
White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of Leeds.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/246982/

Version: Accepted Version

Article:
Chakhar, S., Panaro, S., Greco, S. et al. (Accepted: 2026) A novel decision graph model to visualise dominance based if-then decision rules. Omega: The International Journal of Management Science. ISSN: 0305-0483 (In Press)

This is an author produced version of an article accepted for publication in Omega, made available via the University of Leeds Research Outputs Policy under the terms of the Creative Commons Attribution License (CC-BY), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited.

Reuse
This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here:
https://creativecommons.org/licenses/

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.





Revised manuscript (unmarked, editable source file)

A novel decision graph model to visualise dominance based if-then decision rules

Salem Chakhar[a,b,c,*], Simona Panaro[d], Salvatore Greco[e], Alessio Ishizaka[f], Sajid Siraj[g]

[a] *Portsmouth Business School, University of Portsmouth, Portsmouth, UK.*
[b] *Centre for Operational Research & Logistics, University of Portsmouth, Portsmouth, UK.*
[c] *Responsible AI Decision Lab, University of Portsmouth, Portsmouth, UK.*
[d] *Department of Architecture and Industrial Design, University of Campania Luigi Vanvitelli, Aversa, Italy.*
[e] *Faculty of Economics, University of Catania, Catania, Italy.*
[f] *NEOMA Business School, Mont-Saint-Aignan, France.*
[g] *Leeds University Business School, University of Leeds, UK.*

Abstract

When decision makers rely on large sets of if-then decision rules to guide complex choices, information overload quickly becomes a barrier to effective use. The Dominance-based Rough Set Approach (DRSA) is a widely used preference learning method whose primary output is a set of if-then decision rules incorporating preference orders over attributes and decision classes. Although effective for knowledge representation and classification, large rule sets can overwhelm decision makers and hinder interpretation. To address this challenge, this paper proposes a new decision graph model for visualising dominance based if-then decision rules. The construction process begins with a preprocessing phase that extracts a binary decision matrix from rule descriptions. To avoid generating nonexistent rules, this matrix may require transformation through column splitting or a combination of column splitting and ordering-based strategies. The transformed matrix is then used to build a directed acyclic decision graph in which nodes represent elementary conditions and decisions, and edges denote their logical relationships. Each path from the root to a decision node corresponds to a complete decision rule. The paper introduces the core concepts of decision graphs and presents algorithms for their construction, linearisation, and exploitation. Both linearisation and exploitation algorithms are based on the Breadth-First Search graph traversal algorithm. The approach is demonstrated using real-world cultural heritage data from Europe, and expert feedback indicates that decision graphs reduce cognitive effort, improve communication, and provide a clear and accessible complement to traditional textual and tabular representations of decision rules.

Keywords: Preference learning, Dominance-based rough set approach, Decision rules, Decision graph, Graphic representation

1. Introduction

The Dominance-based Rough Set Approach (DRSA) (Greco et al., 1999, 2001a; Slowiński et al., 2002, 2009) is a powerful preference learning method. Unlike the traditional Indiscernibility-based Rough Set Approach (IRSA) (Pawlak, 1982, 1991), DRSA incorporates preference orders within attribute and decision-class domains. It offers a key advantage over traditional machine learning through improved explainability and interpretability, addressing limitations caused by poor explainability (Sekulić & Strube, 2019) and model complexity (Wu & Zhu, 2025). By generating explicit and interpretable decision rules (Blaszczyński et al., 2012a), DRSA enables transparent justification of outcomes, which is particularly valuable in high-stakes domains such as cultural heritage, urban regeneration, and adaptive reuse of historic buildings. Although deep learning models may provide greater explainability than traditional techniques, they require large datasets (Pananookooln et al., 2023), remain opaque (Aguilera et al., 2021), and demand substantial computational resources. Beyond interpretability, DRSA avoids preference-parameter specification, reduces cognitive burden, and produces simple if–then rules accessible to non-experts (Blaszczyński et al., 2012a). It also handles missing data (Blaszczyński et al., 2012b; Slowiński et al., 2002) and detects inconsistencies (Deng et al., 2013; Slowiński et al., 2000), which is important in noisy environments. While DRSA shares with other multicriteria methods the use of a learning set, it remains distinctive in providing transparent and interpretable if–then rules (Blaszczyński et al., 2012a). Additionally, DRSA has been successfully applied in domains including healthcare, industrial manufacturing, and risk assessment, demonstrating its versatility and practical relevance (Chakhar et al., 2026; Egaji et al., 2019; Hu et al., 2017; Labib et al., 2022; Legay et al., 2015; Saad & Chakhar, 2009; Younsi et al., 2020).

The main output of DRSA is a set of transparent, interpretable, and preference-consistent if–then decision rules, whose success stems from their resemblance to human reasoning and natural language. They are also the only aggregation operators capable of capturing complex interactions among criteria, remaining non-compensatory, and operating directly on ordinal scales without converting them into cardinal ones (Słowiński et al., 2002). Each rule represents a causal scenario linking evaluations on selected criteria to a comprehensive judgement, revealing the values driving decision makers' preferences (Słowiński et al., 2002). Depending on whether they originate from lower, upper, or boundary approximations of class unions, different rule types can be induced, each consisting of conditions and a conclusion assigning an object to a decision class. These rules are typically presented in textual or tabular form or embedded within classification algorithms; while textual and tabular formats support justification and accountability, algorithmic integration suits large datasets. However, as the number of induced rules increases—particularly in

*Corresponding author.
Email addresses: salem.chakhar@port.ac.uk (Salem Chakhar), simona.panaro@unicampania.it (Simona Panaro), salgreco@unict.it (Salvatore Greco), alessio.ishizaka@neoma-bs.fr (Alessio Ishizaka), S.Siraj@leeds.ac.uk (Sajid Siraj)

complex real-world settings with multiple criteria, large datasets, or numerous decision classes—textual or tabular presentations become difficult to navigate, causing cognitive overload. This challenge motivates alternative representations that preserve DRSA interpretability while improving usability and reducing cognitive effort.

In this context, graph-based visualisation provides an important complement to conventional textual and tabular representations. Graph models support knowledge representation and analysis by offering simple graphical representations and effective communication tools (Jimenez-Roa et al., 2021). Typical graph models include fast-and-frugal trees, fault trees, reliability block diagrams, decision trees, and binary decision diagrams. Fast-and-frugal trees provide rapid decision-making with limited complexity (Martignon et al., 2003, 2022) and have been applied in fields including psychology, artificial intelligence, management science, medical decision making, finance, supply chain management, and customer management (Banks et al., 2020; Jenny et al., 2013; Hafenbrädl et al., 2016; Keller & Katsikopoulos, 2016; Neth et al., 2014). Fault trees and reliability block diagrams support reliability engineering by analysing failure modes and component interactions (Ishizaka & Labib, 2014; Labib, 2014; Signoret & Leroy, 2021) and have been widely applied in reliability studies (Distefano & Puliafito, 2009; Ishizaka & Labib, 2014; Ruijters & Stoelinga, 2015; Signoret & Leroy, 2021; Waghen & Ouali, 2019). Decision trees are flexible machine learning models used for classification and regression (Quinlan, 1986, 1987; Rokach & Maimon, 2014), with variants including decision lists, CART, and ensemble methods such as boosted trees, bagging, random forests, and rotation forests. They have been applied across numerous domains (Díaz-Ramírez et al., 2023; Meena et al., 2024; Salazar-Serna et al., 2025). Binary Decision Diagrams (BDDs) provide compact representations of Boolean functions and are widely used in digital logic design and Boolean optimisation (Akers, 1978; Bryant, 2018), computer-aided design, Datalog interpreters, manufacturing, reliability assessment, and autonomous systems (Jensen, 2004; Lipmaa, 2010; Whaley et al., 2005; Ke et al., 2024; Brînzei & Aubry, 2018; Prescott et al., 2009). While these graph models remain widely used, recent research has expanded graph-based knowledge representation to include probabilistic, causal, and rule-based frameworks. A brief review of classical and recent graph-based models is provided in Section 2.

Most existing graph-based models can be translated into if–then decision rules, particularly fast-and-frugal trees, fault trees, and decision trees, yet none has been designed to represent dominance-based if–then rules. Although some models, especially fast-and-frugal trees and fault trees, can be adapted for this purpose, doing so requires modifying their fundamental principles and may introduce unnecessary complexity. These limitations motivate a dedicated graphical formalism for dominance-based decision rules. The decision graph model proposed in this paper addresses this gap through a directed acyclic graph where nodes represent elementary conditions or decisions and edges encode logical relationships, with each root-to-decision path corresponding to a unique rule and providing a compact and interpretable representation of the rule set. Construction begins with a preprocessing phase that derives a binary decision matrix whose rows represent decision rules and columns represent elementary conditions and decisions. To avoid nonexistent rules, the matrix may be transformed using column splitting or a combination of splitting and ordering strategies: the former increases nodes and edges but requires less computation, whereas the latter controls graph growth at the cost of additional optimisation time. The transformed matrix is then used to construct the decision graph.

This paper introduces the fundamental concepts of decision graphs together with algorithms for their construction, linearisation, and exploitation. The model follows standard graph-theoretic principles, with Breadth-First Search supporting both linearisation and exploitation. Unlike most earlier graph models, decision graphs can be directly processed using established graph algorithms, and suitable weighting schemes allow the application of various pathfinding methods. In contrast to fast-and-frugal trees, decision graphs represent complex decision rules, while unlike decision trees, they are derived from rule sets induced by learning algorithms. The main contributions are: (1) a formal decision graph model for dominance-based if–then decision rules; (2) a systematic construction methodology based on a binary decision matrix and two transformation strategies that prevent nonexistent rules while controlling complexity; (3) algorithms for construction, linearisation, and exploitation using established traversal techniques; and (4) a general framework capable of representing other rule types, including probabilistic rules from Variable Consistency DRSA (VC-DRSA) (Greco et al., 2001c), ranking and choice DRSA rules, classical Rough Set Approach rules (Pawlak, 1982, 1991), and association rules (Agrawal et al., 1993). The practical value of the model lies in reducing cognitive effort, improving communication, and complementing textual and tabular representations through a transparent and interpretable decision-support tool.

The remainder of this paper is organised as follows. Section 2 reviews existing graph-based models. Section 3 introduces the foundations, structures, and construction principles of decision graphs. Section 4 presents binary matrix extraction, transformation, and graph construction. Section 5 describes decision graph exploitation, including linearisation, assignment, and decision aggregation. Section 6 demonstrates the methodology using real-life cultural heritage data from Europe. Section 7 compares decision graphs with established graph-based models and discusses their theoretical and practical aspects. Section 8 concludes the paper and presents future research directions.

2. Overview on graph models

Visual and graphical tools play a vital role in decision support systems, enabling knowledge representation and extraction for informed decision-making. A variety of graph models have been proposed and applied across domains. Among the most prominent are fast-and-frugal trees, decision trees, fault trees, reliability block diagrams, and binary decision diagrams. Each provides a distinctive way of structuring information and supporting decision processes. A brief overview of these models, together with their main features and extensions, is presented in this section.

Fast-and-frugal trees (FFTs), introduced by Martignon et al. (2003) and further developed by Martignon et al. (2022), employ a simple graphical structure to categorise objects or make decisions through binary (yes/no) questions. Designed to be rapid and economical in information use, FFTs are valuable when decisions must be reached quickly with limited data. Nodes correspond to binary conditions, while edges represent possible outcomes leading to subsequent questions or final decisions (see Figure 1(a)).

Applications span psychology (Jenny et al., 2013), military contexts (Banks et al., 2020; Keller & Katsikopoulos, 2016), high-stakes decisions such as aviation and medicine (Hafenbrädl et al., 2016), and financial risk management (Neth et al., 2014).

Fault trees (FTAs) provide a graphical and logical representation of causal relationships among events (Ishizaka & Labib, 2014). Structured hierarchically, the top event denotes the undesirable outcome, while component failures are represented as basic events (see Figure 1(b)). Logical gates, most commonly OR and AND, define the relationships between events (Labib, 2014). An OR gate indicates that the output occurs if any input event occurs, whereas an AND gate requires all input faults to occur simultaneously. FTAs are often used as a systematic brainstorming tool to understand the pathways leading to adverse incidents, and they have been widely applied in reliability engineering (Distefano & Puliafito, 2009; Ruijters & Stoelinga, 2015; Signoret & Leroy, 2021; Waghen & Ouali, 2019).

Reliability block diagrams (RBDs), also known as dependence diagrams, illustrate how component reliability influences the overall success or failure of a system. An RBD is depicted as a set of blocks arranged in series or parallel (see Figure 1(c)). Parallel blocks represent redundancy, reducing failure rates, while series paths fail if any single component fails. Switches may also be used, with open switches denoting failed components and closed switches denoting functioning ones. When statistical independence holds, closed-form solutions can be applied; otherwise, more advanced approaches such as dynamic RBDs are required (Distefano & Puliafito, 2009). RBDs can be transformed into fault trees using mapping rules: OR gates correspond to series structures, and AND gates correspond to parallel structures. De Morgan's theorem allows conversion between success and fault trees. In practice, RBDs are frequently used alongside FTAs in reliability engineering (Signoret & Leroy, 2021).

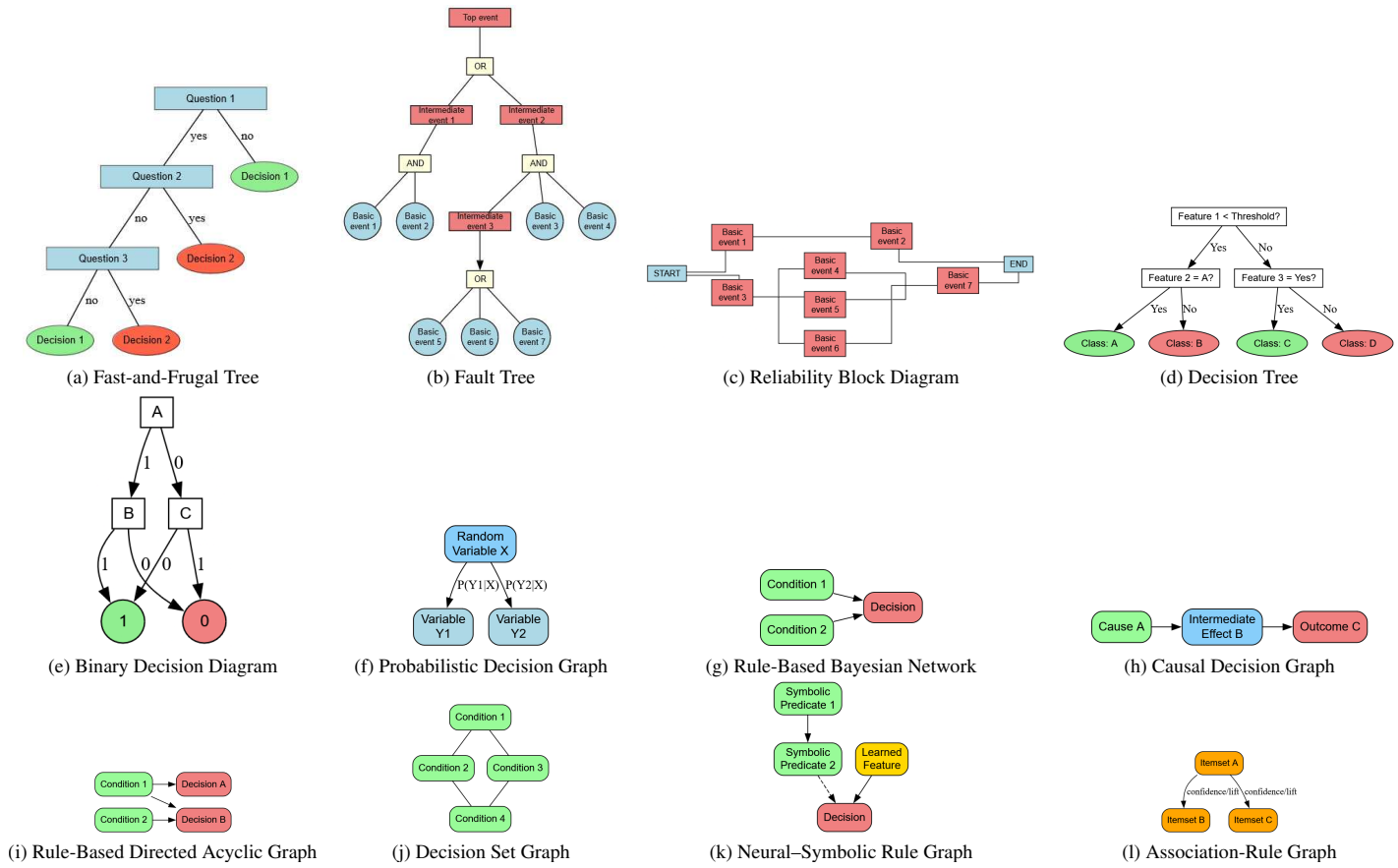
Decision trees constitute one of the most widely used predictive modelling approaches in statistics, data mining, and machine learning (Quinlan, 1986, 1987; Rokach & Maimon, 2014). They operate by mapping observations about an item, represented in the branches, to conclusions about its target value, represented in the leaves. When the target variable takes discrete values, the resulting structures are known as classification trees: leaves denote class labels, while branches represent conjunctions of features leading to those labels (see Figure 1(d)). When the target variable is continuous, the structures are referred to as regression trees. Decision trees are valued for their intelligibility and simplicity, making them popular across diverse application domains (Díaz-Ramírez et al., 2023; Meena et al., 2024; Salazar-Serna et al., 2025; Xu et al., 2023). In decision analysis, they provide a clear visual representation of choices and outcomes, while in data mining they serve as descriptive models that can subsequently inform decision-making. A decision tree can be linearised into decision rules (Quinlan, 1987), where the leaf node outcome is expressed as the conclusion and the conditions along the path form the antecedent. Numerous variants exist, ranging from decision lists (one-sided trees) to more advanced methods such as Classification and Regression Trees (CART) and ensemble approaches including boosted trees, bootstrap aggregation, random forests, and rotation forests.

Binary decision diagrams (BDDs) offer a compact data structure for representing Boolean functions (Bryant, 2018; Akers, 1978). They can be viewed as condensed abstractions of sets or relations, enabling operations to be performed directly on the compressed representation without decompression. A Boolean function is expressed as a rooted, directed, acyclic graph comprising decision nodes and two terminal nodes (see Figure 1(e)). The terminal nodes are labelled 0 (FALSE) and 1 (TRUE). Each decision node u is associated with a Boolean variable x_i and has two child nodes: a low child and a high child. The edge to the low child corresponds to assigning FALSE to x_i, while the edge to the high child corresponds to assigning TRUE. An ordered BDD requires that variables appear in the same order along all paths from the root, while a reduced BDD applies two simplification rules: merging isomorphic subgraphs and eliminating nodes whose children are identical. A path from the root to the 1-terminal represents a variable assignment for which the Boolean function evaluates to true. BDDs have found extensive application in computer-aided design (Jensen, 2004; Lipmaa, 2010), datalog interpreters (Whaley et al., 2005), manufacturing (Ke et al., 2024), reliability assessment (Brînzei & Aubry, 2018), and autonomous systems (Prescott et al., 2009).

Recent advances in graph-based knowledge representation have led to several frameworks for representing dependencies, rules, and decision structures. Probabilistic graphical models, including Probabilistic Decision Graphs (PDGs) and Rule-Based Bayesian Networks (RBBNs), have been widely applied in medical diagnosis, risk assessment, and decision support (Bozga & Maler, 1999; Heckerman, 1998; Jaeger, 2004; Nielsen et al., 2012). PDGs provide compact graph-based representations of probabilistic dependencies and encode probabilistic if-then relationships between variables (see Figure 1(f)). RBBNs extend Bayesian Networks (Heckerman, 1998) by incorporating symbolic rules into conditional probability tables, enabling probabilistic inference over rule-based knowledge (Friedman et al., 1997) (see Figure 1(g)).

Other non-probabilistic graph-based models have been proposed for representing decision rules and structured decision knowledge, including Causal Decision Graphs, Rule-Based Directed Acyclic Graphs, Decision Set Graphs, Neural-Symbolic Rule Graphs, and Association-Rule Graphs. Causal Decision Graphs represent causal if-then relationships using directed acyclic structures derived from causal rules or structural equations, supporting causal inference and intervention analysis (Pearl, 2013; Spirtes et al., 2001) (see Figure 1(h)). Rule-Based Directed Acyclic Graphs compactly represent overlapping deterministic rule sets by mapping shared conditions to internal nodes and decisions to terminal nodes (Byeon & Lee, 2023) (see Figure 1(i)). Decision Set Graphs represent unordered rule collections and highlight interactions between conditions (Lakkaraju et al., 2016) (see Figure 1(j)). Neural-Symbolic Rule Graphs combine symbolic rules with learned representations to support hybrid reasoning (see Figure 1(k)), while Association-Rule Graphs represent relationships between itemsets through weighted edges and support analysis of support, confidence, and lift (Bhuyan et al., 2024; Agrawal et al., 1993) (see Figure 1(l)).

Graphical representations are valuable for modelling complex decision problems, requiring minimal cognitive effort and little or no programming. Several software tools have been developed, including KNIME (https://www.knime.com/), VisiRule (https://www.visirule.co.uk/), and the FFTrees toolbox (Phillips et al., 2017), which supports the creation and evaluation of fast-and-frugal trees. Other visual workflow platforms include RapidMiner (https://www.rapidminer.com/) (Hofmann & Klinkenberg, 2014), Orange Data Mining (https://orangedatamining.com/) (Demšar et al., 2013), Weka (https://www.cs.waikato.ac.nz/ml/weka/) (Hall et al., 2009), Node-RED (https://nodered.org/), and IBM SPSS Modeler (https://www.ibm.com/products/spss-modeler).



(a) Fast-and-Frugal Tree (b) Fault Tree (c) Reliability Block Diagram (d) Decision Tree

(e) Binary Decision Diagram (f) Probabilistic Decision Graph (g) Rule-Based Bayesian Network (h) Causal Decision Graph

(i) Rule-Based Directed Acyclic Graph (j) Decision Set Graph (k) Neural–Symbolic Rule Graph (l) Association-Rule Graph

Figure 1: Illustration of graph models

These tools allow users to construct analytical workflows or decision models by connecting graphical components, supporting visual exploration and interpretation of complex data and decisions.

3. Foundations and structural principles of decision graphs

This section introduces the foundational notions necessary for understanding decision graphs, beginning with the dominance-based rough approximation and its formal structure (Section 3.1). It then presents the syntax and semantics of dominance-based decision rules (Section 3.2) and the definition and structure of decision graphs themselves (Section 3.3). Finally, the principles guiding the construction and exploitation of decision graphs are discussed (Section 3.4).

3.1. Dominance-based rough approximation

A brief overview of the dominance-based rough approximation is provided here, while a more formal presentation is given in Appendix A. Further information can be found in the relevant literature (Greco et al., 1999, 2001a; Slowiński et al., 2002, 2009). Let U be a set of decision objects described with respect to a set Q of preference-ordered attributes. The set Q is divided into two non-empty subsets: a set A of condition attributes (or criteria) and a set D of decision attributes, such that $A \cup D = Q$ and $A \cap D = \emptyset$. The decision part is assumed to consist of a single attribute, denoted d. This attribute induces a partition of U into a finite number of preference-ordered decision classes, written as $Cl = Cl_t : t \in T$, where each object in U belongs to exactly one class. The classes are preference-ordered according to an increasing order of class indices, i.e. for all $r,s \in T$ such that $r \geq s$, the objects from Cl_r are strictly preferred to the objects from Cl_s. For each attribute $q \in Q$, let V_q denote its domain, and let V be the union of all such domains. An information function $f : U \times Q \to V$ assigns to every object-attribute pair a value from the corresponding domain. The quadruple $S = \langle U,Q,V,f \rangle$ is referred to as a decision table.

The central idea of rough set theory is to approximate one type of knowledge using another. In the dominance-based rough approximation, the knowledge to be approximated consists of the upward and downward unions of decision classes, defined as:

$$Cl_t^{\geq} = \bigcup_{s \geq t} Cl_s, \qquad t \in T, \tag{1}$$

$$Cl_t^{\leq} = \bigcup_{s \leq t} Cl_s, \qquad t \in T. \tag{2}$$

Upward unions group all classes at least as good as a given class, whereas downward unions group all classes no better than a given class.

The knowledge used for rough approximation of these unions consists of dominating and dominated sets, defined using the dominance relation Δ associated with the condition attributes A. For each pair of objects x and y:

$$x\Delta y \;\Leftrightarrow\; f(x,q) \propto f(y,q), \qquad \forall q \in A, \tag{3}$$

where $\propto$ corresponds to the comparison operator "$\geq$" for cardinal attributes with increasing preference, "$\geq$" for ordinal attributes with increasing preference, or "=" for nominal attributes. For attributes with decreasing preference, the operator becomes "$\leq$" (cardinal) or "$\trianglelefteq$" (ordinal).

The dominating set $\Delta^+(x)$ and the dominated set $\Delta^-(x)$ of an object $x \in U$ are defined as: $\Delta^+(x) = \{y \in U : y\Delta x\}$, and $\Delta^-(x) = \{y \in U : x\Delta y\}$. These dominance-based sets provide the foundation for constructing rough approximations of class unions. Each class union $Cl_t^{\geq}$ (resp. $Cl_t^{\leq}$) is characterised by two fundamental rough approximations:

- **Lower approximation** $\underline{A}(Cl_t^{\geq})$ (resp. $\underline{A}(Cl_t^{\leq})$): the set of objects whose dominating or dominated sets (depending on the union) are entirely contained within the class union. These objects certainly belong to the union.
- **Upper approximation** $\overline{A}(Cl_t^{\geq})$ (resp. $\overline{A}(Cl_t^{\leq})$): the set of objects whose dominating or dominated sets have a non-empty intersection with the class union. These objects may belong to the union.

The **boundary region** $Bn(Cl_t^{\geq})$ (resp. $Bn(Cl_t^{\leq})$) is defined as the difference between the upper and lower approximations. It contains all objects whose membership in the class union is indeterminate, as they cannot be assigned with certainty to the union nor conclusively excluded from it.

3.2. Syntax and semantics of dominance-based decision rules

Decision rule induction algorithms can be categorised into three categories: (i) algorithms inducing a minimum set of non-redundant decision rules; (ii) algorithms inducing an exhaustive set of decision rules; and (iii) algorithms inducing a satisfactory set of decision rules that satisfy specific user requirements. The most popular rule induction algorithm for DRSA is DOMLEM (DOMinance-based Learning from Examples Module) (Greco et al., 2001c), which generates a minimal set of rules, together with its successor, the VC-DomLEM algorithm (Błaszczyński et al., 2011), a sequential covering rule induction algorithm proposed for DRSA and VC-DRSA. The Variable Consistency DRSA (VC-DRSA) (Greco et al., 2001b) is a variant of DRSA that relaxes the conditions for assigning objects to lower approximations by allowing a limited proportion of negative examples, which is particularly useful for large decision tables.

Having outlined the main induction procedures, we now turn to the structure of the decision rules they produce. Let $A = \{q_k \mid k = 1,...,p\}$ and $r_{q_k} \in V_{q_k}$ for $k = 1,...,p$. Five basic types of decision rules can be induced from dominance-relation-based rough approximations:

- certain decision rules supported by objects form the lower approximation of $Cl_t^{\geq}$:
  if $f(x,q_1) \geq r_{q_1}$ and $f(x,q_2) \geq r_{q_2}$ and ... and $f(x,q_p) \geq r_{q_p}$, then $x \in Cl_t^{\geq}$ **(type 1)**
- possible decision rules supported by objects from the upper approximations of $Cl_t^{\geq}$:
  if $f(x,q_1) \geq r_{q_1}$ and $f(x,q_2) \geq r_{q_2}$ and ... and $f(x,q_p) \geq r_{q_p}$, then possibly $x \in Cl_t^{\geq}$ **(type 2)**
- certain decision rules supported by objects from the lower approximations of $Cl_t^{\leq}$:
  if $f(x,q_1) \leq r_{q_1}$ and $f(x,q_2) \leq r_{q_2}$ and ... and $f(x,q_p) \leq r_{q_p}$, then $x \in Cl_t^{\leq}$ **(type 3)**
- possible decision rules supported by objects from the upper approximation of $Cl_t^{\leq}$:
  if $f(x,q_1) \leq r_{q_1}$ and $f(x,q_2) \leq r_{q_2}$ and ... and $f(x,q_p) \leq r_{q_p}$, then possibly $x \in Cl_t^{\leq}$ **(type 4)**
- approximate decision rules supported by objects from $Cl_s \cup Cl_{s+1} \cup ... \cup Cl_t$ without possibility of discerning to which class:
  if $f(x,q_1) \geq r_{q_1}$ and $f(x,q_2) \geq r_{q_2}$ and ... and $f(x,q_k) \geq r_{q_k}$ and $f(x,q_{k+1}) \leq r_{q_{k+1}}$ and $f(x,q_{k+2}) \leq r_{q_{k+2}}$ and ... and $f(x,q_p) \leq r_{q_p}$, then $x \in Cl_s \cup Cl_{s+1} \cup ... \cup Cl_t$ **(type 5)**

Decision rules obtained from VC-DRSA have the same syntax plus an associated consistency level. The decision rules within the choice and ranking versions of the DRSA (Greco et al., 2001a) have slightly different syntax as elementary condition represent pairs of objects x and y evaluated on a given criterion and the decision part is of the form xSy or xS^cy where S or S^c are the outranking relation and non-outranking relation, respectively. More details on the syntax of decision rules can be found in e.g. Greco et al. (2016).

The quality of decision rules can be assessed through a series of basic measures, namely support (the number of objects/pairs that support the rule), strength (defined as the ratio of the support to the number of objects/pairs), confidence (defined as the ratio of support and the number of objects or pairs of objects that satisfy all elementary conditions of the rules) and coverage factor (defined as the ratio of support and the number of objects or pairs of objects that satisfy the decision part of the rule). More details on decision rule characterisation and attractiveness measures can be found in relevant literature (Atteya et al., 2024; Chakhar et al., 2020; Greco et al., 2004, 2012; Polkowski & Artiemjew, 2015; Stefanowski & Vanderpooten, 2001).

3.3. Definition and structure of decision graphs

Let $S = \langle U,Q,V,f \rangle$ be a decision table defined as in Section 3.1. Let $R = \{R_i \mid i = 1,...,|R|\}$ be a set of decision rules induced from S, and let $A(R_i)$ and $D(R_i)$ denote the condition and decision parts of rule $R_i \in R$, respectively. Let also $C = \{C_j \mid j = 1,...,|C|\}$ and $K = \{K_t \mid t = 1,...,|K|\}$ be non-empty, non-redundant sets of elementary conditions and decisions included in the condition and decision parts of rules in R, respectively.

Definition 1 (Decision rule)**.** *A decision rule $R_i \in R$ is of the form*

$$C_1 \wedge C_2 \wedge ... \wedge C_n \quad \rightarrow \quad K_t, \tag{4}$$

where $C_1 \wedge ... \wedge C_n = A(R_i)$ is the condition part, and $K_t = D(R_i)$ is the decision part.

The condition part $A(R_i)$ is defined as a conjunction of elementary conditions over the condition attributes, and the decision part $K_t = D(R_i)$ specifies an assignment to a decision class or to a union of decision classes. The number $n = |R_i|$ is the length of the rule.

Definition 2 (Elementary condition)**.** *An elementary condition C_j is an atomic statement of the form*

$$C_j \quad \equiv \quad f(x,q_k) \propto r_{q_k}, \tag{5}$$

where $f(x,q_k)$ is the value of object x on attribute q_k, r_{q_k} is a threshold value, and $\propto$ denotes a comparison operator.

Depending on the type and preference direction of the attribute q_k (i.e., nominal, ordinal, or numerical), $\propto$ can take one of the following values: $\geq,\leq,\geq,\leq$, or $=$.

The decision part K_t can take different forms according to the type of decision rule (see Section 3.2):

$$K_t \quad \equiv \quad x \in Cl_t^{\diamond} \tag{6}$$

$$K_t \quad \equiv \quad \text{possibly } x \in Cl_t^{\diamond} \tag{7}$$

$$K_t \quad \equiv \quad x \in Cl_s \cup Cl_{s+1} \cup ... \cup Cl_t \tag{8}$$

where $\diamond \in \{\geq,\leq\}$. The first form corresponds to at least or most certain decision rules (type 1 or 3), the second form to at least or at most possible decision rules (type 2 or 4), and the last form to approximate decision rules (type 5).

Building on these notions, a graphical representation of decision rules captures all possible rules. Condition and decision components are represented as nodes, and the logical connections between them as directed edges. To formalise this representation, we define a decision graph as follows.

Definition 3 (Decision graph)**.** *A decision graph is a directed acyclic graph $G = (N,E)$ where:*

- *N is a set of nodes corresponding to elementary conditions, decisions, and the root;*
- *$E = \{(u,v) \mid u,v \in N \wedge u \neq v\}$ is a set of directed edges linking root, condition, and decision nodes.*

There are three basic types of nodes: root nodes, condition nodes, and decision nodes. Let $N = N_r \cup N_c \cup N_d$, where N_r is the set of root nodes, $N_c = \{j \mid C_j \in C\}$ is the set of condition nodes, and $N_d = \{t \mid K_t \in K\}$ is the set of decision nodes. N_r is often a singleton, but additional root nodes may exist when the decision graph is composed of several subgraphs, each with its own root.

There are also three basic types of edges: root-to-condition edges, condition-to-condition edges, and condition-to-decision edges. Let $E = E_r \cup E_c \cup E_{cd}$, with $E_r = \{(r,u) : r \in N_r \wedge u \in N_c\}$, $E_c = \{(u,v) : u,v \in N_c\}$, and $E_{cd} = \{(u,v) : u \in N_c \wedge v \in N_d\}$. The root node and corresponding edges can be omitted if all rules start with the same elementary condition. All leaf nodes correspond to decisions.

To formalise how sequences of edges correspond to decision rules, we define a path in a decision graph:

Definition 4 (Path in a decision graph)**.** *Let $G = (N,E)$ be a decision graph. A path from a root node $r \in N_r$ to a decision node $d \in N_d$ is an ordered sequence of edges $(e_1,e_2,...,e_m) \subseteq E$ such that e_1 originates at r, e_m terminates at d, and consecutive edges are connected through intermediate nodes.*

Each such path represents a decision rule, where root-to-condition edges correspond to 'if', condition-to-condition edges to 'and', and condition-to-decision edges to 'then'.

An important constraint on the construction of a decision graph is that all rules in R must be included, and no path in G may correspond to a nonexistent rule. These constraints are formalised as follows.

Proposition 1 (Construction constraints for a decision graph)**.** *Let P denote the set of paths from the root node r to all decision nodes in G. The construction of G must satisfy the following requirements:*

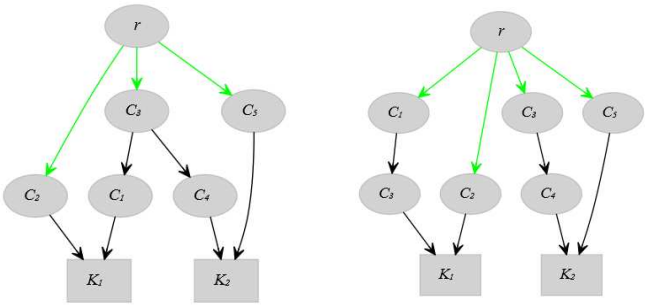
- *For any path $p \in P$, there exists a unique rule $R_i \in R$ such that R_i is a linearisation of p.*
- *For any rule $R_i \in R$, there exists a unique path $p \in P$ such that p linearises into R_i.*

The following didactic example illustrates how a set of decision rules can be represented using a decision graph.

Example 1 (Visualising decision rules through decision graphs)**.** *Let us reconsider the following set of decision rules:*

- R_1: *If* C_1 *and* C_3, *then* K_1
- R_2: *If* C_2, *then* K_1
- R_3: *If* C_3 *and* C_4, *then* K_2
- R_4: *If* C_5, *then* K_2

As in the collection of decision rules above, the elementary condition C_3 *is included in the condition parts of two different rules. Two decision graphs can be associated with this set of decision rules (see Figure 2). In the first version, shown in Figure 2(a), the elementary condition* C_3 *is used only once, while in the second version, shown in Figure 2(b), this elementary condition is split. Observe, in particular, that the decision graph in Figure 2(a) contains one edge fewer than the alternative representation in Figure 2(b). Clearly, splitting shared elementary conditions leads to more edges, but it is permitted to avoid the creation of new and nonexistent rules, as explained in Section 4.2.*



(a) (b)

Figure 2: Examples of decision graphs

In the decision graphs presented in Figures 2(a) and 2(b), nodes are labelled with the elementary conditions, as in fast-and-frugal trees. Alternatively, condition nodes could be labelled with the attribute name, while the conditions themselves are associated with the edges, as is common in decision trees. Without loss of generality, the first option is adopted in this paper. An illustration of the second option is provided in Section 6, within the context of the case study.

The order in which elementary conditions are considered is irrelevant; different orderings may produce distinct yet equivalent decision graphs, meaning that the same classification outcomes are obtained for all objects. Prioritising the most frequently cited elementary conditions can, however, reduce computation time during exploitation of the decision graph, due to the conjunctive nature of these conditions. Specifically, during object classification, a path (representing a decision rule) is discarded as soon as an elementary condition along the path is not satisfied.

For large sets of decision rules, the following strategy can be used: first, decision rules are grouped according to their decision parts; second, for each group of decision rules, a decision subgraph is constructed; finally, a root node is added and linked with the sub-roots of the subgraphs. Figure 3 illustrates this idea using the set of decision rules R_1 to R_4 enumerated earlier in this section.

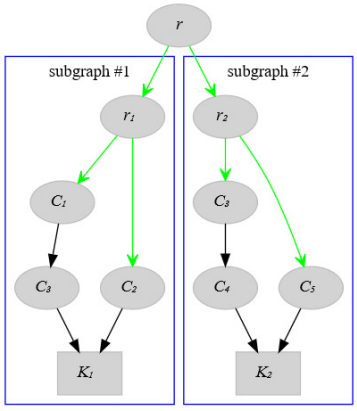


Figure 3: Decision graph using subgraphs

3.4. Principles for constructing and exploiting decision graphs

Figure 4 presents the workflow for constructing and exploiting decision graphs. The process comprises two main phases: the construction of the decision graph from an initial set of if–then rules, and the subsequent exploitation of this graph for rule regeneration, assignment, and decision aggregation. These phases rely on a sequence of operations and algorithms, described in detail in Sections 4 and 5.

he construction phase (Section 4) begins with a set of if–then decision rules, which are encoded into an initial binary matrix B (Section 4.1). Each row of B corresponds to a rule, and each column represents a condition appearing in the rule base. The entries of the matrix indicate the presence or absence of these conditions.

Direct use of the initial matrix may lead to the generation of non-existent rules. To address this, two transformation strategies are employed. The first relies on column splitting, implemented through Algorithm 1, which refines the representation of conditions and captures structural relationships between them (Section 4.2). The second strategy, implemented through Algorithm 2, combines column ordering with splitting to produce a more structured and interpretable matrix (Section 4.3). Both strategies yield a final binary matrix fB, which serves as the input for the decision graph construction algorithm (Section 4.4). Algorithm 3 then constructs a decision graph that provides a compact and hierarchical representation of the logical structure embedded in the original rule set.

The exploitation phase (Section 5) uses the constructed decision graph for linearisation and assignment of objects. The linearisation (Algorithm 4) of a decision graph generates a set of if–then rules that is logically equivalent to the original rule base but may be more concise or better organised (Section 5.1). Assignments of objects exploit the graph to derive assignments (using Algorithm 5) for existing or new objects (Section 5.2). The assignment procedure is supported by a decision aggregation strategy (Algorithm 6), described in Section 5.3, which consolidates outcomes from different parts of the graph to ensure coherent and robust decisions.

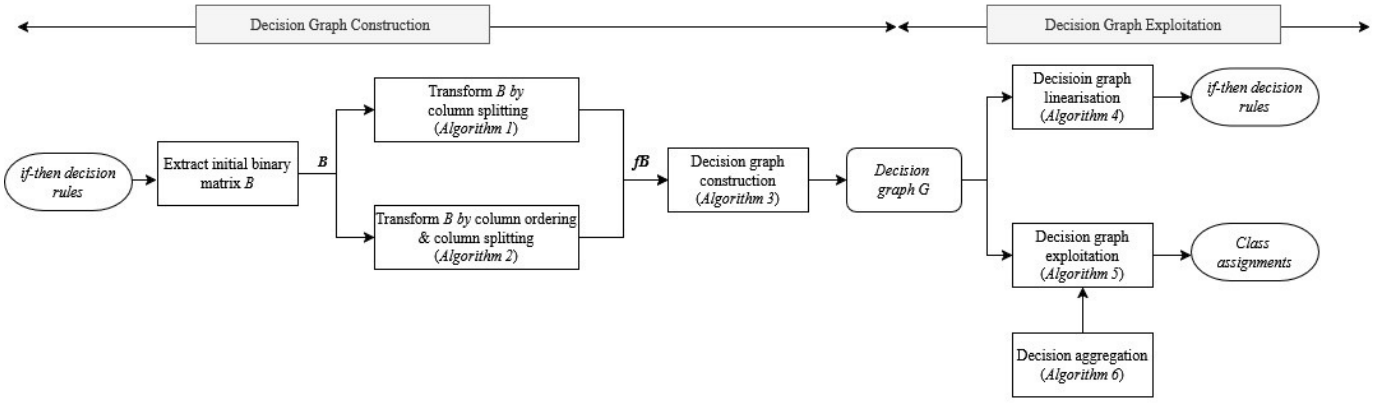


Figure 4: Workflow for constructing and exploiting decision graphs

4. Constructing decision graphs from binary matrices

This section presents the process of constructing decision graphs from a set of if–then decision rules. It begins with the encoding of the rules into a binary decision matrix (Section 4.1), then describes two transformation strategies to prevent the creation of nonexistent rules: the column-splitting approach (Section 4.2) and the mixed column-ordering and column-splitting strategy (Section 4.3). Finally, the construction of the decision graph from the transformed matrix is detailed in Section 4.4. Together, these steps ensure that the resulting decision graph faithfully represents the original rule set while minimising spurious nodes and edges.

4.1. Binary decision matrix

The decision graph construction algorithm introduced later in Section 4.4 relies on a binary decision matrix B of size $|R| \times (|N_c| + |N_d|)$.

Definition 5 (Binary decision matrix)**.** *Let $R = \{R_i\}$ be a set of decision rules with condition nodes N_c and decision nodes N_d. The binary decision matrix B has entries defined as follows:*

$$B_{i,j} = \begin{cases} 1, & \text{if } C_j \in A(R_i), \\ 0, & \text{otherwise,} \end{cases} \quad j = 1,\ldots,|N_c|, \tag{9}$$

$$B_{i,t} = \begin{cases} 1, & \text{if } K_t \in D(R_i), \\ 0, & \text{otherwise,} \end{cases} \quad t = 1,\ldots,|N_d|. \tag{10}$$

The binary decision matrix provides a compact representation of the conditions and decisions present in a set of rules. Each row corresponds to a decision rule, while each column represents either a condition or a decision. An entry of '1' indicates that the condition is part of the rule or that the rule leads to the corresponding decision, and '0' otherwise. This matrix serves as the foundation for constructing the decision graph and ensures a systematic mapping between rules, conditions, and decisions.

The definition of a binary decision matrix is subject to two fundamental constraints, as follows:

$$\sum_{j=1}^{|N_c|} B_{i,j} \geq 1, \quad i = 1,...,|B| \tag{11}$$

$$\sum_{t=N_c+1}^{|N_c|+|N_d|} B_{i,t} = 1, \quad i = 1,...,|B| \tag{12}$$

The first constraint ensures that each row (decision rule) contains at least one elementary condition, while the second ensures that each decision rule is mapped to exactly one decision. Additional, more specific constraints are introduced below.

Initial investigations showed that the application of the decision graph construction algorithm using the binary decision matrix B as defined above may lead to the creation of nonexistent decision rules when some elementary conditions are shared by more than one decision rule. This phenomenon is illustrated in the following example.

Example 2 (Illustrating the creation of nonexistent rules)**.** *The binary decision matrix associated with decision rules R_1 to R_4 from the previous section is shown in Table 1. Applying the decision graph construction algorithm using this binary decision matrix will lead to the decision graph shown in Figure 5. Analysis of this decision graph reveals that it represents the initial decision rules R_1 to R_4, along with two newly introduced decision rules:*

- *R_5: If C_3, then K_1*
- *R_6: If C_1 and C_3 and C_4, then K_2*

These additional rules, R_5 and R_6, are nonexistent in the original rule set: they do not correspond to any rule in R_1–R_4. They arise because some elementary conditions, such as C_3, are shared across multiple rules, and the straightforward construction of the decision graph from the initial binary matrix inadvertently creates new paths.

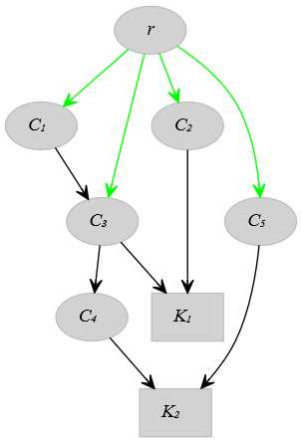


Figure 5: Decision graph from initial binary matrix in Table 1

Table 1: Binary decision matrix associated with decision rules R_1 to R_4

	C_j					K_t	
R_i	C_1	C_2	C_3	C_4	C_5	K_1	K_2
R_1	1	0	1	0	0	1	0
R_2	0	1	0	0	0	1	0
R_3	0	0	1	1	0	0	1
R_4	0	0	0	0	1	0	1

Two methods for eliminating the creation of nonexistent decision rules are presented in the following two sections. The first is a simple method based on column splitting, while the second is more advanced and uses optimisation. Both approaches prevent the creation of nonexistent decision rules, but the second often leads to smaller decision graphs, particularly for large sets of decision rules.

4.2. Binary matrix transformation using the column-splitting approach

When constructing a decision graph from a set of decision rules, it is essential to ensure that the binary decision matrix does not produce nonexistent or spurious decision rules. Such issues arise when a single column represents multiple occurrences of the same elementary condition across different rules. In such cases, the decision graph construction algorithm may combine entries from different rows, creating decision paths that do not correspond to any original rule.

To prevent this, the transformed binary matrix must satisfy:

$$\sum_{i=1}^{|R|} B_{i,j} \leq 1, \qquad j = 1,...,|N_c|. \tag{13}$$

This condition ensures that each condition column contains at most one entry equal to 1. A straightforward way to enforce this is to split any column with more than one occurrence of the corresponding elementary condition. Each split column then represents a distinct occurrence in a specific decision rule, preserving unique decision paths. This effect becomes clear when the column-splitting transformation is applied to the initial binary matrix, as illustrated in the following example.

Example 3 (Column splitting in the binary matrix)**.** *Applying this transformation to the initial binary decision matrix in Table 1 produces the updated matrix shown in Table 2. Because nodes of a decision graph must be uniquely identifiable, the split column is labelled C_3'. Running the decision graph construction algorithm on this transformed matrix yields the correct decision graph shown in Figure 2(b).*

Table 2: Binary decision matrix associated with decision rules R_1 to R_4 after column splitting

	C_j						K_t	
R_i	C_1	C_2	C_3	C_3'	C_4	C_5	K_1	K_2
R_1	1	0	1	0	0	0	1	0
R_2	0	1	0	0	0	0	1	0
R_3	0	0	0	1	1	0	0	1
R_4	0	0	0	0	0	1	0	1

The column-splitting transformation is formalised in Algorithm 1. The transformation is entirely mechanical. Each column is processed independently, and splitting requires only copying column vectors. The overall time and space complexity is $O(|R| \cdot |N_c|)$, linear in the size of the matrix; details are provided in Appendix B.1.

Algorithm 1: Column-splitting transformation of a binary decision matrix

Input : B: Binary decision matrix with condition columns $C_1,...,C_{|N_c|}$
Output: fB: Transformed binary decision matrix with at most one 1 per condition column

```
for j = 1 to |N_c| do
    Count the number of 1s in column C_j;
    if count > 1 then
        Split C_j as many times as needed so each new column has exactly one 1;
        Replace C_j in B with the set of split columns;
fB := B after all necessary splits;
return fB;
```

4.3. Binary matrix transformation using mixed column-ordering and column-splitting strategy

Column-splitting eliminates nonexistent decision rules but increases the size of the decision graph and computation time. An alternative column-ordering strategy can reduce added nodes and edges. This second approach relies on the following constraint:

$$\sum_{i=1}^{|R|} B_{i,j} \;\geq\; \sum_{i=1}^{|R|} B_{i,j'}, \qquad j,j' \in \{1,...,|N_c|\} \wedge j < j'. \tag{14}$$

However, the column-ordering strategy cannot entirely prevent nonexistent rules. Combining both column-splitting and column-ordering strategies yields a unified optimisation: before graph construction, B is analysed to determine which columns can be safely ordered and which must be split. The transformed matrix is then constructed by placing the safely orderable columns first, followed by the split columns, ensuring correctness while minimising added nodes and edges.

Let B be the binary decision matrix of size $|R| \times (|N_c| + |N_d|)$, and focus on the elementary-condition columns $j = 1,...,|N_c|$. For each C_j, define the set of rules in which it appears:

$$S_j = \{i \in \{1,...,|R|\} : B_{i,j} = 1\}. \tag{15}$$

Two columns C_j and $C_{j'}$ interact as follows:

- Safe: $S_j \subseteq S_{j'}$ or $S_{j'} \subseteq S_j$. Ordering these columns cannot create new combinations.
- Unsafe: $S_j \cap S_{j'} \neq \emptyset$ but neither is a subset of the other. At least one column must be split to avoid nonexistent rules.

The partial order defined by $C_j \preceq C_{j'} \iff S_j \subseteq S_{j'}$ allows extraction of a maximum chain of safely orderable columns, X^O. Remaining columns, X^D, are split into X^S. The transformed matrix places X^O first, followed by X^S.

Each split column C_j appearing in k rules ($S_j = \{i_1,...,i_k\}$) is replaced by k new columns $C_{j,1},...,C_{j,k}$, each containing a single 1:

$$C_{j,r}(i) = \begin{cases} 1 & \text{if } i = i_r, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

This preserves the original rule base, prevents artificial rules, and ensures each split column is a distinct node. Columns in X^D are fully replaced by X^S; the final matrix therefore contains only X^O and X^S.

Column selection for X^D can be exact (minimum hitting set) or greedy (select column with most unsafe interactions iteratively). The order of X^O columns is determined solely by subset relations. The order of columns within X^O is determined solely by the subset relations among their co-occurrence sets, not by the number of 1-entries they contain. Consequently, a column that appears in only one rule may legitimately precede columns with multiple occurrences, provided its co-occurrence set is a subset of theirs. This behaviour is illustrated in the example given below.

Example 4 (Illustration of mixed column-ordering and column-splitting strategy)**.** *Consider five decision rules R_7-R_{11} defined over six conditions C_1-C_6 and two decisions K_1, K_2:*

- R_7: $C_1 \wedge C_3 \rightarrow K_1$
- R_8: $C_2 \rightarrow K_1$
- R_9: $C_3 \wedge C_4 \rightarrow K_2$
- R_{10}: $C_5 \rightarrow K_2$
- R_{11}: $C_4 \wedge C_6 \rightarrow K_2$

The initial binary decision matrix (Table 3(a)) shows which conditions trigger which decisions. From this, we derive co-occurrence sets for each condition, indicating in which rules they appear. In this case, the co-occurrence sets are: $S_1 = \{7\}$, $S_2 = \{8\}$, $S_3 = \{7,9\}$, $S_4 = \{9,11\}$, $S_5 = \{10\}$, and $S_6 = \{11\}$. A maximum chain under subset inclusion is $S_1 \subseteq S_3 \subseteq S_4$, giving the ordered columns: $X^O = \{C_1,C_3,C_4\}$. The remaining columns $X^D = \{C_2,C_5,C_6\}$ are treated as split columns, producing: $X^S = \{C_2,C_5,C_6\}$.

The final transformed matrix (Table 3 (b)) places the chain X^O first, followed by the split columns X^S. This arrangement ensures that the decision graph faithfully represents all original rules, avoids creating nonexistent rules, and reduces unnecessary nodes and edges.

Table 3: Initial and transformed binary decision matrices associated with decision rules R_7–R_{11}

(a) Initial binary decision matrix

	C_j						K_t	
R_i	C_1	C_2	C_3	C_4	C_5	C_6	K_1	K_2
R_7	1	0	1	0	0	0	1	0
R_8	0	1	0	0	0	0	1	0
R_9	0	0	1	1	0	0	0	1
R_{10}	0	0	0	0	1	0	0	1
R_{11}	0	0	0	1	0	1	0	1

(b) Transformed binary decision matrix

	C_j (reordered)						K_t	
R_i	C_1	C_3	C_4	$C_{2,1}$	$C_{5,1}$	$C_{6,1}$	K_1	K_2
R_7	1	1	0	0	0	0	1	0
R_8	0	0	0	1	0	0	1	0
R_9	0	1	1	0	0	0	0	1
R_{10}	0	0	0	0	1	0	0	1
R_{11}	0	0	1	0	0	1	0	1

Algorithm 2 implements the mixed column-ordering and column-splitting strategy, differing only in how columns involved in unsafe pairs are selected (exact vs. greedy variant). This approach efficiently preserves all original decision rules. Its overall time complexity is $O(|R| \cdot |N_c|^2)$ for the greedy variant and may be exponential in $|N_c|$ for the exact variant. The overall space complexity is $O(|R| \cdot |N_c|^2)$. Detailed analysis of time and space complexity is provided in Appendix B.2.

The mixed column-ordering and column-splitting strategy can yield multiple valid solutions, as different maximum chains may arise from the co-occurrence sets $\{S_j\}$. All solutions preserve the original rule structure without introducing spurious paths and typically produce decision graphs of similar size. Additional considerations regarding alternative valid transformations and ordered-column placement are provided in Appendix C. A further practical consideration concerns the selection of columns to be split in Algorithm 2. This selection can be performed using either the exact or greedy variant, depending on the size and structure of the decision-rule base. Let p denote the number of decision rules, $m = |N_c|$ the number of condition attributes, and k the average rule length. As a practical guideline, the exact variant is appropriate when the rule base is relatively small, for example when p contains only a few tens or hundreds of rules, m is limited to a few tens of attributes, and the number of unsafe interactions remains moderate. Under these conditions, the minimum hitting-set optimisation can be solved within an acceptable time while providing the smallest possible set of split columns and, consequently, the most compact transformed representation.

When the rule base becomes larger, for example when p reaches thousands of rules, m grows to hundreds of attributes, or the combination of p, m, and the average rule length k produces a large number of unsafe column pairs, the greedy variant is generally more suitable. In such cases, the exact optimisation may become computationally expensive because the number of possible column selections increases rapidly. The greedy strategy avoids this combinatorial search by iteratively selecting columns that resolve the largest number of remaining conflicts. Although this may not always produce the minimum possible number of split columns, it provides an effective balance between transformation quality and computational scalability for large rule bases.

Algorithm 2: Optimised binary decision matrix transformation with column splitting (exact or greedy variant)

Input : B: Binary decision matrix with condition columns $C_1,...,C_{|N_c|}$
Output: fB: Transformed binary decision matrix

for $j = 1$ **to** $|N_c|$ **do**
  $S_j := \{i \mid B_{i,j} = 1\}$;
Define relation $C_j \preceq C_{j'}$ iff $S_j \subseteq S_{j'}$;
Compute a maximum-length chain $X^O = (C_{j_1}, C_{j_2}, ..., C_{j_m})$ such that $S_{j_1} \subseteq S_{j_2} \subseteq \cdots \subseteq S_{j_m}$;
$X^D := \{C_j \mid C_j \notin X^O\}$;
Compute the set $\mathcal{U}$ of unsafe pairs among columns in $X^O \cup X^D$;
while $\mathcal{U} \neq \emptyset$ **do**
  Select C_j using either:
    (a) *Exact*: choose C_j as part of a minimum hitting set for $\mathcal{U}$, or
    (b) *Greedy*: choose C_j that appears in the largest number of unsafe pairs;
  Move C_j to X^D;
  Remove from $\mathcal{U}$ all pairs involving C_j;
$X^S := \emptyset$;
foreach $C_j \in X^D$ **do**
  Let $S_j = \{i_1, ..., i_k\}$;
  for $r = 1$ **to** k **do**
    Create a new column $C_{j,r}$ where

$$C_{j,r}(i) = \begin{cases} 1 & \text{if } i = i_r, \\ 0 & \text{otherwise.} \end{cases}$$

    Add $C_{j,r}$ to X^S;
Reorder columns so that X^O appear first in chain order;
$fB := [X^O \mid X^S]$;
Reorder columns so that X^O appear first in chain order;
$fB := [X^O \mid X^S]$;
return fB;

4.4. Decision graph construction algorithm

The decision graph construction procedure is given in Algorithm 3. It takes as input a collection R of decision rules and generates a decision graph $G = (N,E)$. It can be organised into three phases: (1) extraction and transformation of the binary decision matrix; (2) addition of root-to-condition edges; and (3) addition of condition-to-condition and condition-to-decision edges. The description of the algorithm is straightforward. The first instruction in Algorithm 3 initialises an order-zero graph $G = (N,E)$ that will contain the decision graph. Algorithm 3 then starts by extracting the binary decision matrix B from the collection of decision rules R. The function *TransformBDM* then applies one of the solutions given in Sections 4.2 and 4.3 to transform the initial binary decision matrix B into a final binary decision matrix fB. After the transformation of the binary decision matrix, we can then obtain the final list of nodes N. The first 'for-loop' in Algorithm 3 simply adds the root-to-condition edges by examining all the rows in fB and, for each row i, identifying the first column j such that $fB_{i,j} = 1$. It then adds the edge (r,j) to the set of edges E. The next 'for-loop' is devoted to adding condition-to-condition and condition-to-decision edges. It loops over the columns in fB and, for each column j, scans all the rows in fB. For any row i such that $fB_{i,j} = 1$, it identifies the first column $j' > j$ such that $fB_{i,j'} = 1$. If this holds, the edge (j,j') is added to E.

Algorithm 3: Decision graph construction

Input : R: A collection of decision rules;
Output: $G = (N,E)$: A decision graph;

Let $G = (N,E)$ be an order-zero graph;
//—Phase 1 — Extraction and transformation of the binary decision matrix—//
B := binary decision matrix extracted from R;
fB := **TransformBDM**(B);
$N := \{r\} \cup \{1,...,|N_c| + |N_d|\}$;
//—Phase 2 — Addition of root-to-condition edges—//
for $i \in \{1,...,|R|\}$ **do**
  $j := 1$;
  while $fB_{i,j} \neq 1$ **do**
    $j := j + 1$;
  if $(r,j) \notin E$ **then**
    add (r,j) to E;
//—Phase 3 — Addition of condition-to-condition and condition-to-decision edges—//
for $j \in \{1,...,|N_c|\}$ **do**
  $i := 1$;
  while $fB_{i,j} \neq 1$ **do**
    $i := i + 1$;
  $h := j + 1$;
  while $fB_{i,h} \neq 1$ **do**
    $h := h + 1$;
  if $(j,h) \notin E$ **then**
    add (j,h) to E;
return $G = (N,E)$;

Algorithm 3 has linear time and space complexity in terms of the total size of the input, i.e. $O(m \times n)$. A detailed analysis of the complexity of Algorithm 3 is given in Appendix B.3. The following example demonstrates how the algorithm constructs a

decision graph step by step.

Example 5 (Illustration of the decision graph construction algorithm)**.** *Consider the following set of decision rules:*

- R_{12}*: If* C_1 *and* C_3 *and* C_6*, then* K_1
- R_{13}*: If* C_2 *and* C_5*, then* K_1
- R_{14}*: If* C_3 *and* C_4*, then* K_2
- R_{15}*: If* C_4 *and* C_6 *and* C_7*, then* K_3
- R_{16}*: If* C_2*, then* K_3

The algorithm first extracts the binary decision matrix B from rules R_{12}*-*R_{16} *(Table 4(a)) and transforms it into the final matrix fB using TransformBDM (Table 4(b)). We assume the column-splitting solution (Section 4.2). Columns* C_2*,* C_3*,* C_4 *and* C_6 *are therefore split into* C'_2*,* C'_3*,* C'_4 *and* C'_6*, respectively. The integers in the second row of Table 4(b) uniquely identify the nodes.*

The algorithm then proceeds in two stages:

- *Addition of root-to-condition edges. The first condition of each rule is connected to the root node, producing the edges* (r,C_1)*,* (r,C_2)*,* (r,C'_3)*,* (r,C'_4)*, and* (r,C'_2)*.*
- *Addition of condition-to-condition and condition-to-decision edges. Edges are then created between successive conditions and finally to the decision node for each rule. The resulting paths are* $(C_1,C_3) \rightarrow (C_3,C_6) \rightarrow (K_1)$*,* $(C_2,C_5) \rightarrow (K_1)$*,* $(C'_3,C_4) \rightarrow (K_2)$*,* $(C'_4,C'_6) \rightarrow (C'_6,C_7) \rightarrow (K_3)$*, and* (C'_2,K_3)*.*

The resulting decision graph is shown in Figure 6. Each root-to-decision path corresponds to one original rule, and no spurious paths are created due to the column-splitting transformation.

Table 4: Binary decision matrices associated with decision rules R_{12}–R_{16} before and after column splitting

(a) Original binary decision matrix

	C_j							K_t		
R_i	C_1	C_2	C_3	C_4	C_5	C_6	C_7	K_1	K_2	K_3
R_{12}	1	0	1	0	0	1	0	1	0	0
R_{13}	0	1	0	0	1	0	0	1	0	0
R_{14}	0	0	1	1	0	0	0	0	1	0
R_{15}	0	0	0	1	0	1	1	0	0	1
R_{16}	0	1	0	0	0	0	0	0	0	1

(b) Binary decision matrix after column splitting

	C_j											K_t		
Node #	1	2	3	4	5	6	7	8	9	10	11	12	13	14
R_i	C_1	C_2	C'_2	C_3	C'_3	C_4	C'_4	C_5	C_6	C'_6	C_7	K_1	K_2	K_3
R_{12}	1	0	0	1	0	0	0	0	1	0	0	1	0	0
R_{13}	0	1	0	0	0	0	0	1	0	0	0	1	0	0
R_{14}	0	0	0	0	1	1	0	0	0	0	0	0	1	0
R_{15}	0	0	0	0	0	0	1	0	0	1	1	0	0	1
R_{16}	0	0	1	0	0	0	0	0	0	0	0	0	0	1

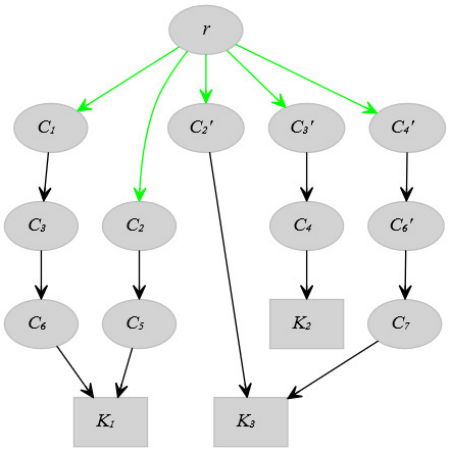


Figure 6: Decision graph associated with decision rules R_{12} to R_{16}

5. Linearising and exploiting decision graphs

This section introduces the principles of linearising and exploiting decision graphs. In particular, subsection 5.1 presents a method for linearising a decision graph into a set of decision rules, facilitating validation and rule extraction. Then, subsection 5.2 details the exploitation of decision graphs for classifying unseen decision objects using a BFS-based approach. Finally, subsection 5.3 describes a decision aggregation strategy for resolving multiple or conflicting rule assignments, ensuring a single, admissible class is assigned to each object.

5.1. Linearisation of decision graphs

Several well-known graph algorithms can be applied to decision graphs, often with minor adjustments to accommodate their specific characteristics. In particular, a decision graph can be linearised into a set of decision rules using the Breadth-First Search (BFS) algorithm, which generates all pathways from the root node to every decision node. Linearisation is mainly useful for validation, as it ensures that all original decision rules are accurately represented in the decision graph and that no non-existent rules are inadvertently introduced. Algorithm 4 outlines a BFS-based decision graph linearisation (BFSL) algorithm. It takes a decision graph $G = (V,E)$, a root node and a decision node, and generates a list of decision rules. Algorithm 4 explores nodes in a graph level by level, starting from the root node and exploring all its neighbours before moving to the next level. The BFSL algorithm performs a breadth-first search on a decision graph G starting from the root node, exploring all paths and collecting those that reach the decision node. The paths are stored as lists of nodes in a queue, and when a path reaches the decision node, it is added to the set R.

Since the basic BFS algorithm returns only the shortest path from a source node to each leaf node, finding all paths from the root node to a decision node requires avoiding the premature marking of nodes as visited during the entire graph traversal process. This can be achieved by storing paths while exploring nodes and by avoiding global marking of visited nodes, as a node may need to be revisited along different paths. The main modification to the basic BFS is therefore to store paths (instead of individual nodes) in the queue. Each path is represented as a vector of integers. The queue can thus hold multiple paths, which are progressively extended and explored during the traversal of the graph. To construct a path, instead of simply visiting nodes, a new path is created by appending the current adjacent node to the existing path, and the resulting path is then pushed into the queue. This approach enables the exploration of every possible path to the destination.

Algorithm 4: BFS-based decision graph linearisation (BFSL)

```
   Input  : G:=(V,E): A decision graph
            root: The root of G
            decision_node: A decision node
   Output: R: A set of decision rules
 1 //— Initialize a queue Q to store paths —
 2 let Q be an empty queue;
 3 //— Initialize the starting path with the root node —
 4 initialize start_path as [root];
 5 //— Enqueue the start_path to the queue —
 6 enqueue(Q,start_path);
 7 //— While there are paths in the queue —
 8 while Q is not empty do
 9     //— Dequeue the first path from the queue —
10     path:=dequeue(Q);
11     //— Get the last node in the current path —
12     current_node:=last_element(path);
13     //— If the current node is the decision node, build and add the rule to R —
14     if current_node = decision_node then
15         build a decision rule based on path and add it to R;
16     //— Explore all adjacent nodes of the current node —
17     for each adj_node adjacent to current_node do
18         //— Create a new path by appending the adjacent node to the current path —
19         new_path:= path + [adj_node];
20         //— Enqueue the new path to the queue —
21         enqueue(Q,new_path);
22 return R;
```

Algorithm 4 generates the decision rules from the root to a given decision node. To obtain all decision rules, BFSL can simply be applied several times, each time using a different decision node. Algorithm 4 has linear time and space complexity with respect to the total size of the input, i.e. $O(|V| + |E|)$. A detailed analysis of the complexity of Algorithm 4 is provided in Appendix B.4.

5.2. Exploitation of decision graphs

As with decision graph linearisation, a BFS-like algorithm can also be used to classify unseen decision objects. The outline of the proposed BFS-based decision graph exploitation (BFSE) algorithm is given in Algorithm 5. The BFSE algorithm takes a decision graph $G = (V,E)$, its root, and a decision object x, and determines the decisions for x. In essence, BFSE scans each path in G and identifies those paths supported by the object. A decision object x supports a path p if and only if its scores satisfy the elementary conditions contained in the nodes composing that path. Algorithm 5 is similar to the decision graph linearisation algorithm, but the two algorithms serve different purposes. BFSE is used for decision-making (classifying an object based on the graph), whereas BFSL is used for rule extraction (generating decision rules from the graph). Both algorithms employ BFS to traverse the graph, but their focus and outputs differ. The main differences between the two algorithms are: (i) BFSE identifies the possible decisions for a given object, whereas BFSL generates all decision rules in the decision graph; (ii) when exploring the adjacent nodes of the current node, BFSE considers only those nodes corresponding to elementary conditions satisfied by the decision object, whereas BFSL considers all adjacent nodes.

The BFSE algorithm returns the admissible class set $A(x)$, which contains all decision values suggested by the decision nodes reached along the paths supported by the object x. This set represents the full range of possible classifications for x according to the decision graph. When a single decision is required, the decision aggregation strategy described in Section 5.3 can be applied to reduce $A(x)$ to a unique final assignment.

Algorithm 5 has linear time and space complexity with respect to the total size of the input, i.e. $O(|V|+|E|)$. A detailed analysis of the complexity of Algorithm 5 is provided in Appendix B.5.

Algorithm 5: BFS-based decision graph exploitation (BFSE)

```
Input  : G:=(V, E): A decision graph
         root: The root node of G
         x: Decision object to classify
Output: A(x): Admissible class set for decision object x
// — Create and initialise the queue Q to store paths —
let Q be an empty queue;
// — Initialise the admissible class set —
let A(x) be an empty set;
// — Initialize the starting path with just the root node —
initialize start_path as [root];
// — Enqueue the start_path into the queue —
enqueue(Q,start_path);
// — While the queue Q is not empty, explore paths —
while Q is not empty do
    // — Dequeue the first path from the queue —
    current_path := dequeue(Q);
    // — Get the last node in the current path —
    current_node := last_element(current_path);
    // — If the current node is a decision node, classify the object x —
    if current_node is a decision node then
        cl(x) := decision_class(current_node);
        add_to_set(A(x), decision_class(current_node));
    // — Explore all adjacent nodes of the current node —
    for each adj_node adjacent to current_node do
        // — Exit the for-loop if x is not covered by the elementary condition in adj_node —
        if (adj_node is a condition node and x is not covered by elem_condition(adj_node)) then
            break;
        // — Create a new path by appending adj_node to the current path —
        new_path := current_path + [adj_node];
        // — Enqueue the new path to the queue —
        enqueue(Q, new_path);
// — Return the class of x —
return A(x);
```

For exploiting a decision graph with subgraphs, the BFSE algorithm can be applied to each subgraph separately, after which the results can be combined and refined using simple aggregation rules.

5.3. Decision aggregation strategy

At the end of the exploitation algorithm, three situations may arise for the assignment of a decision object x: (i) exactly one rule covers x, (ii) several rules cover x and point to different decision classes, or (iii) no rule covers x. In the first case, the assignment is straightforward: $cl(x)$ is simply the class indicated by the single covering rule. In the second case, if all covering rules are of the same type, the classical solutions of Greco et al. (2002) apply directly. Otherwise, some basic aggregation operators (e.g., min, max, median) can be used to reduce the assignment interval to a single decision. This idea has been applied in Chakhar & Saad (2012) and Chakhar et al. (2016) in a DRSA-based group decision context. Another possible solution is the one proposed by Błaszczyński et al. (2007). In the third case, following the convention used in Błaszczyński et al. (2007); Chakhar & Saad (2012); Chakhar et al. (2016), the object is assigned to all decision classes.

When several rules cover x, an aggregation strategy is required. If all covering rules are of the same type (all at-most or all at-least), the solutions of Greco et al. (2002) apply directly: for at-most rules, $cl(x)$ is the smallest decision value among the covering rules; for at-least rules, it is the largest. This corresponds to interpreting the rules as compatible constraints whose intersection yields a single admissible class. When covering rules are of mixed types, the possible assignments form an interval. Let $\Omega = \{Cl_1, Cl_2, ..., Cl_{|\Omega|}\}$ denote the ordered set of decision classes. Let $l(x)$ denote the lowest admissible class suggested by any at-least rule covering x (or Cl_1 if none exist), and let $u(x)$ denote the highest admissible class suggested by any at-most rule (or $Cl_{|\Omega|}$ if none exist). The interval $[l(x), u(x)]$ therefore represents all classes consistent with the rules covering x. Two cases may occur:

- If $l(x) = u(x)$, then the assignment is unique: the interval $[l(x), u(x)]$ degenerates to a single class. In this case, $cl(x) = l(x) = u(x)$.
- If $l(x) < u(x)$, then, in order to obtain a single final assignment, the interval $[l(x), u(x)]$ must be reduced using an aggregation operator. Depending on the decision context, this operator may be optimistic (choosing $u(x)$), pessimistic (choosing $l(x)$), or based on another context-dependent criterion. Following DOMLEM, we adopt the optimistic choice, i.e., $cl(x) = u(x)$, which selects the most optimistic admissible class.

The optimistic aggregation operator is adopted by default because it is consistent with selecting the highest decision class supported by the available evidence and follows the convention used in DOMLEM. This choice is appropriate when decision support seeks the best achievable outcome or when underestimating an object's class is considered more costly than overestimation. In contrast, the pessimistic operator may be preferred in risk-sensitive applications where conservative decisions are required and assigning an excessively high class should be avoided. A median or other intermediate aggregation strategy may be appropriate when a balanced compromise is required, reducing the influence of extreme admissible classes.

The decision aggregation strategy is formalised in Algorithm 6. The algorithm takes as input the admissible class set $A(x)$ of an object x. It partitions the set into $K^{\leq}(x)$ (at-most decisions) and $K^{\geq}(x)$ (at-least decisions). The final class assignment is computed

Algorithm 6: Decision aggregation strategy (compact version)

Input : $A(x)$: admissible class set for decision object x
Output: $cl(x)$: final assignment for decision object x

```
// Separate decision types
```
let $K^{\leq}(x) := \{d \in A(x) : d \text{ is at-most}\}$;
let $K^{\geq}(x) := \{d \in A(x) : d \text{ is at-least}\}$;
```
// Compute admissible interval
```
$l(x) := \min(K^{\geq}(x))$ if $K^{\geq}(x) \neq \emptyset$, else Cl_1;
$u(x) := \max(K^{\leq}(x))$ if $K^{\leq}(x) \neq \emptyset$, else $Cl_{|\Omega|}$;
```
// Assign final class
```
if $l(x) = u(x)$ **then**
    $cl(x) := l(x)$;
else
    // Apply aggregation operator (context-dependent)
    - Optimistic: $cl(x) := u(x)$
    - Pessimistic: $cl(x) := l(x)$
    - Other operator depending on context
return $cl(x)$;

efficiently using interval bounds, with time complexity $O(|A(x)|)$ and space complexity $O(|A(x)|)$. A detailed analysis is provided in Appendix B.6.

6. Salerno case study

To illustrate the proposed approach, we consider the assessment of historic building adaptive reuse projects in the city of Salerno (Southern Italy). The software 4eMka2 (4eMka, 2000) has been used for rough approximations. All the algorithms proposed in this paper have been implemented in Python. The full source code, together with the data used in this study, is available at `https://github.com/chakhar1702/DeciGraph`. Finally, all decision graphs presented in this paper have been generated using the open source graph visualisation software Graphviz (freely available at `https://graphviz.org/`).

6.1. Problem description and motivation

The Municipality of Salerno has undertaken several projects for the regeneration of its historic centre, but not all adaptive reuse interventions have been satisfactory. Therefore, the Municipality of Salerno was interested in understanding the factors that may have influenced the success or failure of local adaptive reuse projects in order to better direct resources in the future. Hence, the DRSA has been used to analyse some historic building adaptive reuse projects in the city of Salerno and provide insights based on the lessons learned from previous projects. The DRSA allowed the extraction of a set of if-then decision rules that made it possible to explore the conditions/factors involved in the success or failure of the reuse projects. However, we observed that although local experts are able to understand decision rules taken separately, they have more difficulty in having a clear overview of the set of decision rules, especially when the full list of decision rules is presented to them. At the same time, presenting all the induced decision rules was challenging to the experts, as the list is quite long.

These issues motivated the authors to think of innovative solutions to communicate decision rules to experts. Initially inspired by Fast & Frugal Trees and then by Decision Trees, the idea of developing visual and graphical tools to represent dominance-based if-then decision rules quickly emerged.

6.2. Specification of criteria, identification and assessment of adaptive reuse projects and rough approximation

The characteristics of the criteria adopted in this study are presented in Table 5. In addition, a set of 16 adaptive reuse projects was identified by the experts for inclusion in the study. These projects comprise different types of historical buildings owned by the City of Salerno and represent different stages of adaptive reuse. The selected projects were evaluated by local experts with respect to the criteria listed in Table 5. Using a four-level Likert scale (Low (L), Moderate (M), High (H), and Very High (VH)), the experts assessed the overall satisfaction with the current utilisation of each project. The characteristics of the selected projects and the complete assessment results are reported in Appendix D.1.

Table 5: List of criteria

Code	Name	Description	Preference	Data type	Domain values for ordinal and nominal datatypes
A1	Date_of_Construction	Date of construction defined as the century of the building's construction	Cost	Integer	
A2	Area_Covered	Surface of covered area in Sq metre	Gain	Continuous	
A3	Current_Condition	Current condition of the building	Cost	Ordinal	Restored (best); Partially restored; Restoration on going; To be restored (worst)
A4	Costs_of_Intervention	The budget in Euros used to renovate and reuse the building	Cost	Continuous	
A5	Economic_Self-Sustainability	Indicates whether or not the construction can sustain itself or not	Gain	Ordinal	No (worst); Yes (best)

The DRSA is then applied to approximate the decision classes Cl_1, Cl_2, Cl_3 and Cl_4 corresponding to Low, Moderate, High and Very High overall satisfaction levels, respectively. Appendix D.2 provides a brief overview of results of the rough approximation.

6.3. Induction and representation of decision rules

Two different sets of decision rules are used to illustrate the decision graph models. The first set corresponds to the minimal cover induced by DOMLEM algorithm Greco et al. (2002)Greco et al. (2001b)Greco et al. (2001c) while the second set is a selection from all possible decision rules induced by All Rules on Examples (AllRules) algorithm which induces all possible decision rules. AllRules uses the Explore algorithm Stefanowski & Vanderpooten (2001) without any stopping conditions. The software 4eMka2 (4eMka, 2000) offers both rule induction algorithms.

6.3.1. Graphical representation of decision rules in the minimal cover

The DOMLEM algorithm has been used to induce a minimal set of rules. The obtained minimal cover is shown in Table 6. To generate the decision graph associated with this set of decision rules, we need first to extract the elementary conditions included in the decision parts of the decision rules. The obtained list is shown in Table 7. A careful examination of Table 6 shows that elementary conditions C_3 (Date_of_Construction≥18) and C_{16} (Economic_Self_Sustainability=No) have been used twice while all the other elementary conditions have been used once.

Table 6: List of decision rules in the minimal cover

Rule #	Rule Description	Strength (%)
1	**If** (Date_of_Construction≥18) & (Economic_Self_Sustainability=No) & (Costs_of_Intervention≥500000) **then** (Overall_Satisfaction=at most Low)	100
2	**If** (Economic_Self_Sustainability≤No) & (Current_Condition≥Partially_restored) **then** (Overall_Satisfaction=at most Moderate)	80.00
3	**If** (Costs_of_Intervention≥3135879.53) & (Area_Covered≤1300) **then** (Overall_Satisfaction=at most Moderate)	40.00
4	**If** (Costs_of_Intervention≥3778368.93) **then** (Overall_Satisfaction=at most High)	25.00
5	**If** (Date_of_Construction≥18) **then** (Overall_Satisfaction=at most High)	50.00
6	**If** (Area_Covered≥3000) & (Costs_of_Intervention≤1550000) **then** (Overall_Satisfaction=at least Very_High)	50.00
7	**If** (Costs_of_Intervention≤920937.37) & (Date_of_Construction≤14) & (Current_Condition≤Partially_restored) **then** (Overall_Satisfaction=at least Very_High)	50.00
8	**If** (Economic_Self_Sustainability=Yes) **then** (Overall_Satisfaction=at least High)	81.82
9	**If** (Costs_of_Intervention≤39400.48) **then** (Overall_Satisfaction=at least High)	18.18
10	**If** (Area_Covered≥4000) **then** (Overall_Satisfaction=at least High)	27.27
11	**If** (Date_of_Construction≤16) **then** (Overall_Satisfaction=at least Moderate)	71.43
12	**If** (Area_Covered≥1900) **then** (Overall_Satisfaction=at least Moderate)	42.86

Table 7: List of elementary conditions in the minimal cover

Code	Description	Code	Description
C1	Date_of_Construction≤14	C10	Costs_of_Intervention≤920937.37
C2	Date_of_Construction≤16	C11	Costs_of_Intervention≤39400.48
C3	Date_of_Construction≥18	C12	Costs_of_Intervention≤1550000
C4	Area_Covered≤1300	C13	Costs_of_Intervention≥3135879.53
C5	Area_Covered≥1900	C14	Costs_of_Intervention≥3778368.93
C6	Area_Covered≥3000	C15	Costs_of_Intervention≥500000
C7	Area_Covered≥4000	C16	Economic_Self_Sustainability=No
C8	Current_Condition≥Partially_restored	C17	Economic_Self_Sustainability=Yes
C9	Current_Condition≤Partially_restored		

We first applied the decision graph construction algorithm using column splitting solution for binary decision matrix transformation. Algorithm 3 starts by extracting the initial binary decision matrix, given in Table 8. The application of Algorithm 3 led to a decision graph of order (i.e. number of nodes) 26 and size (i.e. number of edges) 31, shown in Figure 7. The 26 nodes of the graph correspond to a decision node r, a 19 condition nodes (corresponding to the 17 elementary conditions in Table 7, plus two nodes corresponding to the duplicate of elementary condition C_3 and C_{16}) and 6 decision nodes. As showing in Figure 7, the condition nodes are labelled with elementary conditions while edges are unlabelled.

Table 8: Initial binary matrix

Rule	Elementary conditions																	Decisions					
																		at most			at least		
#	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	L	M	H	M	H	VH
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
5	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
6	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
7	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
10	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
11	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
12	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

The application of the decision graph construction algorithm using column ordering solution leads to two nonexistent nodes. We then applied the same algorithm using mixed column ordering - column splitting solution for binary decision matrix transformation. This transformation strategy first ranks order the elementary conditions according to the their frequency in the decision rules, starting from the left to the right side. Then, some of most cited elementary conditions are split while the other will stay

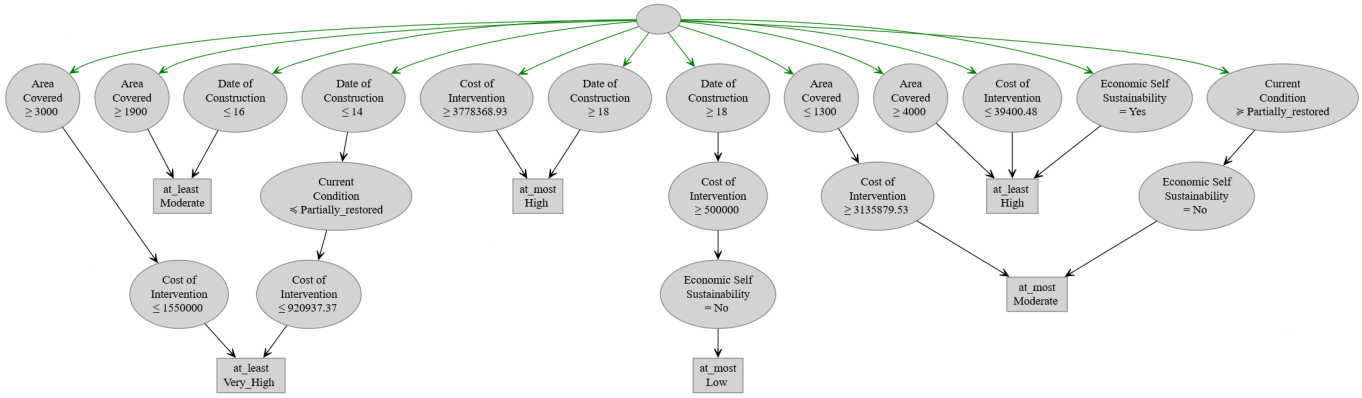


Figure 7: Decision graphs issued from the minimal cover using column splitting strategy

as they are. In this illustrative case study, only elementary conditions C_3 and C_{16} are cited more than once in the decision rules. So, we can either duplicate only the column corresponding to C_3 or only the column corresponding to C_{16}. The first option does not solve the creation of nonexistent decision rules while the second option do and led to a decision graph of order 25 and size 30 as shown in Figure 8. As showing in Figure 7, the condition nodes are labelled with the Left-hand-side (LHS) of elementary conditions while edges are labelled with the comparison operator and Right-hand-side (RHS) of elementary conditions.

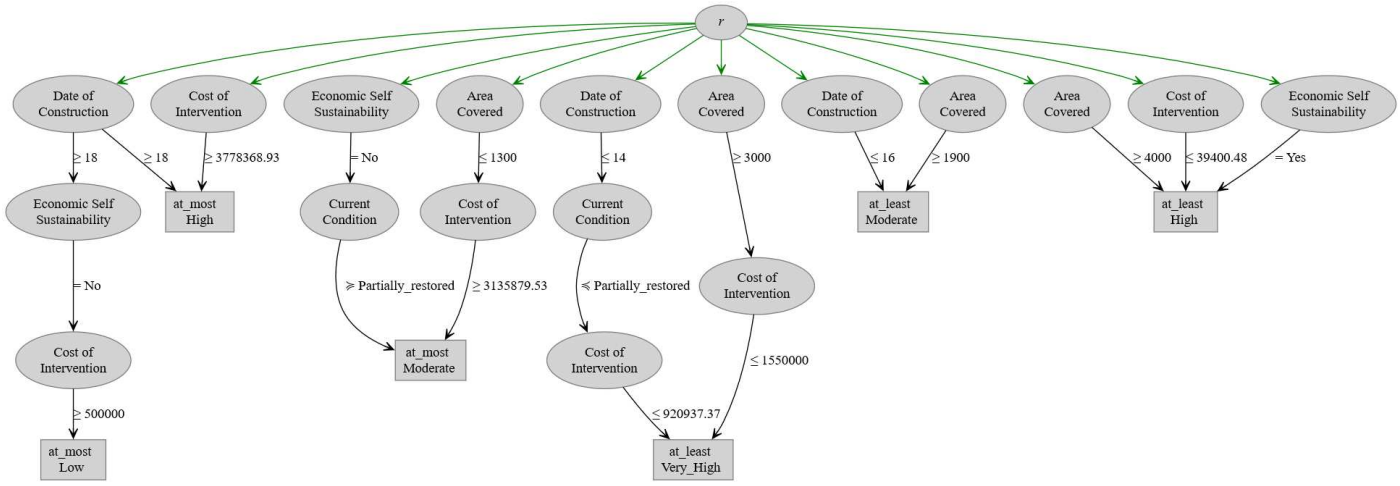


Figure 8: Decision graphs issued from the minimal cover using mixed column ordering - column splitting strategy

6.3.2. Graphical representation of a selection of decision rules

The AllRules algorithm has been used to induce an executive set of decision rules. This resulted in 37 certain decision rules, plus two approximate decision rules. From this list, we selected all certain decision rules with a length of at least 2, i.e., containing at least two elementary conditions. The selected decision rules are shown in Table 9. The list of elementary conditions included in these rules is given in Table 11, while the initial binary decision matrix is given in Table 11.

Applying the decision graph construction algorithm using the column splitting solution for binary decision matrix transformation produces a decision graph with 60 nodes (including a root node and 5 decision nodes) and 76 edges. To reduce the size and complexity of the decision graph, we used a mixed strategy. To minimize the number of nodes to duplicate, we adopted the following approach: first, by visually inspecting the full decision graph, we clustered elementary conditions such that each cluster contains the same elementary condition used in different decision rules of the same type. During clustering, we ensured that no elementary condition was added to a cluster if its inclusion would create nonexistent decision rules. Each cluster is then replaced by a single node, updating the corresponding edges. Figure 9(a) shows the initial subgraph associated with the class union "at most Low" with five clusters (dotted blue line). This initial decision graph contains 18 nodes and 21 edges.

A careful examination of Figure 9(a) shows that the elementary condition "Area_Covered ≤ 1459.56" appears twice but is not clustered because clustering it with the other node would create nonexistent decision rules. It is possible to group these two nodes only if the leftmost elementary condition "Date_of_Construction ≥ 18" is removed from the cluster, which does not reduce the number of nodes or edges. Similarly, the rightmost elementary condition "Current_Condition ≥ Partially_restored" is not included in the cluster with the other two similar conditions to avoid creating nonexistent decision rules.

By replacing each cluster with a single node and updating the edges, we obtain the reduced subgraph shown in Figure 9(b), containing 10 nodes and 13 edges. The same clustering process has been applied to all class unions, leading to the decision graph shown in Figure 10, which contains 24 nodes and 55 edges, a substantial reduction compared to the initial graph obtained using the column splitting strategy.

Table 9: List of selected decision rules

Rule #	Rule Description	Strength (%)
1	**If** (Date_of_Construction≥18) & (Area_Covered≤1459.56) & (Costs_of_Intervention≥500000) **then** (Overall_Satisfaction at most Low)	100
2	**If** (Date_of_Construction≥18) & (Current_Condition≥Partially_restored) & (Costs_of_Intervention≥500000) **then** (Overall_Satisfaction at most Low)	100
3	**If** (Date_of_Construction≥18) & (Current_Condition≥Partially_restored) & (Economic_Self_Sustainability≤No) **then** (Overall_Satisfaction at most Low)	100
4	**If** (Date_of_Construction≥18) & (Costs_of_Intervention≥500000) & (Economic_Self_Sustainability≤No) **then** (Overall_Satisfaction at most Low)	100
5	**If** (Area_Covered≤1459.56) & (Current_Condition≥Partially_restored) & (Costs_of_Intervention≥500000) & (Economic_Self_Sustainability≤No) **then** (Overall_-Satisfaction at most Low)	100
6	**If** (Area_Covered≤150) & (Current_Condition≥Partially_restored) **then** (Overall_Satisfaction at most Moderate)	20
7	**If** (Area_Covered≤150) & (Costs_of_Intervention≥100105.13) **then** (Overall_Satisfaction at most Moderate)	20
8	**If** (Current_Condition≥Partially_restored) & (Economic_Self_Sustainability≤No) **then** (Overall_Satisfaction at most Moderate)	80
9	**If** (Date_of_Construction≥16) & (Current_Condition≥Partially_restored) & (Costs_of_Intervention≥100105.13) **then** (Overall_Satisfaction at most Moderate)	60
10	**If** (Date_of_Construction≥16) & (Costs_of_Intervention≥100105.13) & (Economic_Self_Sustainability≤No) **then** (Overall_Satisfaction at most Moderate)	60
11	**If** (Area_Covered≤1459.56) & (Costs_of_Intervention≥500000) & (Economic_Self_Sustainability≤No) **then** (Overall_Satisfaction at most Moderate)	60
12	**If** (Area_Covered≤1720) & (Costs_of_Intervention≥1179793.2) & (Economic_Self_Sustainability≤No) **then** (Overall_Satisfaction at most Moderate)	60
13	**If** (Date_of_Construction≥13) & (Costs_of_Intervention≥3135879.53) **then** (Overall_Satisfaction at most Moderate)	40
14	**If** (Area_Covered≤1300) & (Costs_of_Intervention≥3135879.53) **then** (Overall_Satisfaction at most Moderate)	40
15	**If** (Current_Condition≥Partially_restored) & (Costs_of_Intervention≥1179793.2) **then** (Overall_Satisfaction at most High)	25
16	**If** (Area_Covered≥3000) & (Costs_of_Intervention≤1550000) **then** (Overall_Satisfaction at least Very_High)	50
17	**If** (Date_of_Construction≤15) & (Current_Condition≤Restored) & (Costs_of_Intervention≤1550000) **then** (Overall_Satisfaction at least Very_High)	50
18	**If** (Date_of_Construction≤14) & (Current_Condition≤Partially_restored) & (Costs_of_Intervention≤920937.37) **then** (Overall_Satisfaction at least Very_High)	50
19	**If** (Date_of_Construction≤14) & (Costs_of_Intervention≤920937.37) **then** (Overall_Satisfaction at least High)	18.18
20	**If** (Current_Condition≤Restored) & (Costs_of_Intervention≤2473914.97) **then** (Overall_Satisfaction at least High)	54.55
21	**If** (Date_of_Construction≤11) & (Current_Condition≤Restored) **then** (Overall_Satisfaction at least High)	27.27

Table 10: List of elementary conditions in the selected decision rules

Code	Description	Code	Description
1	Date_of_Construction ≤ 11	12	Current_Condition ≤ Partially_restored
2	Date_of_Construction ≤ 14	13	Current_Condition ≤ Restored
3	Date_of_Construction ≤ 15	14	Current_Condition ≥ Partially_restored
4	Date_of_Construction ≥ 18	15	Costs_of_Intervention ≤ 1550000
5	Date_of_Construction ≥ 16	16	Costs_of_Intervention ≤ 2473914.97
6	Date_of_Construction ≥ 17	17	Costs_of_Intervention ≤ 920937.37
7	Area_Covered ≤ 1300	18	Costs_of_Intervention ≥ 100105.13
8	Area_Covered ≤ 1459.56	19	Costs_of_Intervention ≥ 1179793.2
9	Area_Covered ≤ 150	20	Costs_of_Intervention ≥ 3135879.53
10	Area_Covered ≤ 1720	21	Costs_of_Intervention ≥ 500000
11	Area_Covered ≥ 3000	22	Economic_Self_Sustainability = No

Table 11: Initial binary matrix

Rule	Elementary conditions																						Decisions: at most			Decisions: at least	
#	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	L	M	H	H	VH
1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0
3	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0
6	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0
9	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0
10	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0
11	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0
12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0
13	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0
14	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0
16	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
17	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1
18	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
19	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0
20	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0
21	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0

6.4. Linearisation and exploitation of decision graphs

In this section, we illustrate the proposed decision graph linearisation and exploitation algorithms. For the exploitation phase, we first apply the exploitation algorithm to reclassify the data used to generate the minimal cover. We then apply the same algorithm to a new, unseen set of projects.

6.4.1. Linearisation

We applied the decision graph linearisation algorithm BFSL to the decision graph shown in Figure 8 to extract all paths from the root to the decision nodes, with each path corresponding to a decision rule. The resulting rules are listed in Table 12. Comparing these rules with the input table used to construct the decision graph, we observe that they contain exactly the same set of decision rules; the only differences are the ordering of the rules and the ordering of elementary conditions within each rule. This behaviour is expected: during linearisation, conditions are appended according to their position along the path in the decision graph. For example, a rule such as *If (Date_of_Construction ≥ 18) & (Economic_Self_Sustainability = No) & (Costs_of_Intervention ≥ 500000)* may appear as *If (Costs_of_Intervention ≥ 500000) & (Date_of_Construction ≥ 18) & (Economic_Self_Sustainability =*

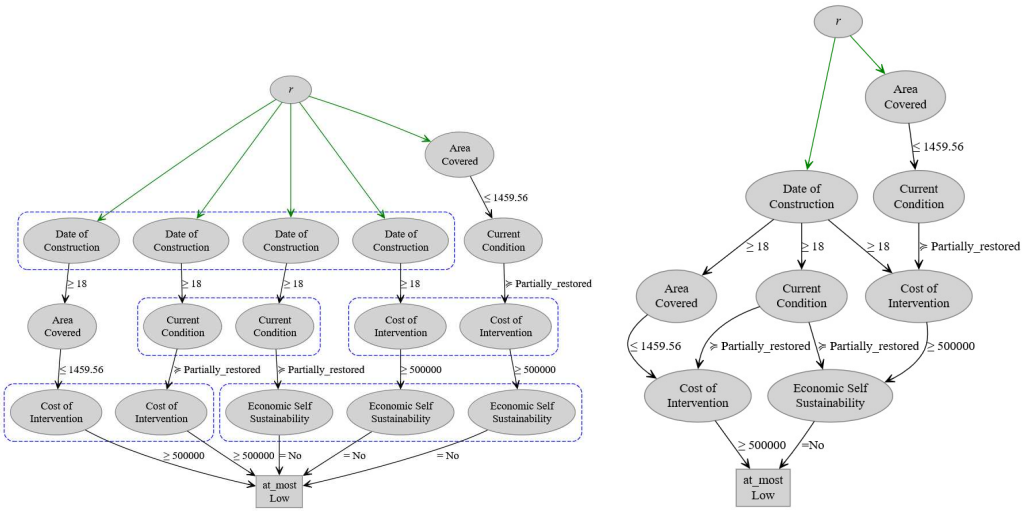


Figure 9: Initial (a) and reduced (b) subgraphs associated with class union “at most Low”

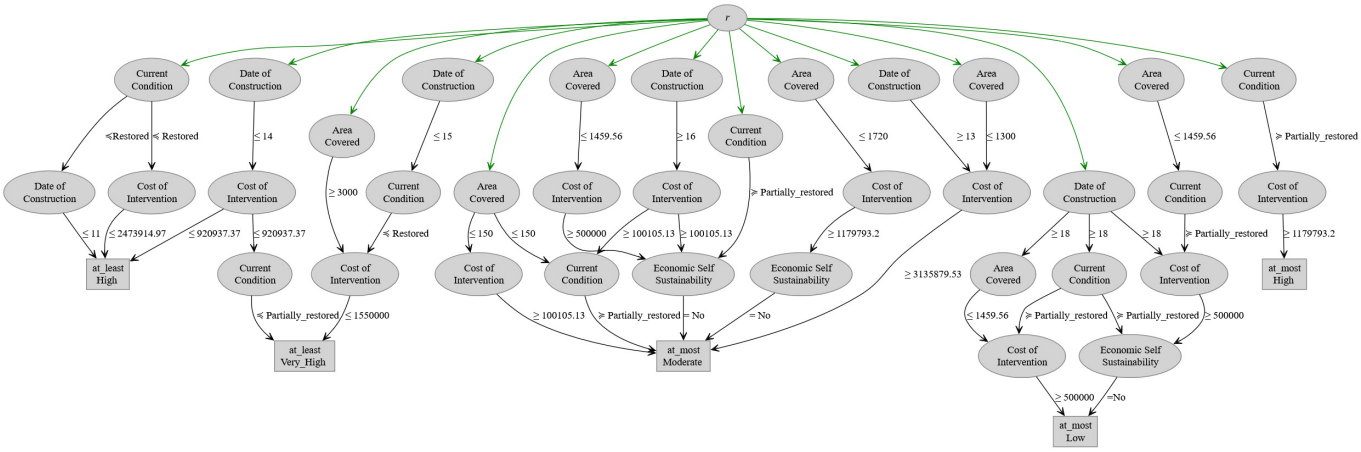


Figure 10: Decision graph associated with selected decision rules

No). Since all conditions are conjunctive, their order does not affect the logical meaning of the rule. Consequently, both tables express the same knowledge.

Table 12: Results of linearisation by BFSL algorithm applied on decision graph in Figure 8

Rule #	Description
1	**If** (Date_of_Construction ≥ 18) & (Economic_Self_Sustainability = No) & (Costs_of_Intervention ≥ 500000) **then** (Overall_Satisfaction = at most Low)
2	**If** (Economic_Self_Sustainability ≤ No) & (Current_Condition ≥ Partially_restored) **then** (Overall_Satisfaction = at most Moderate)
3	**If** (Costs_of_Intervention ≥ 3135879.53) & (Area_Covered ≤ 1300) **then** (Overall_Satisfaction = at most Moderate)
4	**If** (Costs_of_Intervention ≥ 3778368.93) **then** (Overall_Satisfaction = at most High)
5	**If** (Date_of_Construction ≥ 18) **then** (Overall_Satisfaction = at most High)
6	**If** (Area_Covered ≥ 3000) & (Costs_of_Intervention ≤ 1550000) **then** (Overall_Satisfaction = at least Very High)
7	**If** (Costs_of_Intervention ≤ 920937.37) & (Date_of_Construction ≤ 14) & (Current_Condition ≤ Partially_restored) **then** (Overall_Satisfaction = at least Very High)
8	**If** (Economic_Self_Sustainability ≥ Yes) **then** (Overall_Satisfaction = at least High)
9	**If** (Costs_of_Intervention ≤ 39400) **then** (Overall_Satisfaction = at least High)
10	**If** (Area_Covered ≥ 4000) **then** (Overall_Satisfaction = at least High)
11	**If** (Date_of_Construction ≤ 16) **then** (Overall_Satisfaction = at least Moderate)
12	**If** (Area_Covered ≥ 1900) **then** (Overall_Satisfaction = at least Moderate)

6.4.2. Re-classification of learning set using decision graphs

We applied the decision graph exploration algorithm BFSE to reclassify the learning set, which was used to generate the decision rules in Table 6. The results are presented in Table 13. For each project, the table lists the supporting decision rules from Table 12 and the final decision computed using the decision aggregation strategy introduced in Section 5.3. Comparing the initial decisions with those obtained by BFSE shows a 100% agreement.

To illustrate the behaviour of the aggregation procedure under different situations, we consider four representative projects from the decision graph: one covered exclusively by at-most rules, one covered exclusively by at-least rules, and two covered by mixed sets of rules. The rule numbers referenced correspond to the rule base in Table 12.

The assignment examples are as follows:

Table 13: Decisions obtained by BFSE algorithm for projects in the learning set

Project	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Supporting rules	6,8,11,12	5,8	7,8,11	2,11	1,2,5	2,11	8,9,11	8,10,11,12	1,2,3,5	8,11	5,8,12	2,5,9	4,8,10,11,12	2,3,11	2,4,10,11,12	5,8,12
Final decision	Very_High	High	Very_High	Moderate	Low	Moderate	High	High	Low	High	High	High	High	Moderate	High	High

- Project 5: only at-most rules. Project 5 is covered exclusively by at-most rules: Rule 1 (*at_most Low*), Rule 2 (*at_most Moderate*), and Rule 5 (*at_most High*). Since all covering rules are of the same type, the aggregation strategy follows the classical at-most interpretation: the final assignment is the *lowest* decision value among the suggested rules. Project 5 is assigned to Low.
- Project 3: only at-least rules. Project 3 satisfies three at-least rules: Rule 7 (*at_least Very High*), Rule 8 (*at_least High*), and Rule 11 (*at_least Moderate*). As all rules are at-least, the assignment is obtained by selecting the *highest* decision value supported by the rules. Project 3 is assigned to Very High.
- Project 2: mixed rules with a unique admissible class. Project 2 is covered by one at-most rule and one at-least rule: Rule 5 (*at_most High*) and Rule 8 (*at_least High*). The aggregation identifies the lowest admissible class from the at-least rules and the highest from the at-most rules. Both bounds coincide at *High*, giving a unique assignment of High.
- Project 15: mixed rules with a non-collapsing interval. Project 15 is covered by both at-most and at-least rules. The at-most rules are Rule 2 (*at_most Moderate*) and Rule 4 (*at_most High*); the at-least rules are Rule 10 (*at_least High*), Rule 11 (*at_least Moderate*), and Rule 12 (*at_least Moderate*). The lowest admissible class from the at-least rules is *Moderate*, while the highest from the at-most rules is *High*, producing the interval [Moderate,High]. Applying an aggregation operator (optimistic choice) selects the upper bound, assigning Project 15 to High.

The first three cases correspond to homogeneous at-most rules, homogeneous at-least rules, and a mixed configuration where the admissible interval collapses to a single class. The final case illustrates a genuinely mixed situation with an interval containing more than one admissible class, highlighting the influence of the chosen aggregation operator on the final assignment.

6.4.3. Assignment of unseen projects

Having evaluated the behaviour of the exploitation procedure on the learning set, we now apply it to new, unseen projects. This step assesses the robustness of the decision graph and the generalisability of the induced rule base beyond the data used to construct the minimal cover. The unseen projects and their scores with respect to the criteria in Table 5 are presented in Table 14. These projects were selected and scored by the authors with the support of local experts. As they are preliminary projects that have not yet been realised, they were not considered in earlier stages of the analysis.

Table 14: Characteristics of new adaptive reuse projects

Project #	Name	Current reuse process	Criteria				
			Date of Construction (A1)	Area Covered (A2)	Current Condition (A3)	Costs of Intervention (A4)	Economic Self-Sustainability (A5)
17	Palazzo San Massimo	Not in use	9	4500	To_be_restored	9900000	Yes
18	Convento di San Francesco	Not in use	13	8500	To_be_restored	18700000	Yes
19	Convento di Santa Maria della Consolazione	Not in use	16	3900	To_be_restored	8580000	Yes
20	Convento di San Pietro a Maiella e San Giacomo	Not in use	14	2500	To_be_restored	5500000	Yes

Applying the BFSE algorithm to this set yields the classifications reported in Table 15. The following explanations clarify these decisions:

- Project 17: mixed at-most and at-least rules (non-collapsing interval). Project 17 is covered by one at-most rule and four at-least rules. The at-most rule is Rule 4 (*at_most High*). The at-least rules are Rules 8 and 10 (*at_least High*) and Rules 11 and 12 (*at_least Moderate*). The lowest admissible class from the at-least rules is *Moderate*, and the highest from the at-most rule is *High*, producing the interval [Moderate,High]. Applying the optimistic operator, the final assignment is High.
- Project 18: mixed at-most and at-least rules (non-collapsing interval). Project 18 satisfies the same set of rules as Project 17. The lowest admissible class suggested by the at-least rules is *Moderate*, and the highest from the at-most rule is *High*, yielding the interval [Moderate,High]. Using the optimistic aggregation operator, the final assignment is High.
- Project 19: mixed at-most and at-least rules (non-collapsing interval). Project 19 is covered by one at-most rule (Rule 4) and three at-least rules (Rules 8, 11, and 12). The lowest admissible class from the at-least rules is *Moderate*, and the highest from the at-most rule is *High*, giving the interval [Moderate,High]. The final assignment, using the optimistic operator, is High.
- Project 20: mixed at-most and at-least rules (non-collapsing interval). Project 20 satisfies the same set of rules as Project 19. The admissible interval is [Moderate,High], and applying the optimistic aggregation operator assigns Project 20 to High.

Table 15: Decisions obtained by the BFSE algorithm for unseen projects

Project	17	18	19	20
Supporting rules	4, 8, 10, 11, 12	4, 8, 10, 11, 12	4, 8, 11, 12	4, 8, 11, 12
Final decision	High	High	High	High

A comparison of the four unseen projects reveals a consistent pattern in the decision graph's behaviour. Despite differences in attribute values, they activate largely the same subset of decision rules, with all four satisfying Rule 4 (*at_most High*) along with a combination of at-least rules, mainly Rules 8, 11, and 12, and in two cases also Rule 10. The resulting admissible interval is consistently [Moderate,High], with the lower bound imposed by at-least rules and the upper bound by the at-most rule. Since the interval does not collapse to a single class, the final assignment depends on the aggregation operator. Applying the default maximum operator assigns all four projects to the class *High*, illustrating both the stability of the decision graph and the coherent influence of the rule base when applied to previously unseen cases.

6.5. Discussion of results

The decision graph's performance in decision-making scenarios, especially in the context of adaptive reuse projects, reveals a solid foundation for decision support systems. Its application to real-world cultural heritage data from Europe illustrates its practicality, with expert feedback emphasising its advantages. Decision graphs have been found to reduce cognitive effort, improve communication, and provide a clear and accessible complement to traditional textual and tabular representations of decision rules. In terms of practical decision-making, the decision graph offers a solid framework for consistently evaluating complex scenarios.

Furthermore, this application also demonstrated the effectiveness and robustness of decision graph models for classifying and exploiting decision rules in complex systems. The linearisation process, using the BFSL algorithm, successfully extracted the decision rules while preserving their logical integrity, providing a transparent and interpretable decision-making process. The reclassification of the learning set showed a perfect match with initial decisions, highlighting the accuracy and consistency of the decision graph approach. When applied to unseen projects, the decision graph classified successfully all projects, showing its ability to generalise well to new cases.

7. Comparison and discussion

This section compares the proposed decision graph model with existing graph-based models for representing decision rules. It examines their suitability for representing DRSA-induced rules, introduces formal measures for evaluating structural fidelity, complexity, interpretability, and computational performance, and presents comparative analyses of their structural and semantic capabilities.

7.1. Representation of DRSA rules using existing graph models

As highlighted earlier, the graph models discussed in Section 2 were not originally conceived to represent if–then decision rules induced from dominance-based rough approximation. Their primary design objectives lie elsewhere, which makes their direct application to DRSA-derived rules problematic. Nevertheless, certain models can be adapted to serve this purpose, albeit with varying degrees of suitability. Fast-and-frugal trees and fault trees are the most amenable to adaptation, while reliability block diagrams and binary decision diagrams offer partial applicability. In contrast, decision trees extracted from decision tables are inherently limited, as they cannot directly operate on decision rules. Similarly, binary decision diagrams are grounded in Boolean functions, and their use for representing if–then rules requires non-trivial transformations that reduce transparency and interpretability.

To illustrate these limitations and adaptations, consider the following set of decision rules R_1 to R_4 introduced in Example 1, where C_i (i = 1,...,5) denotes an elementary condition and K_j represents the decision outcome:

- R_1: If C_1 and C_3, then K_1
- R_2: If C_2, then K_1
- R_3: If C_3 and C_4, then K_2
- R_4: If C_5, then K_2

Graph-based representations of these rules using fast-and-frugal trees, fault trees, reliability block diagrams, and decision trees are shown in Figure 11. The fast-and-frugal tree corresponding to rules R_1 to R_4 is presented in Figure 11(a). Each decision is represented in a separate fast-and-frugal tree, with a root node added to link them together. The principles of fast-and-frugal trees do not permit the representation of multiple decision rules within a single tree. This naturally increases the size of the graph and the time required for its exploration.

The use of a fault tree results in a more compact representation, as illustrated in Figure 11(b). However, fault trees require the addition of fictitious events because, according to their principles, two logic gates cannot be directly connected. Figure 11(b) therefore includes two fictitious events, namely C_a and C_b. The binary decision diagram for rules R_1 to R_4 is shown in Figure 11(c), providing the most compact representation. Figure 11(d) shows a tentative representation of decision rules R_1–R_4 as a decision tree. Although it preserves the assumptions of a classical decision tree, it requires complement conditions through negative

branches. For example, the branch $C_1 = no$ implicitly introduces the condition $\neg C_1$, even though this condition is absent from the original rule set. Consequently, the original rule semantics are not preserved exactly. Moreover, the resulting tree may produce incorrect classifications. For example, an object satisfying C_1, C_3, and C_4 is covered by both rules R_1 and R_3. However, in the tree representation, the path to K_2 is reachable only through the branch $C_1 = no$. The object therefore cannot reach the leaf associated with K_2, despite satisfying the antecedent of R_3. Thus, the representation changes the coverage of the original rules and may produce classifications that differ from those obtained directly from the rule set. It should also be noted that this is not a classical decision tree induced from data, but a tree-shaped encoding of the rule set. Consequently, decision-tree learning mechanisms such as the Gini index, information gain, pruning, and recursive partitioning of the instance space are not applicable.

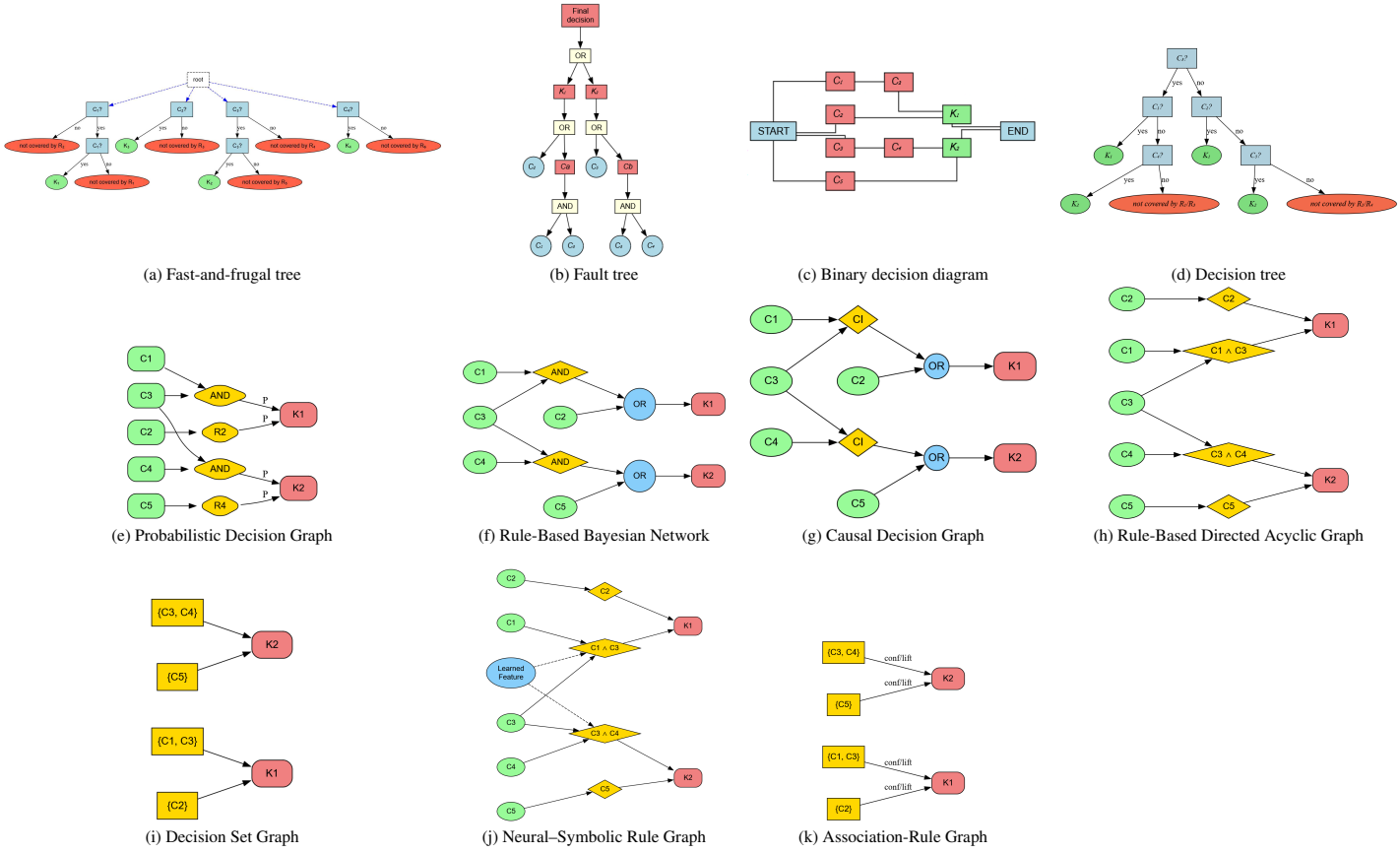


Figure 11: Representation of decision rules using adapted graph models

The graph models introduced in Section 2 were also used to represent decision rules R_1–R_4. In their standard forms, these models are not designed to explicitly capture the conjunctive and disjunctive semantics of rule-based classifiers, as they primarily represent probabilistic dependencies, causal relationships, associative patterns, or structural connections. Therefore, additional auxiliary constructs are introduced to enable a consistent comparison while preserving the original rule semantics. The Probabilistic Decision Graph (PDG) (Figure 11(e)) uses auxiliary nodes to represent conjunctive antecedents and disjunctive combinations. The Rule-Based Bayesian Network (RBN) (Figure 11(f)) introduces deterministic AND/OR nodes to make implicit rule structures explicit. The Causal Decision Graph (CDG) (Figure 11(g)) uses intermediate causal nodes to represent conjunctive conditions. The Rule-Based Directed Acyclic Graph (RB-DAG) (Figure 11(h)) applies rule aggregation nodes while preserving acyclic evaluation. The Decision Set Graph (DSG) (Figure 11(i)) represents each conjunctive antecedent as a rule node, with disjunctions arising from multiple paths to the same decision. The Neural–Symbolic Rule Graph (NSRG) (Figure 11(j)) encodes symbolic expressions through conjunctive rule nodes, while the Association-Rule Graph (ARG) (Figure 11(k)) introduces intermediate rule nodes to aggregate antecedents before reaching decisions.

7.2. Comparative evaluation of graph-based models

While the illustrative example highlights the practical challenges of adapting existing graph models to represent DRSA rules, a broader comparative perspective is required. Table 16 provides a systematic comparison of the graph-based models introduced in Section 2 and the proposed Decision Graph framework. The table extends the comparative analysis originally presented in Díaz-Ramírez et al. (2023) by evaluating models across five categories: (i) purpose and application, (ii) structural representation, (iii) analytical capabilities, (iv) construction methods, and (v) benefits and limitations.

These categories cover key aspects including node and leaf representations, qualitative and quantitative analysis capabilities, cut set computation, construction approaches, and model strengths and limitations. The comparison highlights that the proposed Decision Graph framework is specifically designed to capture complex rule interactions, aggregate multiple decision rules, and provide transparent decision support. Unlike existing models, it combines interpretability with flexibility, supporting both qualitative and quantitative analysis. Although its complexity increases with rule-base size, Decision Graphs provide a faithful representation of DRSA-induced rules while achieving a balance between compactness, transparency, and analytical capability.

Table 16: Comparison of decision graph model with existing graph models (adapted and extended from Jimenez-Roa et al. (2021))

Category	Aspect	Fast-and-Frugal Trees	Fault Trees	Reliability Block Diagrams	Decision Trees	Binary Decision Diagrams	Decision Graphs
(i) Purpose and application	–	Quick decision-making; psychology; management; medicine; finance	Reliability engineering; root cause analysis; decision-making	System reliability assessment; redundancy analysis; failure/success modelling	Classification; regression; decision-making	Model checking; protocol validation; verification	Knowledge extraction; rule induction; decision support; aggregation of rules
(ii) Structural representation	(ii.a) Structure	Sequential binary tree; simple hierarchical graph	Top-down directed acyclic graph	Blocks in series/parallel paths; network representation	Hierarchical top-down graph	Directed acyclic graph	Directed acyclic graph with rule nodes and aggregation operators
	(ii.b) Main symbols	Binary questions (yes/no); exit nodes	Top/intermediate/basic events; logic gates	Blocks; series/parallel connectors; switches	Root / decision / leaf nodes	Root / transitional nodes	Rule nodes; decision nodes; aggregation operators
	(ii.c) Leaves information	Categorical outcomes (decision exits)	Failure frequency / probability density function	System success/failure states; reliability values	Numeric & nominal data	Boolean data {0,1}	Decision classes (Low, Moderate, High, Very High)
	(ii.d) No. of children	2 (yes/no)	Multiple	Multiple (series/parallel)	2*	2	Variable (depends on rule coverage)
	(ii.e) Sub-node sharing?	No	Yes	Yes	No	Yes	Yes
	(ii.f) Gates/Nodes information	Binary conditions (thresholds, yes/no)	Logic rules (AND, OR, VOT)	Series/parallel reliability relations	Attribute splitting criteria	Binary splitting criteria	Decision rules; aggregation strategy
(iii) Analysis	(iii.a) Direction	Top-down (sequential questions)	Bottom-up	Bottom-up (components to system)	Top-down	Top-down	Top-down and bottom-up (rule evaluation and aggregation)
	(iii.b) Type	Qualitative (fast heuristics)	Qualitative & quantitative	Quantitative (probabilistic reliability)	Quantitative	Qualitative & quantitative	Qualitative & quantitative (rule-based aggregation)
	(iii.c) Cut sets computation	No	Yes	Yes	Yes**	Yes	Yes
	(iii.d) Extensions	Adaptive FFTs; multi-exit FFTs	Dynamic FT; repairable FT; attack FT; state/event FT	Dynamic RBD; success trees	Causal DT; clustering DT; binary DT; survival DT; oblivious DT	Ordered BDD; reduced OBDD	Extended decision graphs with fuzzy rules; probabilistic decision graphs
(iv) Construction	(iv.a) Common way	Expert-designed; simple heuristics	Manually built; expert judgement	Expert modelling; reliability data	Data-driven (supervised/unsupervised learning)	Through Boolean functions	Data-driven or expert-driven rule induction; aggregation strategies
	(iv.b) Induction algorithms	Heuristic ordering of questions; exit rule design	IFT, LIFT, DDFT, ILTA	Reliability estimation; Monte Carlo simulation	ID3, C4.5, CART	Shannon expansion formula; IFT method	Rule induction algorithms; graph linearisation; BFSE/BFSL
(v) Benefits and limitations	(v.a) Benefits	Fast and frugal; low information cost; simple to use	Dependability analysis; reliability metrics	Clear visualisation of redundancy; supports reliability computation	Intuitive and easy to follow; easily derived from data	Boolean algebra enables faster computations	Captures complex rule interactions; supports aggregation; flexible decision support
	(v.b) Limitations	Lower accuracy; limited complexity handling	Subjective; assumes statistical independence	Assumes independence unless extended; complex for large systems	Changes in root node can generate new structure	Constrained to Boolean data; root changes regenerate BDD	Complexity grows with rule base; aggregation choice may affect outcomes

Notes: * Except when using multivariate splitting criteria, then it is > 2.
** Only when the DT maps to classes *no failure* and *failure*.

To further extend this analysis, Table 17 compares recent graph-based models for representing if–then rules, including probabilistic, causal, deterministic, and hybrid approaches. Unlike the previous comparison of general decision-oriented graphs, this analysis focuses on rule representation, structural characteristics, construction principles, and preservation of rule semantics. Existing models effectively represent dependencies, uncertainty, causality, and rule interactions; however, they generally lack explicit support for dominance relations, preference-ordered decision classes, and monotonicity constraints. This motivates the need for a dedicated graph-based model that represents dominance-based decision rules while maintaining interpretability, compactness, and semantic fidelity.

Table 17: Characteristics of recent graph-based models capable of representing if–then rules

Category	Aspect	PDGs	RBBNs	CDGs	RB-DAGs	DSGs	NSRGs / ARGs
(i) Purpose and application	–	Probabilistic inference; uncertainty modelling; decision support	Rule-based probabilistic reasoning; Bayesian inference over symbolic knowledge	Causal reasoning; intervention and counterfactual analysis	Compact representation of overlapping deterministic rules	Compact representation and analysis of unordered rule sets	Neural-symbolic reasoning; association analysis and pattern discovery
(ii) Structural representation	Structure	Probabilistic directed acyclic graph	Bayesian network with rule-derived conditional probability tables	Causal directed acyclic graph based on structural relationships	Deterministic directed acyclic graph of conditions and decisions	Graph/hypergraph representation of rule conditions and decisions	Predicate graph / weighted rule graph
	Nodes	Variables and probabilistic states	Variables and conditional probability tables	Causal variables and causal factors	Condition nodes and decision nodes	Condition nodes and decision nodes	Predicates, learned features, or itemsets
	Leaves	Probabilistic outcomes	Probabilistic decisions	Causal outcomes	Deterministic decision nodes	Decision nodes	Decisions or itemsets
	Sub-node sharing?	Yes	Yes	Yes	Yes	Yes	Yes
(iii) Analysis	Type	Quantitative (probabilistic)	Quantitative (probabilistic)	Qualitative and quantitative (causal)	Qualitative (deterministic rule analysis)	Qualitative (rule-set analysis)	Qualitative and quantitative (symbolic or weighted analysis)
	Supports rule fidelity?	Partial (probabilistic representation may alter deterministic semantics)	Partial (rules require probability assignments)	No (focuses on causal influence rather than rule semantics)	Yes (preserves deterministic rule paths)	Partial (does not preserve rule ordering)	Partial (depends on extracted or weighted representations)
(iv) Construction	Common method	Probabilistic parameterisation and conditional probability specification	Translation of symbolic rules into conditional probability tables	Causal modelling and structural equation specification	Direct encoding of deterministic rules	Encoding of rule sets and condition interactions	Symbolic extraction, learning, or association mining
	Requires probabilities?	Yes	Yes	No	No	No	No
(v) Benefits and limitations	Benefits	Represents uncertainty and compact probabilistic dependencies	Combines symbolic rules with probabilistic inference	Supports causal analysis and intervention reasoning	Compact representation of deterministic rules	Flexible representation of overlapping rules	Supports hybrid reasoning and weighted relationships
	Limitations	Requires probability distributions; does not preserve deterministic semantics	Requires probability assignments; may distort original rule meaning	Assumes causal relationships; does not represent preference ordering	Does not encode dominance relations or monotonicity constraints	Represents unordered rules; lacks dominance-based semantics	Reduced interpretability or absence of preference-ordered decision classes

Although these graph-based models provide effective mechanisms for representing, analysing, and visualising different types of rules, they are not designed to capture the specific characteristics of dominance-based if-then decision rules. In particular, they do not explicitly represent preference-ordered decision classes, monotonicity constraints, or dominance relations. Representing dominance-based if-then decision rules within these formalisms therefore requires additional assumptions and auxiliary constructs to preserve their conjunctive and disjunctive semantics while maintaining interpretability. These constructs serve only to improve transparency and should not be considered extensions or modifications of the original models. Consequently, existing graph formalisms provide useful foundations but lack a dedicated representation for dominance-based if-then decision rules.

7.3. Evaluating the effectiveness of decision graphs and comparing graph-based rule models

This section proposes quantitative measures to evaluate decision graphs. In what follows, $G = (V,E)$ denotes a directed graph representing a rule base, where V is the set of nodes and E the set of edges. Let $\mathcal{R}$ denote the original rule set, $\mathcal{P}$ the set of root-to-decision paths extracted from the graph, and C the set of unique conditions.

The proposed measures are organised into four complementary categories:

- **Structural fidelity measures**. Structural fidelity evaluates how faithfully a graph preserves the original rule base. Three measures are defined:

$$\mathrm{RPF} = \frac{|\mathcal{P} \cap \mathcal{R}|}{|\mathcal{P}|}, \tag{17}$$

$$\mathrm{SRR} = \frac{|\mathcal{P} \setminus \mathcal{R}|}{|\mathcal{P}|}, \tag{18}$$

$$\mathrm{RRC} = \frac{|\mathcal{P} \cap \mathcal{R}|}{|\mathcal{R}|}. \tag{19}$$

  Rule–Path Fidelity (RPF) measures the proportion of graph paths corresponding to original rules, Spurious Rule Rate (SRR) measures the proportion of invalid paths, and Rule Representation Completeness (RRC) measures the proportion of original rules represented by the graph.

- **Graph complexity and compactness measures**. Let $\mathcal{P}$ denote the set of graph paths, $\ell(p)$ the length of path p, L_i the set of nodes at level i, and C' the total number of condition occurrences.

$$\mathrm{NC} = |V|, \tag{20}$$

$$\mathrm{EC} = |E|, \tag{21}$$

$$\mathrm{GD} = \max_{p \in \mathcal{P}} \ell(p), \tag{22}$$

$$\mathrm{GW} = \max_{i} |L_i|, \tag{23}$$

$$\mathrm{RR} = \frac{|C'| - |C|}{|C'|}. \tag{24}$$

  NC, EC, GD and GW measure graph size and shape, whereas RR quantifies repeated condition occurrences, with lower values indicating less redundancy.

- **Interpretability and cognitive complexity measures**. Let $\deg^+(v)$ denote the out-degree of node v and $\mathcal{S}$ the set of unique condition subsets.

$$\mathrm{APL} = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \ell(p), \tag{25}$$

$$\mathrm{CRF} = \frac{|C'|}{|C|}, \tag{26}$$

$$\mathrm{VD} = \frac{|E|}{|V|}, \tag{27}$$

$$\mathrm{BF} = \frac{1}{|V|} \sum_{v \in V} \deg^+(v), \tag{28}$$

$$\mathrm{CRO} = \frac{|\{s \in \mathcal{S} : s \text{ appears in more than one rule}\}|}{|\mathcal{S}|}. \tag{29}$$

  APL measures average decision path length, CRF the average reuse of conditions, VD the average edge density, BF the average branching, and CRO the proportion of condition subsets shared across rules.

- **Algorithmic performance measures**. Let $\mathcal{R}_{lin}$ denote the rule set after graph linearisation, U_{test} the test dataset, $\hat{d}(u)$ the predicted decision of object u, and $d(u)$ its true decision.

$$\mathrm{GTC} = O(|V| + |E|), \tag{30}$$

$$\mathrm{LL} = \frac{|\mathcal{R}| - |\mathcal{R}_{lin}|}{|\mathcal{R}|}, \tag{31}$$

$$\mathrm{CC} = \frac{|\{u \in U_{test} : \hat{d}(u) = d(u)\}|}{|U_{test}|}. \tag{32}$$

  Graph Traversal Complexity (GTC) denotes graph traversal complexity, Linearisation Loss (LL) measures information lost during linearisation, and Classification Consistency (CC) measures agreement between graph-based and rule-based classifiers.

Appendix E provides a detailed explanation of these measures. To illustrate the application of these measures, consider a rule base containing N rules $R_i : C_{i1} \wedge C_{i2} \wedge \cdots \wedge C_{ip_i} \to K_j$, where $1 \le p_i \le M$. Let K denote the number of decision classes. The average number of conditions per rule and the total number of condition occurrences are then defined as follows:

$$\bar{p} = \frac{1}{N} \sum_{i=1}^{N} p_i, \tag{33}$$

$$C' = \sum_{i=1}^{N} p_i = N\bar{p}. \tag{34}$$

For rule-preserving representations such as RB-DAGs and Decision Graphs, RPF = 1 and LL = 0, as all original rules are preserved without information loss during linearisation. Under these assumptions, the redundancy ratio can be expressed as

$$\mathrm{RR} = \frac{N\bar{p} - C}{N\bar{p}}. \tag{35}$$

Higher RR values indicate greater potential for graph compression through condition sharing.

Tables 18 and 19 summarise the structural, computational and semantic characteristics of representative graph-based rule models. Rule-preserving approaches such as RB-DAGs and Decision Graphs maintain complete rule fidelity, whereas probabilistic and aggregation-based models improve compactness but generally sacrifice deterministic rule semantics.

Table 18: Generalised quantitative comparison of graph-based rule models under rule-base assumptions

Measure	FFT	FT	RBD	DT	BDD	PDG	RBBN	CDG	RB-DAG	DSG	NSRG	ARG	Decision Graph
RPF	≤ 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	1	≤ 1	≤ 1	≤ 1	1
RRC	Partial	Partial	Partial	Partial	Partial	Partial	Partial	Partial	1	Partial	Partial	Partial	1
SRR	Medium	Medium	Medium	High	Medium	High	High	Medium–High	0	Medium	Medium	Medium	0
NC	$N(\bar{p}+1)$	$N(\bar{p}+1)$	$O(C+K)$	$O(C+K)$	$O(C)$	$C+K$	$C+K$	$C+K$	$N(\bar{p}+1)$	$C+K$	$C+K+L$	$2N$	$C+K+S$
EC	$N\bar{p}$	$N\bar{p}$	$O(C+K)$	$O(C+K)$	$O(C)$	$O(C+K)$	$O(C+K)$	$O(C+K)$	$N(\bar{p}+1)$	N	$O(C+L)$	N	$O(C+K+S)$
GD	$\bar{p}+1$	$\bar{p}$	Low	Medium	Medium	Low	Low	Low	$\bar{p}+1$	Low	Medium	Low	$\bar{p}+1$
APL	$\bar{p}$	$\bar{p}$	Low	Medium	Medium	Low	Low	Low	$\bar{p}+1$	Low	Medium	Low	$\bar{p}+1$
RR	High	Medium	Low	Low	Low	Low	Low	Low	High	High	Medium	Medium	Controlled
CRF	Medium	Medium	High	Medium	High	Low	Low	Low	Medium	High	Medium	Medium	High
VD	Medium	Medium	Low	Medium	Low	Medium	Medium	Medium	High	Low	Medium	Medium	Controlled
BF	Medium	Medium	Low	Medium	Low	Medium	Medium	Medium	High	Low	Medium	Medium	Controlled
CRO	Medium	Medium	High	Low	High	Low	Low	Medium	High	High	Medium	Medium	High
LL	> 0	> 0	> 0	> 0	> 0	> 0	> 0	> 0	0	> 0	> 0	> 0	0
GTC	$O(\|V\|+\|E\|)$	$O(\|V\|+\|E\|)$	$O(\|V\|+\|E\|)$	$O(\|V\|+\|E\|)$	$O(\|V\|+\|E\|)$	$O(\|V\|+\|E\|)$	$O(\|V\|+\|E\|)$	$O(\|V\|+\|E\|)$	$O(\|V\|+\|E\|)$	$O(\|V\|+\|E\|)$	$O(\|V\|+\|E\|)$	$O(\|V\|+\|E\|)$	$O(\|V\|+\|E\|)$

Structural fidelity was verified experimentally. Reclassification of both the learning set (Section 6.4.2) and the unseen dataset (Section 6.4.3) produced identical classifications to those obtained from the original rule sets. The same results were obtained for all graph-construction strategies and both rule sets, demonstrating preservation of rule semantics and classification consistency.

From a semantic perspective, existing graph models generally lack explicit support for preference-ordered decision classes, dominance relations and monotonicity constraints. The proposed Decision Graph framework preserves deterministic rule fidelity while supporting these DRSA concepts together with compact and interpretable graph representations.

Table 19: Semantic capabilities of graph-based rule models with respect to DRSA requirements

Capability	FFT	FT	RBD	DT	BDD	PDG	RBBN	CDG	RB-DAG	DSG	NSRG	ARG	Decision Graph
Deterministic if-then rules	Yes	Partial	Partial	Partial	Partial	Partial	Partial	Partial	Yes	Yes	Partial	Yes	Yes
Rule fidelity	High	Medium	Low–Medium	Low	Medium	Partial	Partial	No	Yes	Partial	Partial	Partial	Yes
Preference-ordered decision classes	No	No	No	No	No	No	No	No	No	No	No	No	Yes
Dominance relations	No	No	No	No	No	No	No	No	No	No	No	No	Yes
Monotonicity constraints	No	No	No	No	No	No	No	No	No	No	No	No	Yes
Rule interaction representation	Limited	Medium	Medium	Limited	High	High	High	High	High	High	Medium	Medium	High
Uncertainty modelling	No	Partial	Yes	No	No	Yes	Yes	No	No	No	Partial	Partial	Optional
Compact representation	Medium	Medium	High	Medium	High	High	Medium	Medium	High	High	High	High	High
Interpretability	High	High	Medium	High	Medium	Medium	Medium	High	High	Medium	Low–Medium	Medium	High

7.4. Theoretical and practical impacts on policy and managerial decisions

From a theoretical perspective, decision graphs contribute to multicriteria decision analysis by providing a formalism for representing and aggregating dominance-based decision rules. They bridge symbolic rule-based reasoning and graphical knowledge representation through a structured model capable of capturing interactions among criteria. Decision graphs support qualitative and quantitative analysis, enabling managers and policymakers to explore decision pathways, identify critical conditions, and understand the logic behind recommendations. This strengthens rule-based decision support and creates opportunities for integrating DRSA with graph-theoretic methods.

Compared with textual or tabular rule representations, decision graphs provide an intuitive and visually coherent structure for navigating complex rule sets. Each root-to-decision path represents a rule, allowing users to trace the sequence of conditions leading to a decision without processing extensive rule lists. This advantage is particularly relevant for large rule bases, where traditional representations may cause information overload. The cultural heritage case study showed that experts perceived decision graphs as easier to interpret and more efficient for practical use.

Decision graphs also provide analytical and operational benefits by supporting efficient navigation, exploration, and analysis through graph algorithms such as Breadth-First Search (BFS). Their structure enables rule retrieval, traversal, linearisation, aggregation, and the identification of shared conditions and recurring decision patterns, offering insights that are difficult to obtain from conventional representations.

In practical terms, decision graphs reduce cognitive effort and improve transparency, allowing decision-makers to better understand and justify recommendations. This is particularly valuable in domains requiring explainability, such as public policy, healthcare, cultural heritage management, and risk assessment. By providing a shared visual representation of decision logic, decision graphs enhance communication, trust, accountability, and decision quality.

8. Conclusion and future work

Dominance-based if–then decision rules are robust preference models extensively used in knowledge representation and classification problems. Although managers and policymakers can easily understand the if–then decision rules derived from dominance-based rough approximations, there is a risk of information overload and confusion when the rule set is large. Inspired by popular

graph models such as decision trees, fast-and-frugal trees, and fault trees, this paper introduced a new decision graph model to visually represent dominance-based if–then decision rules. The fundamental principles of decision graphs and the algorithms used to generate them were presented. Additionally, decision graph linearisation and exploitation algorithms were described, both based on the well-known Breadth-First Search network traversal algorithm. Real-world data pertaining to European cultural heritage illustrated the practical value of the proposed decision graph model.

Despite these advantages, decision graphs also present several limitations. As the number of decision rules increases, the resulting graph may grow substantially in size, leading to a large number of nodes and edges that can become difficult to visualise and manipulate. Although the mixed column splitting and column ordering transformation strategy proposed in this paper mitigates this issue, the final graph structure may still remain relatively large. It should be noted, however, that these challenges are not specific to decision graphs but are shared by many existing graph models, where structural complexity also tends to increase with the size of the rule base and may even lead to larger or less structured representations. Finally, decision graphs preserve the characteristics of their source rule sets, including any redundancy, inconsistency, or unnecessary complexity.

Several practical and theoretical research directions remain open. From a practical perspective, the Decision Graph model should be evaluated on diverse rule sets to assess its ability to reduce cognitive load and handle large-scale applications. Further work is also needed to improve the optimisation process for transforming rule sets into graphs. Although the current approach combines column splitting and ordering, multi-objective optimisation, metaheuristics, integer programming, and constraint-based methods may further improve the trade-off between compactness and rule fidelity.

Future methodological developments include applying clustering techniques to binary decision matrices to identify structural patterns, investigating alternative traversal strategies such as Breadth-First Search, Depth-First Search, and A* search, and extending the framework to additional rule types, including probabilistic DRSA rules Greco et al. (2001c), ranking and choice DRSA rules Greco et al. (2001a), IRSA-based rules Pawlak (1982, 1991), and association rules Agrawal et al. (1993). Another direction concerns reducing graph size by combining column splitting and ordering strategies. Finally, the framework could be extended to direct rule induction from decision tables through weighted edges and additional evaluation measures inspired by DOMLEM Greco et al. (2001c) and VC-DomLEM.

Acknowledgements The authors would like to express their sincere gratitude to the editor-in-chief and the anonymous reviewers for their detailed and insightful comments, which have greatly helped to improve the quality and clarity of this paper. The authors want also to acknowledge the representative of the municipality and the experts who have interacted with the analysts. The present work is part of the Horizon 2020 EU research project "CLIC Circular models Leveraging Investments in Cultural heritage adaptive reuse." This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 776758. This work reflects only the authors' views: the Agency is not responsible for any use that may be made of the information it contains.

Appendices A to E

See supplementary document.

References

4eMka (2000). *4eMka system - A rule system for multicriteria decision support integrating dominance relation with rough approximation*. Laboratory of Intelligent Decision Support Systems, Institute of Computing Science, Poznan University of Technology Poznan, Poland. URL: https://fcds.cs.put.poznan.pl/IDSS/software/software_and_projects.htm.

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data* SIGMOD '93 (p. 207–216). New York, NY, USA: Association for Computing Machinery. doi:https://doi.org/10.1145/170035.170072.

Aguilera, J., Farías, D. I. H., Montes-y Gómez, M., & González, L. C. (2021). Detecting traces of self-harm in social media: A simple and interpretable approach. In I. Batyrshin, A. Gelbukh, & G. Sidorov (Eds.), *Advances in Soft Computing* (pp. 196–207). Cham: Springer International Publishing. doi:https://doi.org/10.1007/978-3-030-89820-5_16.

Akers, S. (1978). Binary decision diagrams. *IEEE Transactions on Computers*, *C-27*, 509–16. doi:https://doi.org/10.1109/TC.1978.1675141.

Atteya, T., Chakhar, S., Labib, A., Cox, A., & Ishizaka, A. (2024). Estimating relative importance of criteria by post-processing dominance-based rough set approach's outputs. *European Journal of Operational Research*, *315*, 1096–122. doi:https://doi.org/10.1016/j.ejor.2023.12.027.

Banks, A. P., Gamblin, D. M., & Hutchinson, H. (2020). Training fast and frugal heuristics in military decision making. *Applied Cognitive Psychology*, *34*, 699–709. doi:https://doi.org/10.1002/acp.3658.

Bhuyan, B. P., Ramdane-Cherif, A., Tomar, R., & Singh, T. P. (2024). Neuro-symbolic artificial intelligence: A survey. *Neural Computing and Applications*, *36*, 12809–44. doi:10.1007/s00521-024-09960-z.

Błaszczyński, J., Greco, S., & Słowiński, R. (2007). Multi-criteria classification – a new scheme for application of dominance-based decision rules. *European Journal of Operational Research*, *181*, 1030–44. doi:https://doi.org/10.1016/j.ejor.2006.03.004.

Blaszczyński, J., Greco, S., & Slowiński, R. (2012a). Inductive discovery of laws using monotonic rules. *Engineering Applications of Artificial Intelligence*, *25*, 284–94. doi:https://doi.org/10.1016/j.engappai.2011.09.003.

Błaszczyński, J., Słowiński, R., & Szeląg, M. (2011). Sequential covering rule induction algorithm for variable consistency rough set approaches. *Information Sciences*, *181*, 987–1002. doi:https://doi.org/10.1016/j.ins.2010.10.030.

Blaszczyński, J., Slowiński, R., & Szeląg, M. (2012b). Induction of ordinal classification rules from incomplete data. In J. Yao, Y. Yang, R. Slowiński, S. Greco, H. Li, S. Mitra, & L. Polkowski (Eds.), *Rough Sets and Current Trends in Computing* (pp. 56–65). Springer Berlin Heidelberg volume 7413 of *Lecture Notes in Computer Science*. doi:https://doi.org/10.1007/978-3-642-32115-3_6.

Bozga, M., & Maler, O. (1999). On the representation of probabilities over structured domains. In N. Halbwachs, & D. Peled (Eds.), *Computer Aided Verification* (pp. 261–73). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:https://doi.org/10.1007/3-540-48683-6_24.

Brînzei, N., & Aubry, J.-F. (2018). Graphs models and algorithms for reliability assessment of coherent and non-coherent systems. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, *232*, 201–15. doi:https://doi.org/10.1177/1748006X17744381.

Bryant, R. E. (2018). Binary decision diagrams. In E. M. Clarke, T. A. Henzinger, H. Veith, & R. Bloem (Eds.), *Handbook of Model Checking* (pp. 191–217). Cham: Springer International Publishing. doi:https://doi.org/10.1007/978-3-319-10575-8_7.

Byeon, S., & Lee, W. (2023). Directed acyclic graphs for clinical research: A tutorial. *Journal of Minimally Invasive Surgery*, *26*, 97–107. doi:https://doi.org/10.7602/jmis.2023.26.3.97.

Chakhar, S., Crestani, F., & Ishizaka, A. (2026). Preference learning based approach to assess depressive symptoms and depression intensity in online forums. *European Journal of Operational Research*, *335*, 607–28. doi:https://doi.org/10.1016/j.ejor.2026.04.011.

Chakhar, S., Ishizaka, A., Labib, A., & Saad, I. (2016). Dominance-based rough set approach for group decisions. *European Journal of Operational Research*, *251*, 206–24. doi:https://doi.org/10.1016/j.ejor.2015.10.060.

Chakhar, S., Ishizaka, A., Thorpe, A., Cox, J., Nguyen, T., & Ford, L. (2020). Calculating the relative importance of condition attributes based on the characteristics of decision rules and attribute reducts: Application to crowdfunding. *European Journal of Operational Research*, *286*, 689–712. doi:https://doi.org/10.1016/j.ejor.2020.03.039.

Chakhar, S., & Saad, I. (2012). Dominance-based rough set approach for groups in multicriteria classification problems. *Decision Support Systems*, *54*, 372–80. doi:https://doi.org/10.1016/j.dss.2012.05.050.

Demšar, J., Curk, T., Erjavec, A., Gorup, Č., Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., Štajdohar, M., Umek, L., Žagar, L., Žbontar, J., Žitnik, M., & Zupan, B. (2013). Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, *14*, 2349–53. URL: http://jmlr.org/papers/v14/demsar13a.html.

Deng, W., Hu., F., Blaszczyński, J., Slowiński, R., Szeląg, M., & Wang, G. (2013). A novel method for elimination of inconsistencies in ordinal classification with monotonicity constraints. *Fundamenta Informaticae*, *126*, 377–95. doi:https://doi.org/10.5555/2637666.2637672.

Díaz-Ramírez, J., Estrada-García, J., & Figueroa-Sayago, J. (2023). Predicting transport mode choice preferences in a university district with decision tree-based models. *City and Environment Interactions*, *20*, 100118. doi:https://doi.org/10.1016/j.cacint.2023.100118.

Distefano, S., & Puliafito, A. (2009). Dependability evaluation with dynamic reliability block diagrams and dynamic fault trees. *IEEE Transactions on Dependable and Secure Computing*, *6*, 4–17. doi:https://doi.org/10.1109/TDSC.2007.70242.

Egaji, A., Chakhar, S., & Brown, D. (2019). An innovative decision rule approach to tyre pressure monitoring. *Expert Systems with Applications*, *124*, 252–70. doi:https://doi.org/10.1016/j.eswa.2019.01.051.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, *29*, 131–63. doi:https://doi.org/10.1023/A:1007465528199.

Greco, S., Matarazzo, B., & Słowiński, R. (1999). The use of rough sets and fuzzy sets in mcdm. In T. Gal, T. J. Stewart, & T. Hanne (Eds.), *Multicriteria Decision Making: Advances in MCDM Models, Algorithms, Theory, and Applications* (pp. 397–455). Boston, MA: Springer US. doi:https://doi.org/10.1007/978-1-4615-5025-9_14.

Greco, S., Matarazzo, B., & Słowiński, R. (2001a). Rough sets theory for multicriteria decision analysis. *European Journal of Operational Research*, *129*, 1–47. doi:https://doi.org/10.1016/S0377-2217(00)00167-3.

Greco, S., Matarazzo, B., & Słowiński, R. (2002). Rough approximation by dominance relations. *International Journal of Intelligent Systems*, *17*, 153–71. doi:https://doi.org/10.1002/int.10014.

Greco, S., Matarazzo, B., & Słowiński, R. (2016). Decision rule approach. In S. Greco, M. Ehrgott, & J. R. Figueira (Eds.), *Multiple Criteria Decision Analysis: State of the Art Surveys* (pp. 497–552). New York, NY: Springer New York. doi:https://doi.org/10.1007/978-1-4939-3094-4_13.

Greco, S., Matarazzo, B., Słowiński, R., & Stefanowski, J. (2001b). An algorithm for induction of decision rules consistent with the dominance principle. In W. Ziarko, & Y. Yao (Eds.), *Rough sets and current trends in computing* (pp. 304–13). Springer Berlin Heidelberg volume 2005 of *Lecture Notes in Computer Science*. doi:https://doi.org/10.1007/3-540-45554-X_37.

Greco, S., Matarazzo, B., Słowiński, R., & Stefanowski, J. (2001c). Variable consistency model of dominance-based rough sets approach. In W. Ziarko, & Y. Yao (Eds.), *Rough sets and current trends in computing* (pp. 170–81). Springer Berlin / Heidelberg volume 2005 of *Lecture Notes in Computer Science*. doi:https://doi.org/10.1007/3-540-45554-X_20.

Greco, S., Pawlak, Z., & Słowiński, R. (2004). Can bayesian confirmation measures be useful for rough set decision rules? *Engineering Applications of Artificial Intelligence*, *17*, 345–61. URL: https://www.sciencedirect.com/science/article/pii/S0952197604000478. doi:https://doi.org/10.1016/j.engappai.2004.04.008. Selected Problems of Knowledge Representation.

Greco, S., Słowiński, R., & Szczęch, I. (2012). Properties of rule interestingness measures and alternative approaches to normalization of measures. *Information Sciences*, *216*, 1–16. doi:https://doi.org/10.1016/j.ins.2012.05.018.

Hafenbrädl, S., Waeger, D., Marewski, J., & Gigerenzer, G. (2016). Applied decision making with fast-and-frugal heuristics. *Journal of Applied Research in Memory and Cognition*, *5*, 215–31. doi:https://doi.org/10.1016/j.jarmac.2016.04.011.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. *SIGKDD Explor. Newsl.*, *11*, 10–8. doi:https://doi.org/10.1145/1656274.1656278.

Heckerman, D. (1998). A tutorial on learning with bayesian networks. In M. I. Jordan (Ed.), *Learning in Graphical Models* (pp. 301–54). Dordrecht: Springer Netherlands. doi:https://doi.org/10.1007/978-94-011-5014-9_11.

Hofmann, M., & Klinkenberg, R. (2014). *RapidMiner: Data Mining Use Cases and Business Analytics Applications*. Chapman and Hall/CRC. doi:https://doi.org/10.1201/b16023.

Hu, Q., Chakhar, S., Siraj, S., & Labib, A. (2017). Spare parts classification in industrial manufacturing using the dominance-based rough set approach. *European Journal of Operational Research*, *262*, 1136–63. doi:https://doi.org/10.1016/j.ejor.2017.04.040.

Ishizaka, A., & Labib, A. (2014). A hybrid and integrated approach to evaluate and prevent disasters. *Journal of the Operational Research Society*, *65*, 1475–89. doi:https://doi.org/10.1057/jors.2013.59.

Jaeger, M. (2004). Probabilistic decision graphs-combining verification and ai techniques for probabilistic inference. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *12*, 19–42. doi:https://doi.org/10.1142/S0218488504002564.

Jenny, M., Pachur, T., Lloyd Williams, S., Becker, E., & Margraf, J. (2013). Simple rules for detecting depression. *Journal of Applied Research in Memory and Cognition*, *2*, 149–57. doi:https://doi.org/10.1016/j.jarmac.2013.06.001.

Jensen, R. (2004). Clab: A c++ library for fast backtrack-free interactive product configuration. In M. Wallace (Ed.), *Principles and Practice of Constraint Programming - CP 2004* (pp. 816–). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:https://doi.org/10.1007/978-3-540-30201-8_94.

Jimenez-Roa, L., Heskes, T., & Stoelinga, M. (2021). Fault trees, decision trees, and binary decision diagrams: A systematic comparison. In B. Castanier, M. Cepin, D. Bigaud, & C. Bérenguer (Eds.), *Proceedings of the 31st European Safety and Reliability Conference (ESREL 2021)* (pp. 673–80). Research Publishing. doi:https://doi.org/10.3850/978-981-18-2016-8_241-cd.

Ke, Y., Wang, X., Ye, Z., Zhang, S., & Cai, Z. (2024). Binary decision diagram-based reliability modeling of phased-mission manufacturing system processing multi-type products. *Quality Technology & Quantitative Management*, *21*, 1058–75. doi:https://doi.org/10.1080/16843703.2023.2286410.

Keller, N., & Katsikopoulos, K. (2016). On the role of psychological heuristics in operational research; and a demonstration in military stability operations. *European Journal of Operational Research*, *249*, 1063–73. doi:https://doi.org/10.1016/j.ejor.2015.07.023.

Labib, A. (2014). *Learning from failures: decision analysis of major disasters*. (1st ed.). United State: Elsevier Inc. doi:https://doi.org/10.1016/C2013-0-00156-9.

Labib, A., Chakhar, S., Hope, L., Shimell, J., & Malinowski, M. (2022). Analysis of noise and bias errors in intelligence information systems. *Journal of the Association for Information Science and Technology*, *73*, 1755–75. doi:https://doi.org/10.1002/asi.24707.

Lakkaraju, H., Bach, S. H., & Leskovec, J. (2016). Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* KDD '16 (pp. 1675–84). New York, NY, USA: Association for Computing Machinery. doi:https://doi.org/10.1145/2939672.2939874.

Legay, C., Cloutier, G., Chakhar, S., Joerin, F., & Rodriguez, M. (2015). Estimation of urban water supply issues at the local scale: a participatory approach. *Climatic Change*, *130*, 491–503. doi:https://doi.org/10.1007/s10584-015-1366-6.

Lipmaa, H. (2010). First cpir protocol with data-dependent computation. In D. Lee, & S. Hong (Eds.), *Information, Security and Cryptology - ICISC 2009* (pp. 193–210). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:doi.org/10.1007/978-3-642-14423-3_14.

Martignon, L., Erickson, T., & Viale, R. (2022). Transparent, simple and robust fast-and-frugal trees and their construction. *Frontiers in Human Dynamics*, *4*. doi:https://doi.org/10.3389/fhumd.2022.790033.

Martignon, L., Vitouch, O., Takezawa, M., & Forster, M. R. (2003). Naive and yet enlightened: From natural frequencies to fast and frugal decision trees. In *Thinking: Psychological Perspectives on Reasoning, Judgment and Decision Making* chapter 10. (pp. 189–211). John Wiley & Sons, Ltd. doi:https://doi.org/10.1002/047001332X.ch10.

Meena, K. K., Bairwa, D., & Agarwal, A. (2024). A machine learning approach for unraveling the influence of air quality awareness on travel behavior. *Decision Analytics Journal*, *11*. doi:https://doi.org/10.1016/j.dajour.2024.100459. In press.

Neth, H., Meder, B., Kothiyal, A., & Gigerenzer, G. (2014). Homo heuristicus in the financial world: From risk management to managing uncertainty. *Journal of Risk Management in Financial Institutions*, *7*, 134–44. doi:https://doi.org/10.69554/BLMQ2227.

Nielsen, J., Gámez, J., & Salmerón, A. (2012). Modelling and inference with conditional gaussian probabilistic decision graphs. *International Journal of Approximate Reasoning*, *53*, 929–45. doi:https://doi.org/10.1016/j.ijar.2011.09.005.

Pananookooln, C., Akaranee, J., & Silpasuwanchai, C. (2023). Comparing selective masking methods for depression detection in social media. *Computational Linguistics*, *49*, 525–53. doi:https://doi.org/10.1162/coli_a_00479.

Pawlak, Z. (1982). Rough sets. *International Journal of Information & Computer Sciences*, *11*, 341–56. doi:https://doi.org/10.1007/BF01001956.

Pawlak, Z. (1991). *Rough set. Theoretical aspects of reasoning about data*. Dordrecht: Kluwer Academic Publishers. doi:https://doi.org/10.1007/978-94-011-3534-4.

Pearl, J. (2013). *Causality: Models, Reasoning, and Inference*. (2nd ed.). Cambridge, UK: Cambridge University Press. doi:https://doi.org/10.1017/CBO9780511803161.

Phillips, N., Neth, H., Woike, J., & Gaissmaier, W. (2017). FFTrees: A toolbox to create, visualize, and evaluate fast-and-frugal decision trees. *Judgment and Decision Making*, *12*, 344–368. doi:https://doi.org/10.1017/S1930297500006239.

Polkowski, L., & Artiemjew, P. (2015). Learning data classification: Classifiers in general and in decision systems. In *Granular computing in decision approximation: An application of rough mereology* (pp. 33–62). Cham, Switzerland: Springer International Publishing volume 77 of *Intelligent Systems Reference Library*. doi:https://doi.org/10.1007/978-3-319-12880-1_3.

Prescott, D., Remenyte-Prescott, R., Reed, S., Andrews, J., & Downes, C. (2009). A reliability analysis method using binary decision diagrams in phased mission planning. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, *223*. doi:https://doi.org/10.1243/1748006XJRR202.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, *1*, 81–106. doi:https://doi.org/10.1007/BF00116251.

Quinlan, J. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies*, *27*, 221–34. doi:https://doi.org/10.1016/S0020-7373(87)80053-6.

Rokach, L., & Maimon, O. (2014). *Data Mining with Decision Trees*. (2nd ed.). WORLD SCIENTIFIC. doi:https://doi.org/10.1142/9097.

Ruijters, E., & Stoelinga, M. (2015). Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review*, *15-16*, 29–62. doi:https://doi.org/10.1016/j.cosrev.2015.03.001.

Saad, I., & Chakhar, S. (2009). A decision support for identifying crucial knowledge requiring capitalizing operation. *European Journal of Operational Research*, *195*, 889–904. doi:https://doi.org/10.1016/j.ejor.2007.11.021.

Salazar-Serna, K., Barona, S. A., García, I. C., Cadavid, L., & Franco, C. J. (2025). Addressing overfitting in classification models for transport mode choice prediction: a practical application in the aburrá valley, colombia. *Transportation Letters*, *17*, 1213–30. doi:https://doi.org/10.1080/19427867.2024.2422717.

Sekulić, I., & Strube, M. (2019). Adapting deep learning methods for mental health prediction on social media. In W. Xu, A. Ritter, T. Baldwin, & A. Rahimi (Eds.), *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)* (pp. 322–7). Hong Kong, China: Association for Computational Linguistics. doi:https://doi.org/10.18653/v1/D19-5542.

Signoret, J.-P., & Leroy, A. (2021). Reliability block diagrams (RBDs). In *Reliability Assessment of Safety and Production Systems: Analysis, Modelling, Calculations and Case Studies* (pp. 195–208). Cham: Springer International Publishing. doi:https://doi.org/10.1007/978-3-030-64708-7_15.

Słowiński, R., Greco, S., & Matarazzo, B. (2002). Axiomatization of utility, outranking and decision-rule preference models for multiple-criteria classification problems under partial inconsistency with the dominance principle. *Control and Cybernetics*, *31*, 1005–35.

Slowiński, R., Greco, S., & Matarazzo, B. (2002). Rough set analysis of preference-ordered data. In J. Alpigini, J. Peters, A. Skowron, & N. Zhong (Eds.), *Rough Sets and Current Trends in Computing* (pp. 44–59). Springer Berlin Heidelberg volume 2475 of *Lecture Notes in Artificial Intelligence*. doi:https://doi.org/10.1007/3-540-45813-1_6.

Slowiński, R., Greco, S., & Matarazzo, B. (2009). Rough sets in decision making. In R. A. Meyers (Ed.), *Encyclopedia of Complexity and Systems Science* (pp. 7753–87). New York, NY: Springer New York. doi:https://doi.org/10.1007/978-0-387-30440-3_460.

Slowiński, R., Stefanowski, J., Greco, S., & Matarazzo, B. (2000). Rough sets based processing of inconsistent information in decision analysis. *Control and Cybernetics*, *29*, 379–404.

Spirtes, P., Glymour, C., & Scheines, R. (2001). *Causation, Prediction, and Search*. Adaptive Computation and Machine Learning (2nd ed.). Cambridge, MA: The MIT Press. doi:https://doi.org/10.7551/mitpress/1754.001.0001.

Stefanowski, J., & Vanderpooten, D. (2001). Induction of decision rules in classification and discovery-oriented perspectives. *International Journal of Intelligent Systems*, *16*, 13–27. doi:https://doi.org/10.1002/1098-111X(200101)16:1<13::AID-INT3>3.0.CO;2-M.

Waghen, K., & Ouali, M.-S. (2019). Interpretable logic tree analysis: A data-driven fault tree methodology for causality analysis. *Expert Systems with Applications*, *136*, 376–91. doi:https://doi.org/10.1016/j.eswa.2019.06.042.

Whaley, J., Avots, D., Carbin, M., & Lam, M. S. (2005). Using datalog with binary decision diagrams for program analysis. In K. Yi (Ed.), *Programming Languages and Systems* (pp. 97–118). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:https://doi.org/10.1007/11575467_8.

Wu, X., & Zhu, T. (2025). From data to diagnosis: A logical learning method to enhance interpretability in bipolar and major depressive disorder identification. *European Journal of Operational Research*, *325*, 362–80. doi:https://doi.org/10.1016/j.ejor.2025.03.016.

Xu, N., Lovreglio, R., Kuligowski, E., Cova, T., Nilsson, D., & Zhao, X. (2023). Predicting and assessing wildfire evacuation decision-making using machine learning: Findings from the 2019 kincade fire. *Fire Technology*, *59*, 793–825. doi:https://doi.org/10.1007/s10694-023-01363-1.

Younsi, F.-Z., Chakhar, S., Ishizaka, A., Hamdadou, D., & Boussaid, O. (2020). A dominance-based rough set approach for an enhanced assessment of seasonal influenza risk. *Risk Analysis: An International Journal*, *40*, 1323–41. doi:https://doi.org/10.1111/risa.13478.