Algorithm 1: ELIM-MOID

Data: A MOID \( (X, D, P, U) \) with \( p > 1 \) objectives, a legal elimination ordering of the variables \( \tau = (Y_1, \ldots, Y_t) \)

Result: An optimal policy \( \Delta \)

\[ \text{// partition the functions into buckets} \]
\[ \text{// for } l = t \text{ down to } 1 \text{ do} \]
\[ \text{place in } \text{buckets}[l] \text{ all remaining components in } P \text{ and } U \text{ that contain variable } Y_l \text{ in their scope} \]

\[ \text{// top-down step} \]
\[ \text{// for } l = t \text{ down to } 1 \text{ do} \]
\[ \text{let } \Phi^l = \{\phi_1, \ldots, \phi_i\} \text{ and } \Psi^l = \{\psi_1, \ldots, \psi_k\} \text{ be the probability and utility components in } \text{buckets}[l] \]
\[ \text{if } Y_l \text{ is a chance variable then} \]
\[ \phi^l \leftarrow \sum_{i=1}^{k} \prod_{i=1}^{l} \phi_i \]
\[ \psi^l \leftarrow (\phi^l)^{1-1} \times \sum_{i=1}^{k} (\prod_{i=1}^{l} \phi_i) \times (\sum_{j=1}^{k} \psi_j) \]

\[ \text{else if } Y_l \text{ is a decision variable then} \]
\[ \phi^l \leftarrow \max_{i=1}^{k} \left( \prod_{i=1}^{l} \phi_i \times (\sum_{j=1}^{k} \psi_j) \right) \]
\[ \psi^l \leftarrow \max_{i=1}^{k} (\prod_{i=1}^{l} \phi_i) \times (\sum_{j=1}^{k} \psi_j) \]
\[ \text{place each } \phi^l \text{ and } \psi^l \text{ in the bucket of the highest-index variable in its scope} \]

\[ \text{// bottom-up step} \]
\[ \text{// for } l = 1 \text{ to } t \text{ do} \]
\[ \text{if } Y_l \text{ is a decision variable then} \]
\[ \delta_l \leftarrow \arg \max_{i=1}^{k} (\prod_{i=1}^{l} \phi_i) \times (\sum_{j=1}^{k} \psi_j) \]
\[ \Delta \leftarrow \Delta \cup \delta_l \]

\[ \text{return } \Delta \]

compute the set of maximal values of expected utility, up to equivalence (see also [5] for more details).

3.4 VARIABLE ELIMINATION

As well as operation + on sets of utilities, we define operation \( +' \) on finite sets of utility values by \( U +' V = \max_{\phi} (U + V) \). Theorem 1 allows us to apply an iterative variable elimination procedure along a legal elimination ordering where chance variables are eliminated by \( +' \), decision variables by \( \max \), and the probability and (set-valued) utility functions are combined by \( \times \) and \(+\), respectively. The set of maximal expected utility values is equivalent to
\[ \sum_{i=0}^{m} \max_{D_1} \cdots \max_{D_m} \sum_{i=1}^{n} \left( \prod_{i=1}^{n} P_i \times \sum_{j=1}^{r} U_j \right) \]

The variable elimination algorithm, called ELIM-MOID, is described by Algorithm 1. It is based on Dechter’s bucket elimination framework [24] and computes the maximal set \( \text{max}_{\phi} \{ EU_{\Delta} : \text{policies } \Delta \} \) as well as an optimal policy (the algorithm can be easily instrumented to produce the entire set of optimal policies). Given a legal elimination ordering \( \tau = Y_1, \ldots, Y_t \), the input functions are partitioned into a bucket structure, called buckets, such that each bucket is associated with a single variable \( Y_l \) and contains all input probability and utility functions whose highest variable in their scope is \( Y_l \).

ELIM-MOID processes each bucket, top-down from the last to the first, by a variable elimination procedure that computes new probability (denoted by \( \phi \)) and utility (denoted by \( \psi \)) components which are then placed in corresponding lower buckets (lines 3-11). For a chance variable \( Y_l \), the \( \phi \)-message is generated by multiplying all probability components in that bucket and eliminating \( Y_l \) by summation. The \( \psi \)-message is computed as the average utility in that bucket, normalized by the bucket’s compiled \( \phi \) (here \( Y_l \) is eliminated by \( \sum_{l=1}^{k} \)). For a decision variable \( Y_l \), we compute the \( \phi \) and \( \psi \) components in a similar manner and eliminate \( Y_l \) by maximization. In this case, the product of probability components in the bucket is a constant when viewed as a function of the bucket’s decision variable and therefore the compiled \( \phi \)-message is a constant as well [20, 5].

In the second, bottom-up step, the algorithm generates an optimal policy (lines 12-16). The decision buckets are processed in reversed order, from the first variable to the last. Each decision rule is generated by taking the argument of the maximization operator applied over the combination of probability and utility components in the respective bucket, for each combination of the variables in the bucket’s scope (i.e., the union of the scopes of all functions in the bucket minus \( Y_i \)) while remembering the values assigned to earlier decisions. Ties are broken uniformly at random.

As is usually the case with bucket elimination algorithms, the complexity of ELIM-MOID can be bounded exponentially (time and space) by the width of the ordered induced graph that reflects the execution of the algorithm (i.e., induced width of the legal elimination ordering) [24]. Since the utility values are vectors in \( \mathbb{R}^p \), it is not easy to predict the size of the undominated set of expected utility values.

4 APPROXIMATING THE PARETO SET

In this section, we assume without loss of generality a weak Pareto ordering on \( \mathbb{R}^p_+ \) because the proposed approximation method relies on a log transformation of the solution space as we will see next. The cardinality of the Pareto set \( \text{max}_{\phi} \{ EU_{\Delta} : \text{policies } \Delta \} \) (and also the number of optimal policies) can often get very large. What would then be desirable for the decision maker is an approximation of the Pareto set that approximately dominates (or covers) all elements in \( \{ EU_{\Delta} : \text{policies } \Delta \} \) and is of considerably smaller size. This can be achieved by considering the notion of \( \epsilon \)-covering of the Pareto set which is based on \( \epsilon \)-dominance between utility values [8].

DEFINITION 3 (\( \epsilon \)-dominance) For any finite \( \epsilon > 0 \), the \( \epsilon \)-dominance relation is defined on positive vectors of \( \mathbb{R}^p_+ \) by
\[ \bar{u} \geq \epsilon \bar{v} \Leftrightarrow (1 + \epsilon) \cdot \bar{u} \geq \bar{v} \]

DEFINITION 4 (\( \epsilon \)-covering) Let \( \mathcal{U} \subseteq \mathbb{R}^p_+ \) and \( \epsilon > 0 \). Then a set \( \mathcal{U} \subseteq \mathbb{R}^p_+ \) is called an \( \epsilon \)-approximate Pareto set or an \( \epsilon \)-covering, if any vector \( \bar{v} \in \mathcal{U} \) is \( \epsilon \)-dominated by at least one vector \( \bar{u} \in \mathcal{U} \), i.e., \( \forall \bar{v} \in \mathcal{U} \exists \bar{u} \in \mathcal{U} \) such that \( \bar{u} \geq \epsilon \bar{v} \).