boldsymbol{\alpha}$ must lie at the intersection of a number of hyperplanes in $\varXi$. We need to consider intersections of exactly $m-1$ hyperplanes. The reason is that an intersection of k non-parallel hyperplanes in $\mathbb{R}^m$ describe an $m-k$ dimensional subspace of $\mathbb{R}^m$, and the points in this subspace will have varying angular distances to $\boldsymbol{\alpha}$ unless the subspace has dimension 1, and this is only the case when k is 1. If part of such an intersection lies within $\varXi$, we refer to that part as an *edge*. An edge determined by constraints $f_1, \ldots, f_{m-1}$ we represent by a pair $e = (\boldsymbol{I}, \boldsymbol{p})$, where $\boldsymbol{I} = \{f_1, \ldots, f_{m-1}\}$ and $\boldsymbol{p} \in \mathbb{R}^m_+ \cap \boldsymbol{H}(f_1) \cap \cdots \cap \boldsymbol{H}(f_{m-1})$. The set of edges of a pyramid $\varXi$ is denoted $\boldsymbol{E}_{\varXi}$, and we assume it to be kept sorted such that $(\boldsymbol{I}_i, \boldsymbol{p}_i)$ is stored before $(\boldsymbol{I}_j, \boldsymbol{p}_j)$ only if $\angle \boldsymbol{p}_i \boldsymbol{\alpha} \leq \angle \boldsymbol{p}_j \boldsymbol{\alpha}$. Finally, for any constraint f, we denote by $\boldsymbol{\alpha}^{\downarrow f}$ the projection of $\boldsymbol{\alpha}$ onto the subspace $\boldsymbol{H}(f)$. With these terms specified, we can now present the proposed method in Algorithm 2.

Algorithm 2: *Takes as input a minimal pyramid $\varXi$ with a sorted set of edges $\boldsymbol{E}_{\varXi}$ and a constraint f. Outputs a new minimal pyramid $\varXi'$, describing the same volume as $\varXi \cup \{f\}$, along with a sorted set of edges $\boldsymbol{E}_{\varXi'}$.*

1. Partition $\boldsymbol{E}_{\varXi}$ into two sets $\boldsymbol{E}_+$ and $\boldsymbol{E}_-$, such that each edge in $\boldsymbol{E}_+$ has angular distance to $\boldsymbol{\alpha}$ less than or equal to $\angle \boldsymbol{\alpha}\boldsymbol{\alpha}^{\downarrow f}$.
2. Move each edge $(\boldsymbol{I}, \boldsymbol{p})$ in $\boldsymbol{E}_-$, where $f(\boldsymbol{p})$, from $\boldsymbol{E}_-$ to $\boldsymbol{E}_+$
3. If $\boldsymbol{E}_- = \varnothing$ then stop and return $\varXi' = \varXi$ and $\boldsymbol{E}_{\varXi'} = \boldsymbol{E}_{\varXi}$.

4. Let the new set of constraints be
$$\boldsymbol{\Xi}' = \{f\} \cup \bigcup_{(\boldsymbol{I},\boldsymbol{p}) \in \boldsymbol{E}_+} \boldsymbol{I}\,,$$
and the set of constraints possibly defining new edges be
$$\boldsymbol{\Xi}_N = \left(\boldsymbol{\Xi}' \cap \bigcup_{(\boldsymbol{I},\boldsymbol{p}) \in \boldsymbol{E}_-} \boldsymbol{I} \right) \cup \{f\}\,.$$
Then let
$$\boldsymbol{E}_{\boldsymbol{\Xi}'} = \boldsymbol{E}_+ \cup \{(\boldsymbol{I}, \boldsymbol{p}_{\boldsymbol{I}}) \mid \boldsymbol{I} \subseteq \boldsymbol{\Xi}_N,\ f \in \boldsymbol{I},\ |\boldsymbol{I}| = m-1, \text{ and } g(\boldsymbol{p}_{\boldsymbol{I}})\ \forall g \in \boldsymbol{\Xi}'\}\,,$$
where $\boldsymbol{p}_{\boldsymbol{I}}$ is the unique point in $\mathbb{R}^m_+ \cap (\cap_{h \in \boldsymbol{I}} \boldsymbol{H}(h))$ for which $||\boldsymbol{p}_{\boldsymbol{I}}|| = 1$.
5. Return $\boldsymbol{\Xi}'$ and $\boldsymbol{E}_{\boldsymbol{\Xi}'}$.

Proposition 1. *Let $\boldsymbol{\Xi}$ constitute a minimal pyramid and f be a constraint not in $\boldsymbol{\Xi}$. Then the constraints in $\boldsymbol{\Xi} \cup \{f\}$ define the same region as those in $\boldsymbol{\Xi}'$ obtained from using Algorithm 2 to add f to $\boldsymbol{\Xi}$. Furthermore, $\boldsymbol{\Xi}'$ is a minimal pyramid.*

The complexity of inserting an element in the sorted set $\boldsymbol{E}_{\boldsymbol{\Xi}}$ is $O(\log|\boldsymbol{E}_{\boldsymbol{\Xi}}|)$, and we have a maximum of $\binom{|\boldsymbol{\Xi}|}{m-2}$ new edges to add to $\boldsymbol{E}_{\boldsymbol{\Xi}}$ in Step 4, in effect yielding a worst-case complexity of $O(\binom{|\boldsymbol{\Xi}|}{m-2}\log|\boldsymbol{E}_{\boldsymbol{\Xi}}|)$ for insertion of a constraint. This is an improvement with respect to the complexity of $(|\boldsymbol{\Xi}|+1)m^{|\boldsymbol{\Xi}|+1}$, offered by using simplex repeatedly [6], especially when the dimension m is low.

6 Presenting the Support to a Human DM

Given that $\boldsymbol{\Xi}$ has been identified as in Fig. 5(a), we provide two compact abstractions, which are useful for presenting the support to a DM.[6]
An immediate approach to representing $\boldsymbol{\Xi}$ 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 constraints in $\boldsymbol{\Xi}$ (see Fig. 5(b)).
The approach allows for a highly compact representation of $\boldsymbol{\Xi}$ during computation too: 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 abstraction, as seen in Fig. 5(b).
Another, more accurate, representation technique is to present $\boldsymbol{\Xi}$ by the

[6] These techniques do not presuppose that $\boldsymbol{\Xi}$ is minimal.

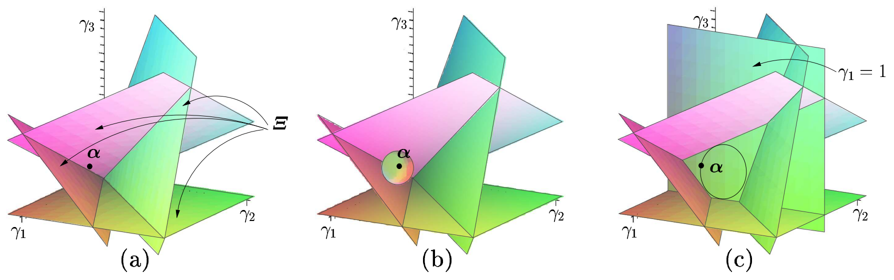

Fig. 5. *Three ways of representing the result of support analysis*

largest ball that will fit into the support defined by it. That is, abandon $\boldsymbol{\alpha}$ as a fixed center of the representation. As $\boldsymbol{\Xi}$ describes an infinite pyramid, no such largest ball exists, though, so we propose to make a cut through the computed pyramid $\boldsymbol{\Xi}$, 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 Fig. 5(c)). 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 in $\boldsymbol{\Xi}$. This is equivalent to that there exists no f in $\boldsymbol{\Xi}$, 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 $\boldsymbol{\Xi}$. If no such i exists, additional linear constraints on parameters will have to be put into $\boldsymbol{\Xi}$, e.g. $\gamma_i < k$ for some currency i and positive constant k.

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

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

where $\boldsymbol{y}$ is in $\mathbb{R}^{m-1}$ and $\mathrm{dist}(\boldsymbol{y}, \boldsymbol{H}(f^i))$ 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 corresponding 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. Furthermore, we are grateful for access to the Hugin software, which has been used in the implementation of the algorithms described in the paper. The implementation can be downloaded from `http://www.cs.aau.dk/~holbech/mcids.tgz`

References

1. Magnus Boman, Paul Davidsson, and Håkan L. Younes. Artificial decision making under uncertainty in intelligent buildings. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 65–70, 1999.
2. Gregory F. Cooper. A method for using belief netwoks as influence diagrams. In *Proceedings of the Fourth Conference on Uncertainty in Artificial Intelligence*, pages 55–63, 1988.
3. James C. Felli and Gordon B. Hazen. Do sensitivity analysis really capture problem sensitivity? an empirical analysis based on information value. *Risk, Decision and Policy*, 4(2):79–98, 1999.
4. Ronald A. Howard and James E. Matheson. Influence diagrams. *Readings on the Principles and Applications of Decision Analysis*, pages 720–763, 1984.
5. Frank Jensen, Finn V. Jensen, and Søren L. Dittmer. From influence diagrams to junction trees. In R. Lopez de Mantaras and D. Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 367–373. Morgan Kaufmann, 1994.
6. V. Klee and G. J. Minty. How good is the simplex algorithm? In *Inequalities III*, pages 159–175. Academic Press Inc., 1972.
7. Anders L. Madsen and Finn V. Jensen. Lazy evaluation of symmetric Bayesian decision problems. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 382–390, 1999.
8. Thomas Dyhre Nielsen and Finn Verner Nielsen. Sensitivity analysis in influence diagrams. *IEEE Transactions on Systems, Man, and Cybernetics*, 33(2):223–234, 2003.
9. Ross D. Shachter. Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams. In G. F. Cooper and S. Moral, editors, *Uncertainty in Artificial Intelligence: Proceedings of the Fourteenth Conference*, pages 480–487. Morgan Kaufmann.
10. Ross D. Shachter. Evaluating influence diagrams. *Operations Research*, 34:871–882, 1986.
11. Prakash P. Shenoy. Valuation-based systems for Bayesian decision analysis. *Operations Research*, 40:463–484, 1991.
12. Claus Skaaning. A knowledge acquisition tool for Bayesian-network troubleshooters. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 549–557, 2000.
13. Joseph A. Tatman and Ross D. Shachter. Dynamic programming and influence diagrams. *IEEE Transactions on Systems Management and Cybernetics*, 20:265–279, 1990.

Index