New Trends in Intelligent Software Methodologies, Tools and Techniques
H. Fujita et al. (Eds.)
© 2026 The Authors.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA260726

A Global Solver for Interactive Interval Decision Analysis

Mats DANIELSON [a,b,c,1] , Xiaosong DING [d] and Xi CHEN [d]
[a] *Dept. of Computer and Systems Sciences, Stockholm University, Sweden*
[b] *International Institute for Applied Systems Analysis, Laxenburg, Austria*
[c] *School of Information Technology, Mapúa University, Makati City, Philippines*
[d] *International Business School, Beijing Foreign Studies University, P.R. China*
ORCiD ID: Mats Danielson https://orcid.org/0000-0001-6502-9670
ORCiD ID: Xiaosong Ding https://orcid.org/0000-0003-3376-5774
ORCiD ID: Xi Chen https://orcid.org/0000-0002-5803-8840

Abstract. This paper introduces a global solver tailored for interval decision analysis, a method addressing decision-making under uncertainty through the use of interval data. Decision analysis frameworks typically rely on quantitative models like decision trees and utility theory, but these approaches become computationally intensive when interval data and bilinear optimisation are introduced. The study uses the DecideIT tool as an example, which operationalises interval decision analysis by representing decision variables and outcomes as bounded intervals, enabling flexible modelling of imprecise inputs. A new computational approach, the CPM (Cutting Plane Method) solver, is presented, simplifying bilinear consistency problems into solvable linear programming tasks while maintaining computational efficiency. This ensures compatibility with interactive decision-making contexts. The paper also discusses some general challenges associated with bilinear optimisation, such as non-convexity and computational demands, and proposes tailored algorithms to address these issues. The algorithm described constitutes a significant advancement toward interactive analysis of complex decision situations.

Keywords. Interval Decision Analysis; Decision-Analytic Software; Bilinear Optimisation

1. Introduction

The field of decision analysis is a structured approach to making informed choices in situations characterised by uncertainty and complexity. Originating from operations research and statistical decision theory, decision analysis employs a combination of quantitative techniques and qualitative assessments to model decision problems and identify optimal solutions. It provides a systematic method for evaluating multiple alternatives while considering various uncertainties, trade-offs, and objectives. By formalising the decision-making process, decision analysis aids decision-makers and policymakers in all of society [1, 2].

[1] Corresponding author: mats.danielson@su.se

The theoretical foundation for decision analysis as a formalised approach to problem-solving dates back to the early 20th century, with significant contributions by mathematicians and statisticians such as von Neumann and Morgenstern, whose work laid the groundwork for utility theory and game theory [3]. These concepts later evolved through the work of pioneers such as Howard, who coined the term "decision analysis" and introduced decision trees as a key tool for structuring and solving decision problems [4]. Decision trees allow for a visual representation of decision paths, uncertainties, and their respective outcomes, providing clarity in complex decision environments. As computational power advanced in the late 20th century, some manual decision analysis frameworks transitioned into automated processes facilitated by increasingly sophisticated software. This paper is a continuation of earlier work on decision software presented at SoMeT [5–7].

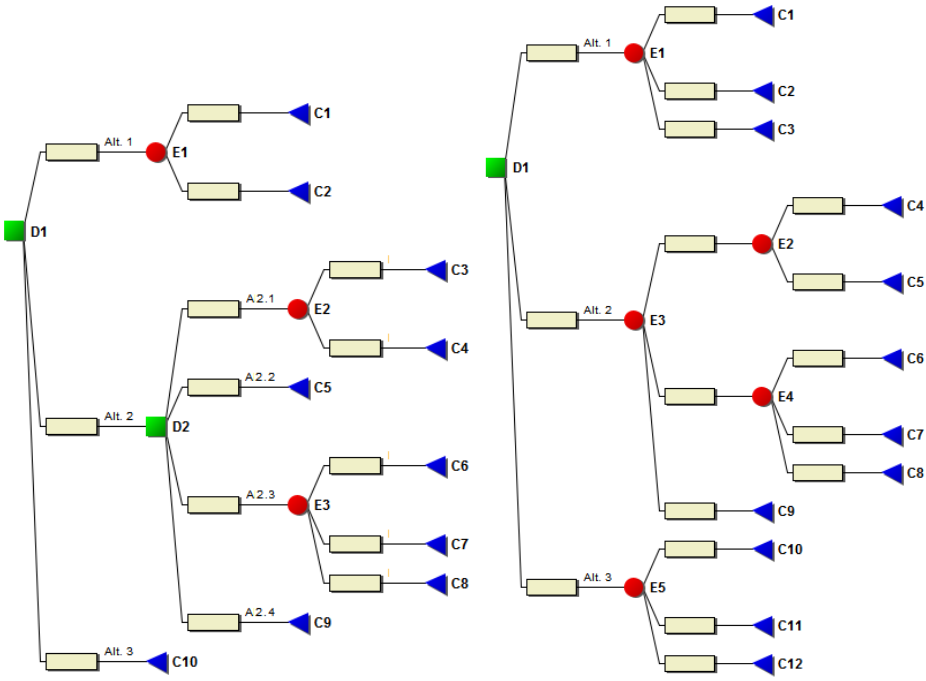
2. Decision Analytic Tools

There are several computer tools available for decision analysis. Mustajoki and Marttunen [8] provide an overview of recent tools on the market, either academic, commercial, or both. One entry found in the list is DecideIT, which will be used in this article to illustrate a tool using interval decision analysis and requiring a bilinear solver to carry out its calculations [9].

A traditional decision problem consists of a decision tree in which choices are made. Each choice is between alternatives and the choice leads either to consequences or to new choices. As shown in Figures 1 and 2, **Di** (i=1,2, ...) is the decision node; **Alt. i** (i=1,2, ...) represent alternatives; **Ei** (i=1,2, ...) represent chance nodes accompanied by their associated probability distributions; and the terminal nodes **Ci** (i=1,2, ...) represent the possible outcomes, each with a corresponding value from a utility function. Figure 1 shows a decision problem with three alternatives, one of which leads to another decision. Figure 2 also shows a decision problem with three alternatives, one of which leads to subsequent events.

The information about a decision problem is in DecideIT collected into structures called bases. A base contains the interval information for each variable, such as $p_{ij} \in [a, b]$ which is a shorthand for two inequalities $p_{ij} > a$ and $p_{ij} < b$. Further, it contains any relations between variables in the form of interval statements such as $p_{ij} - p_{km} \in [a, b]$, similarly being a shorthand for two inequalities $p_{ij} - p_{km} > a$ and $p_{ij} - p_{km} < b$. In the decision problems discussed in this paper, there are two bases, one containing probability variables and another containing value (utility) variables.

Terminology: The presentation in this article is made relative to a decision frame **F**. **F** contains m alternatives $A_1, \ldots, A_m$. Alternative A_i has m_i consequences. **F** contains a consistent probability base **P** and a consistent utility base **U**. **PU** refers to the combined base $P \cup U$. **P** contains interval statements in the variables $p_{i1}, p_{i2}, \ldots, p_{im_i}$ for all A_i and **U** contains interval statements in the variables $u_{i1}, u_{i2}, \ldots, u_{im_i}$ for all A_i.



Figures 1 and 2. Decision problem examples

Definition: Given a decision frame **F** with *m* alternatives. **F** contains a probability base **P** and a utility base **U**. The *expected utility* $E(A_i)$ of an alternative A_i is the sum $\sum_j p_{ij} \cdot u_{ij} = p_{i1} \cdot u_{i1} + p_{i2} \cdot u_{i2} + \ldots + p_{im_i} \cdot u_{im_i}$ over all consequences C_{ij} in the alternative. E_i is used as an abbreviation of $E(A_i)$.

The expected utility is well defined for any finite alternative, i.e. an alternative with a finite number of consequences. It is computationally demanding to calculate in the general case, since both the p_{ij} and the u_{ij} are variables, not numbers. When comparing two alternatives A_i and A_j, the traditional principle of maximising the expected utility (PMEU) is used [3].

Definition: Given a decision frame **F** with *m* alternatives. **F** contains a probability base **P** and a utility base **U**. Consider two alternatives A_i and A_j. The *difference* between their respective expected utilities is denoted $\delta_{ij} = E_i - E_j$. One of *i* or *j* can be 0, denoting the null alternative A_0 with $E_0 = 0$.

In order to obtain the maximal expected utility, $\max(E_i)$ should be computed for each alternative A_i and then the respective E_i are compared. That is, however, not sufficient when comparing alternatives since there can be relational statements involving more than one alternative. The solution is to compute $\max(\delta_{ij})$ instead, a significantly harder problem seen from a computational point of view and will be discussed next.

This should clarify what the computational part of an interval decision analysis primarily aims at when employing a computer tool. However, a key question remains: How can ${}^{PU}\max(\delta ij)$ be found? This leads to the research area of bilinear optimisation and we will call the problem to be solved a bilinear computation problem (BCP).

3. Optimisation

Now that we have seen the problem formulation from a typical decision-analytic tool, how can we compare the relative strengths of the different alternatives computationally? The criterion to evaluate is max(s: $\delta_{ij} > s$ is consistent with $P \cup U$) which leads to a quadratic (bilinear) consistency problem. $\delta_{ij} > s$ is a bilinear equation with all terms of the form $p_{ik} \cdot u_{ik}$. There are many such equations, each with many linear constraints to solve for each instance of a decision problem.

The theory of quadratic programming (QP) is one area within the field of non-linear optimisation. In this paper, only the top-level procedure for searching quadratic optima is discussed, without loss of generality. The general QP problem with both equalities and inequalities in the constraints is

(QPI) $\max (\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{c}^T\mathbf{x})$
when $\mathbf{A}\mathbf{x} \geq \mathbf{b}$

where $\mathbf{A}$ is a m×n matrix with linearly independent rows, $\mathbf{Q}$ is a symmetric n×n matrix, and $\mathbf{c}$ is a vector in $\mathbb{R}^n$. The expression $\mathbf{x}^T\mathbf{Q}\mathbf{x}$ is a quadratic form, and can contain all possible quadratic terms. The BCP problem maps well onto QPI since there is one second-order non-linear equation as the objective function and a larger number of linear constraints in the probability and utility bases. Even though a method similar to Simplex was originally devised by Dantzig and Wolfe to solve QP, most methods today use factorised matrices. For any given solution, the inequality problem QPI can be considered a problem with only equalities (QPE), namely all weak inequalities satisfied without slack. Since the other inequalities are not active at that solution point, they do not need to be considered locally. The QPE problems can be solved using a number of standard methods such as Lagrange methods or null-space methods, depending on matrix sparsity, stability requirements, and other criteria [10]. Still, QPE is computationally too demanding for interactivity, and QPI, being an iterative sequence of QPE's, is even more so. This leads to the fact that tools based on QP are much too slow to be considered for interactive use in decision-analytic tools.

To progress further, we employ a theorem that simplifies the task at hand a great deal. As we saw above, if the objective function is quadratic and all the constraint equations are linear, the optimising problem could be solved within the area of quadratic programming. The BCP objective function is bilinear, which is a special case of a quadratic function where most of the entries in the $\mathbf{Q}$ matrix above are zero. The following theorem offers the basis for a bilinear consistency method for solving the BCP [11].

Theorem: Given a decision frame F with m alternatives. F contains a probability base P and a utility base U. $\delta_{ij} > -t$ is a bilinear inequality in probability and utility variables. The system $PU_{ij}(t) = \{\delta_{ij} > -t\} \cup P \cup U$ can be reduced to a disjunction of linear systems $S_{ij}(t) = S_1 \vee \ldots \vee S_N$ in the probability variables. $PU_{ij}(t)$ is consistent **iff** at least one system in $S_{ij}(t)$ is consistent.

The proof in [11] is constructive and forms the theoretical basis for solving bilinear optimisation problems connected to decision tools. To see how it works, suppose there is a BCP problem in a decision frame with a probability base P and a utility base U. To determine if a particular alternative A_i is a contender, consistency must be tested after adding $\delta_{ij} > -t$ for each of the other alternatives A_j in the frame. From each system $PU_{ij}(t)$ the bilinear equation $\delta_{ij} > -t$ is eliminated using the theorem, yielding a (possibly large) set of linear systems $\{S_i\}$ in the probability variables. Each one of the S_i are then

tested for consistency using the Simplex method until either one S_i is found to be consistent or all are found to be inconsistent. This elimination is carried out for every pair (A_i, A_j). Any algorithm based on this theorem will always terminate since for each pair, the set of remaining S_i's decreases with each consistency test. Thus, algorithms for solving bilinear optimisation problems in decision analytic tools can successfully be based on linear programming components.

4. The CPM Bilinear Solver

It was long thought that a general bilinear solver would have severe time complexity, rendering it unusable for interactive decision tools. However, to remove the restrictions imposed on the utility base and model a general decision situation, it was necessary to develop another approach. The CPM (Cutting Plane Method) bilinear solver is specially designed for this purpose. It is primarily built on CPM in disjoint bilinear programming (DBLP); see for example [12–16]. Mathematically, the aforementioned decision problem can be modelled as below

(DBLP) $\min f(\boldsymbol{p}, \boldsymbol{v}) = \boldsymbol{p}^T \boldsymbol{C} \boldsymbol{v}$

s.t. $\boldsymbol{P} = \{\boldsymbol{p}: \boldsymbol{A_P p} \geq \boldsymbol{a_P}, \boldsymbol{u_P} \geq \boldsymbol{p} \geq \boldsymbol{l_P}\}$, $\boldsymbol{V} = \{\boldsymbol{v}: \boldsymbol{A_V v} \geq \boldsymbol{a_V}, \ \boldsymbol{u_V} \geq \boldsymbol{v} \geq \boldsymbol{l_V}\}$,

where $\boldsymbol{C}$, a matrix with appropriate dimensions, contains the coefficients of the cross terms; $\boldsymbol{p}$ and $\boldsymbol{v}$ are vectors representing decision variables in P- and U- bases, respectively; $\boldsymbol{A_P p} \geq \boldsymbol{a_P}$ and $\boldsymbol{A_V v} \geq \boldsymbol{a_V}$ can be those general linear constraints translated from the imprecise statements; and $\boldsymbol{u_P} \geq \boldsymbol{p} \geq \boldsymbol{l_P}$ and $\boldsymbol{u_V} \geq \boldsymbol{v} \geq \boldsymbol{l_V}$ are upper and lower bounds imposed upon $\boldsymbol{p}$ and $\boldsymbol{v}$, respectively.

DBLP can be regarded as a sub-class of non-convex quadratic programming. Its objective function is not even quasi-concave with only cross terms. As a result, we need a deterministic global optimisation approach rather than heuristics to ensure global optimality. The essential solution property of DBLP is that even though $f(\boldsymbol{p}, \boldsymbol{v})$ is not quasi-concave, its global optimiser, $(\boldsymbol{p}^*, \boldsymbol{v}^*)$, is attained at a pair of vertices of $\boldsymbol{P}$ and $\boldsymbol{V}$, respectively. In spite of this, the strategy of pure enumeration appears impractical as the number of vertices blows up as the dimension of a decision problem increases, let alone the evaluation of $f(\boldsymbol{p}, \boldsymbol{v})$ for each pair of vertices. In the literature, two main streams of solution methods to DBLP exist: CPM and B&B (Branch and Bound) methods. For an interactive decision tool, CPM seems more suitable because it can, in general, identify the actual global optimiser at early stages. This provides a good opportunity for early termination, which is essential to an interactive decision tool. By contrast, the quality of a solution given by a B&B method depends heavily on its linear relaxation, which can seldom identify the actual global optimiser at early stages. Furthermore, a B&B method may suffer from memory problems due to the fast increase in the number of nodes caused by its branching strategy.

In the presentation below, **Algorithm 1** provides a simplified CPM to DBLP, where α is the current best objective value, $\boldsymbol{V}^i$ is the feasible region in the i^{th} iteration, PGM represents a Pseudo-Global Minimizer, $\mathcal{H}^+(\overline{\boldsymbol{v}}^i)$ is a cut generated at $\overline{\boldsymbol{v}}^i$ to remove $\overline{\boldsymbol{v}}^i$ together with a portion of $\boldsymbol{V}^i$ that contains no potential optimisers possessing objective values better than α, and $\mathbf{0}$ is a zero vector with appropriate dimensions. **Algorithm 1** is simplified because it contains no auxiliary procedures to deal with open issues in DBLP, such as neighbourhood problems, singularity, degeneracy, etc.

Algorithm 1. Simplified CPM

Input: $\boldsymbol{C},\ \boldsymbol{P},\ \boldsymbol{V}$

Output: $(\boldsymbol{p}^*, \boldsymbol{v}^*),\ \alpha$

1. $i \leftarrow 0,\ \alpha \leftarrow \infty,\ (\boldsymbol{p}^*, \boldsymbol{v}^*) = (\mathbf{0}, \mathbf{0}),\ \boldsymbol{V}^0 \leftarrow \boldsymbol{V}$;
2. **while** $\boldsymbol{V}^i \neq \emptyset$ **do**
3. $\quad i \leftarrow i + 1,\ \boldsymbol{V}^i \leftarrow \boldsymbol{V}^{i-1}$;
4. $\quad$ derive a PGM, $(\overline{\boldsymbol{p}}^i, \overline{\boldsymbol{v}}^i)$, and $\bar{\alpha}$, by **Algorithm 2**;
5. $\quad$ generate a cut, $\mathcal{H}^+(\overline{\boldsymbol{v}}^i)$;
6. $\quad \boldsymbol{V}^i \leftarrow \boldsymbol{V}^i \cap \mathcal{H}^+(\overline{\boldsymbol{v}}^i)$;
7. $\quad \alpha \leftarrow \min\{\alpha, \bar{\alpha}\},\ (\boldsymbol{p}^*, \boldsymbol{v}^*) = \operatorname{argmin}\{\alpha, f(\overline{\boldsymbol{p}}^i, \overline{\boldsymbol{v}}^i) = \bar{\alpha}\}$;
8. **end**
9. **return** $(\boldsymbol{p}^*, \boldsymbol{v}^*)$ and α.

Note that in **Algorithm 1**, $\boldsymbol{P}$ has never been changed, while the introduced cuts gradually shrink V. Therefore, $\overline{\boldsymbol{p}}^i$ is simply a vertex derived in the i^{th} iteration in $\boldsymbol{P}$, whereas $\overline{\boldsymbol{v}}^i$ is a vertex derived in the i^{th} iteration in $\boldsymbol{V}^i$. Of course, we can choose to generate cuts with respect to both $\boldsymbol{P}$ and $\boldsymbol{V}$. Nevertheless, such a strategy may incur a heavy computational burden that can hardly benefit an interactive decision tool. In addition, cuts involving variables associated with both $\boldsymbol{P}$ and $\boldsymbol{V}$ may also destroy their special structures so that extant efficient solution procedures can no longer be applied [14].

Algorithm 1 is implemented through iterations. In the i^{th} iteration with its reduced feasible region, $\boldsymbol{V}^i$, **Algorithm 1** utilises in line 4 **Algorithm 2**, i.e., the Augmented Mountain Climbing Method, to derive a PGM, $(\overline{\boldsymbol{p}}^i, \overline{\boldsymbol{v}}^i)$. A PGM behaves as if it is a local optimiser in $\boldsymbol{V}^i$ and the global optimiser in $\boldsymbol{P}$. Thereafter, in the global optimisation phase, a cut will be generated at $\overline{\boldsymbol{v}}^i$ to cut off $\overline{\boldsymbol{v}}^i$ together with a portion of $\boldsymbol{V}^i$ that contains no potential optimisers possessing objective values better than α. By doing so, $\boldsymbol{V}^i$ is reduced so that **Algorithm 2** in the following iterations can never locate $\overline{\boldsymbol{v}}^i$ again in a PGM. Meanwhile, necessary updates will be carried out accordingly to α and $(\boldsymbol{p}^*, \boldsymbol{v}^*)$; see line 7. The iterations will not terminate until the entire feasible region $\boldsymbol{V}$ is exhausted.

Algorithm 2 describes the Augmented Mountain Climbing Method aiming at deriving a PGM, where $N_{\boldsymbol{V}^i}(\overline{\boldsymbol{v}})$ denotes the set of all vertices adjacent to $\overline{\boldsymbol{v}}$ in $\boldsymbol{V}^i$, and all other notations are similar to those in **Algorithm 1**.

Algorithm 2. Augmented Mountain Climbing Method

Input: $\boldsymbol{C}$, $\boldsymbol{P}$, $\boldsymbol{V}^i$

Output: $(\overline{\boldsymbol{p}}^i, \overline{\boldsymbol{v}}^i)$, $\bar{\alpha}$

1. find a feasible vertex $\widetilde{\boldsymbol{v}}^1 \in \boldsymbol{V}^i$;
2. **while** $(\widetilde{\boldsymbol{p}}^1, \widetilde{\boldsymbol{v}}^1)$ does not converge to $(\overline{\boldsymbol{p}}, \overline{\boldsymbol{v}})$ **do**
3. [a]. solve $\min\{f(\boldsymbol{p}, \widetilde{\boldsymbol{v}}^1) \mid \boldsymbol{p} \in \boldsymbol{P}\}$ to yield an optimal $\widetilde{\boldsymbol{p}}^1$;
4. [b]. solve $\min\{f(\widetilde{\boldsymbol{p}}^1, \boldsymbol{v}) \mid \boldsymbol{v} \in \boldsymbol{V}^i\}$ to yield an optimal $\widetilde{\boldsymbol{v}}^1$;
5. **end**
6. construct $N_{\boldsymbol{V}^i}(\overline{\boldsymbol{v}})$;
7. **if** there exists $\widehat{\boldsymbol{v}} \in N_{\boldsymbol{V}^i}(\overline{\boldsymbol{v}})$ such that $f(\widehat{\boldsymbol{p}}, \widehat{\boldsymbol{v}}) = \min_{\boldsymbol{p} \in \boldsymbol{P}} f(\boldsymbol{p}, \widehat{\boldsymbol{v}}) < f(\overline{\boldsymbol{p}}, \overline{\boldsymbol{v}})$
8. go to line 4 with $\widetilde{\boldsymbol{p}}^1 \leftarrow \widehat{\boldsymbol{p}}$;
9. **else**
10. **return** with $(\overline{\boldsymbol{p}}^i, \overline{\boldsymbol{v}}^i) = (\overline{\boldsymbol{p}}, \overline{\boldsymbol{v}})$, and $\bar{\alpha} = f(\overline{\boldsymbol{p}}, \overline{\boldsymbol{v}})$.

As $f(\boldsymbol{p}, \boldsymbol{v})$ is not even quasi-concave, a cut developed from a local star minimiser cannot take effect as for those programs with quasi-concave objectives, thus demanding a traditional local minimiser. Nevertheless, analogous to the idea suggested by a local star minimiser, in addition to a traditional local minimiser, a local optimisation algorithm should examine those vertices adjacent to $\overline{\boldsymbol{v}}$ in search of a better candidate; see line 7. Therefore, the entire procedure of **Algorithm 2** is first to try to derive a local minimiser $(\overline{\boldsymbol{p}}, \overline{\boldsymbol{v}})$ and then try to identify a better candidate among $N_{\boldsymbol{V}^i}(\overline{\boldsymbol{v}})$. Such a process will not stop until there exists no room for further improvement.

Generally speaking, **Algorithm 2** is rather efficient in the sense that it can locate the global optimiser at very early stages. A caveat is that **Algorithm 2** cannot determine its global optimality before $\boldsymbol{V}$ is exhausted. This still provides an interactive decision tool with a very good solution if it has to terminate prematurely due to time constraints. Besides, there may exist solutions to fathoming the reduced feasible region. For example, we can realise this by incorporating the lower bounding technique in some B&B methods [17–19] so that the algorithm is able to utilise the advantages from both approaches.

5. Performance

The performance of the CPM solver is compared to a straightforward bilinear solver (called M4P1) that finds optima by means of iterating LP solvers back and forth between the two sets of variables in the bilinear objective function until convergence, i.e. when the value of the objective function stops changing. Optima of the kind ${}^{\mathrm{PU}}\max(\mathrm{E}_1)$ are always global also by M4P1 but for ${}^{\mathrm{PU}}\max(\delta_{12})$ only CPM can guarantee optimality. M4P1 ensures that a local optimum is found, but not a global one. For the performance comparison, a large number of decision situations were generated by Monte Carlo methods. The situations were of five different categories corresponding to typical problem sizes in interactive interval decision analysis. Table 1 shows the different problem sizes in terms of number of variables in each alternative and the number of comparison between probability and value variables respectively.

Type	1	2	3	4	5
Variables	3–4	5–7	8–15	15–30	25–40
Prob. cmp.	3–6	4–10	6–12	9–20	12–30
Value cmp.	4–9	8–15	12–20	15–30	20–40

Table 1.

For each problem set type, a large number of different decision situations were generated and then solved both by CPM and M4P1. Table 2 shows for each problem type the percentage of sets that ended in a local optimum by the M4P1 solver and thus the CPM solver was needed to ensure globality. The last column contains the number of equation systems solved by the both solvers.

Type	1	2	3	4	5
M4P1 local	7.5%	7.1%	7.1%	5.2%	3.7%
Total solved	344,000	225,000	130,000	28,500	7,000

Table 2.

In Table 2, it can be seen that the rate of decision problems that require CPM to guarantee optimality is fairly constant for smaller problems and start decreasing for medium-sized problems. Especially problem types 3–5 are very typical sizes in interactive interval decision analysis. As can be seen, the CPM solver is a considerable improvement over traditional solvers of the type that M4P1 represents since error rates on the order of 4–7% can be fully avoided. CPM is, to the authors' knowledge, the first bilinear solver with these interactive properties.

There is, of course, a time penalty for using CPM compared to M4P1 and similar naïve solvers. The penalty is at an order of magnitude which is still acceptable for small- and medium-sized decision problems. For very large problems, solvers like M4P1 still have to be used, or CPM has to be prematurely terminated, but there are two reasons this is not too problematic. First, large problems (with hundreds of variables), usually have such a structure that smaller optimisation approximations tend to cancel out. Second, the impact of a variable stuck in a non-optimal position is much lower in very large decision problems. From testing, it is feasible to strike a balance where CPM takes care of most problems in an exact way and M4P1 deals with very large problems to stay interactive. Comparing the solutions of PCM and M4P1 in retrospect for large problems show that the impact of local optima is comparatively much less and can be considered reasonable approximations.

6. Conclusion

The study explores advancements in interval decision analysis, focusing on computational frameworks that enable decision-making under uncertainty. This approach models decision variables and outcomes as bounded intervals rather than fixed values, accommodating imprecise and incomplete data. The discussion uses the DecideIT tool, which exemplifies how interval decision analysis can support complex decision-making scenarios through bilinear optimisation techniques. Traditional models like decision trees

and utility theory are foundational, but interval decision analysis extends their applicability by representing uncertainty more flexibly.

The article identifies the challenges of bilinear optimisation, notably the non-convexity of bilinear problems, which complicates computational efficiency. To address this, the authors propose the CPM method, which converts bilinear consistency problems into manageable linear programming tasks. This approach leverages the proportional contraction of intervals and introduces a theorem that simplifies bilinear problems into sets of linear constraints. Additionally, the study highlights methods to evaluate and compare alternative decisions, ensuring consistency and reliability across varied scenarios. The ability to integrate interval data into decision-making provides stakeholders with a robust framework for evaluating trade-offs, assessing uncertainties, and optimising outcomes. By combining the M4P1 solver with the CPM solver, the choice being made based on whether there are statements involving more than one alternative in the utility base, a hierarchy that caters both to computational speed and that globality is achieved.

Acknowledgement. This work was supported by the Chinese Fundamental Research Funds for the Central Universities (2025ZZ009).

References

[1] Danielson, M. Computational Decision Analysis, Ph.D. thesis, Dept. of Computer and Systems Sciences, Royal Institute of technology, 1997.
[2] Danielson, M. Foundations of Computational Decision Analysis, 2nd ed., Sine Metu, 2025.
[3] Danielson, M., Ekenberg, L. Real-Life Decision-Making, 1st ed., CRC Press, 2023.
[4] Howard, R. A. Decision analysis: Applied decision theory. In Proceedings of the Fourth International Conference on Operational Research, 1966.
[5] Danielson M, Ekenberg L. Development of algorithms for decision analysis with interval information. In: Czachórski T, Gelenbe E, Lent R, editors. Proceedings of the 8th International Conference on Software Methods and Tools (SOMET 2009); 2009 Sep 14–16; Istanbul, Turkey. Springer, Lecture Notes in Computer Science, 2009.
[6] Danielson M, Ekenberg L. Development of software for decision analysis. In: Czachórski T, Gelenbe E, Lent R, editors. Proceedings of the 11th International Conference on Software Methods and Tools (SOMET 2013); 2013 Sep 23–25; Budapest, Hungary. Springer, Lecture Notes in Computer Science, 2013.
[7] Danielson M, Ekenberg L. Space-time trade-off in decision analysis software. In: Czachórski T, Gelenbe E, Lent R, editors. Proceedings of the 17th International Conference on Software Engineering and Formal Methods (SOMET 2018); 2018 Sep 17–19; Rome, Italy. Cham: Springer; 2018.
[8] Mustajoki, J., Marttunen, M. Comparison of multi-criteria decision analytical software for supporting environmental planning processes, Environmental Modelling & Software 93, 78–91, 2017.
[9] Larsson, A., Riabacke, M., Danielson, M.., Ekenberg, L. Cardinal and Rank Ordering of Criteria – Addressing Prescription within Weight Elicitation. International Journal of Information Technology & Decision Making 14:06, 1299–1330, 2015.
[10] Luenberger, D. G., Ye, Y. Linear and Nonlinear Programming, 3rd ed., Springer, 2008.
[11] Malmnäs, P-E.: Real-Life Decisions, Expected Utility, and Effective Computability, Research Report HSFR no.677/87, 1990.
[12] Konno, H., A cutting plane algorithm for solving bilinear programs. Mathematical Programming, 11(1):14–27, 1976.
[13] Porembski, M. How to extend the concept of convexity cuts to derive deeper cutting planes. Journal of Global Optimization, 15(4):371–404, 1999.
[14] Shetty, C.M., Sherali, H.D, 1980. Rectilinear distance location-allocation problem: A simplex based algorithm. Proc. of the Internat. Symp. on Extremal Methods and Systems Analysis, Lecture Notes in Economics and Math. Systems, Springer, 1980.
[15] Tuy, H. Concave programming under linear constraints. Soviet Mathematics, 5:1437–1440, 1964.
[16] Tuy, H. On a decomposition method for nonconvex global optimisation. Optimisation Letters, 1:245–258, 2007.

[17] Al-Khayyal, F. and Falk, J. Jointly constrained biconvex programming. Mathematics of Operations Research, 8(2):273–286, 1983.
[18] Audet, C., Hansen, P., Jaumard, B., and Savard, G. A symmetrical linear maxmin approach to disjoint bilinear programming. Mathematical Programming, 85(3):573–592, 1999.
[19] Sherali, H. and Alameddine, A. A new reformulation-linearisation technique for bilinear programming problems. Journal of Global Optimization, 2:379–410, 1992.