University of Granada
Department of Computer Science and
Artificial Intelligence

PhD Programme
Information and Communication Technologies

PhD Thesis Dissertation

New Methods and Data Structures for Evaluating Influence Diagrams

PhD Student
Rafael Cabañas de Paz

Advisors
Andrés Cano Utrera
Manuel Gómez Olmedo

Granada, March 2017

Editor: Universidad de Granada. Tesis Doctorales
Autor: Rafael Cabañas de Paz
ISBN: 978-84-9163-521-5
URI: http://hdl.handle.net/10481/48313

El doctorando / *The doctoral candidate* **Rafael Cabañas de Paz** y los directores de la tesis / *and the thesis supervisors*: **Andrés Cano Utrera, Manuel Gómez Olmedo**.

Garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

/

Guarantee, by signing this doctoral thesis, that the work has been done by the doctoral candidate under the direction of the thesis advisor/s and, as far as our knowledge reaches, in the performance of the work, the rights of other authors to be cited (when their results or publications have been used) have been respected.

Lugar y fecha / *Place and date*:

Granada, 15/03/2017

Directores de la tesis / *Thesis supervisors*

Doctorando / *Doctoral candidate*

Firma / *Signed*

Firma / *Signed*

Agradecimientos

Me gustaría comenzar dándole las gracias a mis directores de tesis Andrés Cano Utrera y Manuel Gómez Olmedo, cuyos consejos y ayuda han sido indispensables para poder acabar con éxito esta etapa. Durante los años de doctorado, he tenido la posibilidad de visitar centros de investigación en el extranjero, primero el Departamento de Ciencias de la Computación de la Universidad de Aalborg, bajo la tutela Anders L. Madsen, y posteriormente en el centro IDSIA de Lugano bajo la supervisión de Alessandro Antonucci. Quisiera agradecerles a ambos el trato recibido y todo lo que aprendí trabajando con ellos.

Deseo extender mi agradecimiento a los miembros del Departamento de Ciencias de la computación e Inteligencia Artificial de la Universidad de Granada, y en especial al resto de miembros del grupo UTAI. Un agradecimiento muy especial a mis compañeros y amigos del CITIC. Con ellos he compartido muchas horas de trabajo en un excelente ambiente.

También me gustaría agradecer a los miembros del grupo de investigación SIMD de la Universidad de Castilla-La Mancha ya que con ellos dí mis PGMs. Un agradecimiento especial a los miembros del Departamento de Matemáticas de la Universidad de Almería y a los miembros del proyecto AMIDST, con quienes he tenido la oportunidad de trabajar durante los meses previos a la defensa de la tesis.

En último lugar, y no por ello menos importante, también debo agradecer el apoyo de mi familia y mis amigos. En especial, a todas las nuevas amistades que han hecho que Granada sea mi segunda casa.

¡MUCHAS GRACIAS!

Esta tesis doctoral ha sido financiada por el Ministerio de Economía y Competitividad y por el Fondo Europeo de Desarrollo Regional (FEDER) con los proyectos TIN2010-20900-C04-01, TIN2013-46638-C3-2-P y TIN2016-77902-C3-2-P, y por la beca FPI BES-2011-050604.

Acknowledgements

First of all, I would like to express my gratitude to my Andrés Cano Utrera and Manuel Gómez Olmedo, whose advise and help have been essential to successfully complete this stage. As a PhD student, I had the opportunity to visit some foreign research centres, first the Department of Computer Sciences of Aalborg University, under the supervision of Anders L. Madsen, and afterwards the Swiss AI Lab IDSIA in Lugano under the protection of Alessandro Antonucci. I would like to thank both of them for treating me so well and for how much I learnt working with them.

I want to extend my gratitude to all the members of the Department of Computer Science and Artificial Intelligence of the University of Granada, and in particular to the rest of the member of the UTAI group. Special thanks to my colleagues and friends from CITIC. With them, I have spent many working hours in an excellent environment.

I also would like to thank to the members of the research group SIMD of the University of Castilla-La Mancha, with whom I started working in the field of PGMs. A very special thanks goes out to the members Department of Mathematics of the University of Almería and to the members of the AMIDST project. I had the opportunity to work with them during the months preceding to thesis presentation.

Last but not least, I also must thank the support of my family and friends. In particular, I express my gratitude to those who have made Granada my second home.

THANKS YOU VERY MUCH!

This doctoral thesis has been jointly supported by the Spanish Ministry of Economy and Competitiveness and by the European Regional Development Fund (FEDER) under the projects TIN2010-20900-C04-01, TIN2013-46638-C3-2-P and TIN2016-77902-C3-2-P, and by the FPI scholarship BES-2011-050604.

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter 1

Introducción (in Spanish)

Los *modelos gráficos probabísticos* (PGMs) son una potente herramienta de modelado para el aprendizaje y razonamiento en dominios con incertidumbre. Constan de una componente cualitativa y otra cuantitativa. En primer lugar, la componente cualitativa viene dada por un grafo que representa un conjunto de dependencias entre las variables (nodos) del dominio modelado. En segundo lugar, la componente cuantitativa consiste en un conjunto de funciones que cuantifican dichas dependencias.

Esta tesis se centra en los *diagramas de influencia (IDs)* [87, 56], que son un tipo específico de PGM usado para el modelado y resolución de problemas de decisión bajo incertidumbre. En este tipo de problemas, el decisor debe elegir entre varias acciones, cada una de las cuales conlleva múltiples consecuencias posibles. Cada consecuencia tiene asociada una utilidad (por ejemplo, beneficio económico) y el decisor preferirá aquella con mayor utilidad. Sin embargo, no siempre se trata de una elección directa, ya que la consecuencia de cada acción puede que no se conozca con certeza.

La evaluación de IDs facilita el cálculo de la utilidad esperada asociada a cada acción y por lo tanto, permite la identificación de las mejores acciones para el decisor. En general, un ID es un PGM cuya componente cualitativa es un grafo acíclico dirigido con tres tipos de nodos: decisión, que corresponde con las acciones que el decisor puede controlar; azar, que representan la incertidumbre del problema; y utilidad, representando las preferencias del usuario. Por otro lado, la componente cuantitativa está formada por un conjunto de distribuciones de probabilidad condicionada (asociadas a los nodos de azar) y de funciones de utilidad (asociadas a los nodos de utilidad).

Otra ventaja de los IDs reside en su naturaleza intuitiva. Esto hace que puedan ser fácilmente comprendidos por expertos de otros campos que no están familiarizados con los métodos de aprendizaje automático. Por esta razón, los IDs se han aplicado en múltiples ámbitos: medicina [76, 3, 88, 77], economía [34], ciencias ambientales [81, 29], defensa [46, 82], etc.

A pesar de todas las ventajas previamente mencionadas, el formalismo de los IDs tiene algunos inconvenientes:

- **Elevado coste computacional:** los potenciales intermedios generados durante la evaluación pueden ser extremadamente grandes. Como consecuencia, la evaluación de IDs tiene un elevado coste computacional en términos de memoria y tiempo. Esta situación es incluso más problemática en diagramas complejos, cuya evaluación puede no ser factible debido a las limitaciones de los ordenadores.

- **Evaluación ineficiente de problemas de decisión asimétricos:** en muchos problemas de decisión reales, los valores que una variable puede tomar dependen del pasado. Por ejemplo, consideremos un problema de decisión en que existe la posibilidad de realizar un test. Si se opta por no realizarlo, entonces cualquier resultado de dicho test no debe ser considerado en la evaluación. Este tipo de problemas recibe el nombre de asimétricos [108, 4] y un inconveniente importante de los IDs está relacionado con la incapacidad para representarlos de forma eficiente. Para poder modelar un problema asimétrico con un ID, hay que transformarlos en simétricos. Esto se consigue añadiendo estados artificiales a los dominios de algunas variables. Además, los potenciales pueden contener algunas configuraciones imposibles (debido a las asimetrías) que no son relevantes para resolver el problema. Esto implica que los algoritmos de evaluación convencionales requieran gran cantidad innecesaria de memoria y tiempo de computación.

- **Incapacidad para expresar imprecisión:** en el formalismo clásico de los IDs, los potenciales son funciones que asignan valores precisos a cada combinación de estados para un conjunto de variables (por ejemplo, la probabilidad de $X = x_1$ es 0.75). Esto puede ser un inconveniente al modelar problemas reales, donde los potenciales se suelen obtener a partir de valoraciones de los expertos o de datos parcialmente fiables.

Para solucionar estos problemas, proponemos varias estructuras de datos para representar la información cuantitativa y cualitativa de un ID. Además, proponemos varios algoritmos de evaluación con dichas estructuras.

1.1 Contribuciones

1.1.1 Reducción del coste computacional

Una de las contribuciones principales de esta tesis es el uso de *árboles binarios (BTs)* en lugar de tablas para representar los potenciales asociados a un ID. Con esto, se pretende resolver el problema de la generación de potenciales de gran tamaño y reducir el coste de la evaluación. Un BT es una estructura en forma de árbol cuyos nodos hoja están etiquetados con los valores numéricos del potencial. Cada nodo interno está etiquetado con una variable y siempre tiene dos arcos salientes etiquetados con subconjuntos de los estados de la variable. La ventaja de utilizar BTs reside en su capacidad para representar formas generales de independencia que no pueden ser codificadas por la estructura de los IDs. Esto es posible porque algunos valores idénticos del potencial se pueden agrupar. Con esto se consigue que la representación de potenciales sea más compacta. Puesto que estas formas generales de independencia son habituales en IDs complejos que representan problemas de decisión reales, su evaluación debe ser más eficiente. Además, si un BT es extremadamente grande, se puede podar con lo que se aproxima el potencial (y por lo tanto la evaluación).

Los conceptos clave para la representación de potenciales como BTs se detallan en el Capítulo 5: se dan las definiciones básicas sobre BTs con especial atención a las formas de independencia que pueden ser codificadas usándolos. También proponemos algoritmos heurísticos para construir (a partir de tablas) y podar BTs que representan potenciales de probabilidad y utilidad. La evaluación de IDs usando esta estructura de datos se explica en el Capítulo 8. Para ello, proponemos algoritmos recursivos para realizar las operaciones básicas directamente sobre BTs. También explicamos como algunos de los principales algoritmos de evaluación pueden se adaptados para trabajar con este tipo de representación de potenciales. En el trabajo experimental comparamos los BTs con otras representaciones en varios aspectos: tiempo de computación, memoria y error. El contenido relacionado con los BTs fue publicado en la revista internacional IJUFKS [19] y presentado en el workshop PGM'12 [17] y en CAEPIA'13 [13].

Algunos de los algoritmos adaptados para funcionar con BTs son algoritmos *greedy* que eligen a cada paso qué variable eliminar. Esta decisión normalmente se toma basándose en alguna heurística que trata de minimizar la complejidad de las operaciones involucradas en la evaluación. Sin embargo, estas heurísticas pueden no ser adecuadas si los potenciales están representados mediante BTs. Por esta razón, en el Capítulo 9 se proponen dos nuevas heurísticas que consideran que los potenciales están representados mediante BTs. Estas nuevas heurísticas estiman el tamaño del BT generado al eliminar una variable. Haciendo esto, pretendemos reducir incluso más la complejidad de la evaluación. Este trabajo fue publicado en la conferencia SCAI'13[16].

En el Capítulo 12 también exploramos distintas alternativas para reducir el coste de la evaluación asumiendo que los potenciales están representados usando tablas. Nos basamos en la siguiente idea: algunos de los principales algoritmos de evaluación requieren la realización de varias combinaciones y marginalizaciones con los potenciales asociados al ID. Encontrar un orden óptimo para dichas operaciones es un problema NP-difícil [5] y es elemento de gran importancia para la eficiencia de la evaluación. Por lo tanto, la evaluación de un ID se puede ver como un problema de optimización combinatoria, es decir, en el problema de encontrar un orden óptimo para las operaciones de combinación y marginalización.

En esta tesis, se proponen varias formas para optimizar el orden de las operaciones involucradas en la evaluación de IDs. En primer lugar, adaptamos el algoritmo *symbolic probabilistic inference (SPI)* para evluar IDs. En segundo lugar, se propone una mejora del algoritmo *Variable Elimination (VE)*. Esta mejora consiste en utilizar un algoritmo greedy para minimizar el coste de cada operación requerida. Se puede ver como una extensión la estructura *binary join tree* propuesta por P.P. Shenoy [106] . En el trabajo experimental, analizamos el comportamiento de todos estos algoritmos utilizando varios diagramas presentes en la literatura. Se prueba de forma empírica que los algoritmos propuestos pueden mejorar la eficiencia de la evaluación. Además, SPI supera a VE en muchos IDs. Versiones preliminares de este contenido fueron presentadas en los congresos IPMU'14 [20] y PGM'14 [14] . Finalmente, una versión extendida fue publicada en la revista internacional IJAR [15].

1.1.2 Evaluación eficiente de problemas de decisión asimétricos

El problema relacionado con la evaluación de IDs modelando problemas de decisión asimétricos también se puede solucionar usando BTs. En nuestra propuesta, la información cualitativa sobre el problema (restricciones debidas a las asimetrías) se repreresenta con BTs y se denominan *binary constraint trees (BCTs)*. La estructura de estos BTs es la misma que la de aquellos que representan potenciales, aunque las hojas sólo pueden estar etiquetadas con un 0 o un 1. Una hoja con un 0 identifica una combinación imposible de estados de las variables en el BCT. Puesto que la misma estructura de datos se utiliza para representar asimetrías y potenciales, los BCTs se pueden aplicar para reducir el número de escenarios a considerar. Como resultado, se mejora la eficiencia de la evaluación de IDs representando problemas de decisión asimétricos. Con este método se pretende mantener el marco de los IDs sin cambios, además de utilizar los algoritmos estándar de evaluación.

En el Capítulo 6 se dan las definiciones básicas sobre BCTs. Detallamos cómo los BCTs se pueden construir a partir de reglas de restricción, que son expresiones lógicas destinadas a definir asimetrías de forma más intuitiva. En el Capítulo 10 explicamos cómo evaluar un ID con asimetrías . En particular, proponemos una extensión de VE que funciona con BCTs. En el trabajo experimental, mostramos que la aplicación de restricciones representadas mediante BTs puede reducir la complejidad de la evaluación en términos de memoria y tiempo. Este método fue presentado en la conferencia internacional ECSQARU'13 [18].

1.1.3 Extensión a modelos imprecisos

Otra contribución importante de esta tesis consiste en extender el formalismo de los IDs a intervalos. Haciendo esto, pretendemos resolver el problema de la incapacidad de los IDs para expresar imprecisión o vaguedad en los potenciales (normalmente obtenida de expertos o de datos parcialmente fiables). En particular, proponemos reemplazar los valores precisos en los potenciales por intervalos. Esta generalización se denomina *potenciales con intervalos*. Un ID con este tipo de potenciales se puede ver como una colección de IDs clásicos (precisos) consistentes con las restricciones impuestas por los intervalos. También se proponen extensiones de algunos algoritmos para poder evaluar IDs con intervalos. Los métodos propuestos son aproximaciones externas[1] de las soluciones exactas. Sin embargo, el uso de métodos de programación lineal evita que se produzcan aproximaciones externas de gran tamaño sin aumentar la complejidad computacional. Usando intervalos, la evaluación exacta es imposible: se require entonces de un modelo de imprecisión más general. Sin embargo, esto aumentaría exponencialmente la complejidad de la evaluación.

La formalización del concepto de potencial con intervalos se da en el Capítulo 7. Los algoritmos propuestos para la evaluación de IDs con intervalos se detallan en el Capítlulo 11. También se realiza una comparación experimental contra un método similar para este tipo de modelos, con lo que se muestra una mejora en términos de tiempo y precisión. Esta extensión a intervalos fue inicialmente

[1] Un método es una *aproximación externa* o ”outer approximation´´ cuando la solución obtenida contiene a la exacta.

presentada en la conferencia ECSQARU'15 [11]. Posteriormente, fue publicada en la revista internacional IJAR [12] con todos los métodos aquí propuestos y con descripciones y ejemplos más precisos.

1.2 Conclusiones y líneas futuras

En esta sección resumimos todas las conclusiones que se presentan a lo largo de esta memoria. También se enumeran las publicaciones con la mayoría del trabajo presentado. Finalmente, concluimos con las líneas de trabajo futuro.

Esta tesis está dedicada, tal y como indica su nombre, a nuevas estructuras de datos y métodos para la evaluación de IDs. En particular, con el trabajo presentado se han abordado algunos problemas de los IDs: alto coste computacional, evaluación ineficiente de problemas de decisión asimétricos e incapacidad para expresar imprecisión.

Hemos propuesto el uso de BTs para representar los potenciales en IDs. Con ello, se ha conseguido reducir el coste computacional de la evaluación debido al menor tamaño de los potenciales. Se han propuesto métodos para construir y aproximar BTs. Además, se ha indicado cómo se pueden adaptar algunos de los algoritmos de evaluación tradicionales para evaluar IDs directamente con BTs. En el trabajo experimental, se ha demostrado que, en general, el uso de BTs requiere menos recursos de memoria que otros tipos de representaciones, como son el caso de los NTs o las tablas. Como consecuencia, la evaluación es habitualmente más rápida usando BTs. Sin embargo, para algunos IDs es necesario aproximar los potenciales para obtener algún beneficio del uso de BTs. Otra conclusión es que el uso de BTs permite obtener mejores soluciones aproximadas: el mismo nivel de error es obtenido con un menor tiempo de cómputo.

En relación a uno de los algoritmos adaptados para trabajar con BTs, en particular VE, hemos propuesto dos nuevas heurísticas para determinar el orden de eliminación. La novedad de dichas heurísticas reside en éstas consideran que los potenciales están representados mediante BTs. En el trabajo experimental se ha demostrado que, con estas heurísticas, el coste computacional se puede reducir aún más.

Para abordar el problema del coste computacional, también hemos estudiado distintas alternativas que permiten representar los potenciales con tablas. En particular, hemos propuesto adaptar el algoritmo SPI para evaluar IDs, y una optimización de VE. Estos nuevos métodos tratan de optimizar el orden de las operaciones involucradas. En la evaluación experimental, se ha demostrado que con los algoritmos propuestos se puede mejorar la eficiencia de la evaluación. Al comparar ambas propuestas, hemos visto que SPI puede superar a VE en muchos IDs.

El segundo inconveniente de los IDs es la evaluación ineficiente de IDs modelando problemas de decisión asimétricos. Este problema también puede ser solventado utilizando BTs. En nuestra propuesta, los potenciales se representan mediante BT pero también las asimetrías. Al usar el mismo tipo de representación, las asimetrías se pueden aplicar fácilmente a los potenciales. Con esto se reduce el número de escenarios a considerar durante la evaluación. En esta tesis hemos demostrado de forma empírica que, con nuestra propuesta, normalmente se mejora la eficiencia de la evaluación de IDs que modelan problemas de decisión asimétricos. Sin embargo, para IDs muy pequeños la aplicación de restricciones puede ser contraproducente (debido a la sobrecarga introducida).

Otra propuesta de representación de potenciales en IDs son los denominados potenciales con intervalos. Se ha formalizado esta representación y se han dado varios algoritmos para evaluar IDs con potenciales imprecisos. En el análisis empírico hemos mostrado que los nuevos métodos basados en técnicas de programación lineal son los más precisos para evaluar IDs con intervalos.

1.2.1 List of publications

El trabajo desarrollado en esta tesis ha sido presentado en las siguientes publicaciones.

Publicaciones en revistas internacionales

R. Cabañas, A. Antonucci, A. Cano, and M Gómez-Olmedo. Evaluating interval-valued influence diagrams. *International Journal of Approximate Reasoning*, 80:393–411, 2017

R. Cabañas, A. Cano, M. Gómez-Olmedo, and A. L. Madsen. Improvements to variable elimination and symbolic probabilistic inference for evaluating influence diagrams. *International Journal of Approximate Reasoning*, 70:13–35, 2016

R. Cabañas, M. Gómez-Olmedo, and A. Cano. Using binary trees for the evaluation of influence diagrams. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 24(01):59–89, 2016

Publicaciones en conferencias internacionales y workshops

R. Cabañas, A. Antonucci, A. Cano, and M. Gómez-Olmedo. Variable elimination for interval-valued influence diagrams. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty: 13th European Conference, ECSQARU 2015, Compiègne, France, July 15-17, 2015. Proceedings*, volume 9161 LNAI, pages 541–551. Springer, 2015

R. Cabañas, A. Cano, M. Gómez-Olmedo, and A. L. Madsen. On SPI-lazy evaluation of influence diagrams. In *Probabilistic Graphical Models: 7th European Workshop, PGM 2014, Utrecht, The Netherlands, September 17-19, 2014. Proceedings*, pages 97–112. Springer International Publishing, 2014

R. Cabañas, A. L. Madsen, M. Gómez-Olmedo, and A. Cano. *On SPI for evaluating Influence Diagrams*, pages 506–516. Springer International Publishing, Cham, 2014

R. Cabañas, M. Gómez-Olmedo, and A. Cano. Evaluating asymmetric decision problems with binary constraint trees. In *Symbolic and Quantiative Approaches to Resoning with Uncertainty: 12th European Conference, ECSQARU 2013, Utrecht, The Netherlands, July 8-10, 2013, Proceedings*, volume 7958 LNAI, pages 85–96. Springer, 2013

R. Cabañas, M. Gómez, and A. Cano. Approximate inference in influence diagrams using binary trees. In *Proceedings of the Sixth European Workshop on*

Probabilistic Graphical Models (PGM-12), 2012

Publicaciones en conferencias nacionales

R. Cabañas, A Cano, M Gómez-Olmedo, and A. L. Madsen. Approximate lazy evaluation of influence diagrams. In *Advances in Artificial Intelligence: 15th Conference of the Spanish Association for Artificial Intelligence, CAEPIA 2013, Madrid, September 17-20, 2013, Proceedings*, volume 8109, page 321. Springer, 2013

R. Cabañas, A. Cano, M Gómez-Olmedo, and A.L. Madsen. Heuristics for determining the elimination ordering in the influence diagram evaluation with binary trees. In *Twelfth Scandinavian Conference on Artificial Intelligence: SCAI 2013*, volume 257 of *Frontiers in Artificial Intelligence and Applications*, pages 65–74. IOS Press, 2013

1.2.2 Líneas Futuras

En relación con las líneas futuras de trabajo con BTs, se pueden estudiar formas alternativas de aplicar la operación de poda durante la evaluación. En la propuesta quí presentada, todos los BTs se podan antes de la evaluación. Sin embargo, podríamos considerar aplicar la poda sólo si se estima que una operación con potenciales va a ser muy costosa. Haciendo esto, se podría minimizar el error de la aproximación. También podríamos considerar el uso de estructuras de datos que representan un modelo completo, tales como los RPTs [23, 24, 25].

En lo que se refiere a la evaluación de problemas de decisión asimétricos, se podría estudiar el comportamiento de los BTs con restricciones usando otros algoritmos, por ejemplo *Arc Reversal* [102], *Lazy evaluation* [79], etc.

Tal y como se presenta en esta tesis, el algoritmo SPI sólo considera el siguiente par de potenciales para combinar. Por lo tanto, una posible línea de investigación podría ser el estudio de dicho algoritmo utilizando un grado mayor de vecindad. También podría ser interesante hacer un estudio que nos permitiese determinar qué características de un ID hacen que SPI funcione mejor que VE.

Aunque los potenciales con intervalos son una alternativa eficiente para modelar la imprecisión en los IDs, podríamos extenderlos a un marco más general de imprecisión, como por ejemplo los conjuntos credales representados por puntos extremos. Esto afectaría a la complejidad computacional de la evaluación, por lo que sería necesario el desarrollo de algoritmos aproximados. Una posible solución sería la utilización de BTs para representar potenciales imprecisos.

Chapter 2

Introduction

Probabilistic Graphical Models (PGMs) constitute a powerful modelling framework for learning and reasoning under uncertainty and are defined by two components: first, a qualitative component in the form of a graph that encodes a set of dependencies among the variables (i.e., nodes) in the domain being modelled; secondly, a quantitative component consisting of a set of functions quantifying such dependencies.

This dissertation focuses on discrete *influence diagrams (IDs)* [87, 56], which are a class of PGM intended to model and solve decision problems under uncertainty. In this kind of problems, the decision maker has to choose between a set of decision options. Each of them could give rise to more than one possible outcome. The decision maker will prefer the outcome with the highest utility (e.g., economic profit). However, this is not a straightforward choice as the outcome of each decision option might not be known with certainty. Then, the expected utility of each decision option should be calculated, which is the probability-weighted average utility of the possible outcomes.

The ID formalism simplifies the computation of expected utilities of various decision options given the information known at the time of the decision. Thus, the evaluation (i.e., inference) of an ID allows the identification of the best decision options for the decision maker given previous observations and decisions. In general, an ID is a PGM whose qualitative component is a *directed acyclic graph (DAG)* with three types of nodes: *decision* nodes that correspond with the actions which the user can control; chance nodes representing the uncertainty; and utility nodes representing the decision maker preferences. On the other hand, the quantitative component of an ID is a set of conditional probability distributions and utility functions associated to the chance and utility nodes respectively. In general, we will talk about *probability* and *utility potentials*.

Besides being useful in the computation of expected utilities, IDs also have an intuitive nature. This makes that IDs could easily be understood by experts from other fields who are usually not familiar with machine learning methods. For that reason, IDs are suitable models for many real application domains: medicine [76, 3, 88, 77], economics [34], environmental sciences [81, 29], defence [46, 82], etc.

In spite of all the advantages previously stated, the classical ID formalism has, however, some drawbacks:

- **High computational cost**: the intermediate potentials generated during the evaluation of IDs may be extremely large. As a consequence, the evaluation has a high computational cost in terms of memory space and time. This situation is even more problematic in complex IDs, whose evaluation may become infeasible due to the limitations of the computer.
- **Inefficient evaluation of asymmetric decision problems**: in many real decision problems, the possible outcomes and decision options for some variables may depend on the past. For example, let us consider a decision problem where you have the option of doing a test, then any possible result for this test is meaningless if you have decided not doing it. This kind of poblems are so-called asymmetric [108, 4] and an important drawback of IDs is related to their inability for efficiently representing them. To be modelled as an ID, asymmetric decision problems must be symmetrized by adding artificial states to the domains of some variables. Even more, potentials might contain some impossible configurations (due to the asymmetries of the problem) that are not relevant for solving the decision problem. This

implies that the evaluation with conventional algorithms, though feasible, involves a considerable amount of unnecessary memory space and computation time.

- **Incapacity to express imprecision**: in the classical ID formalism, potentials are functions that assigns sharp (i.e., precise) values to each possible combination of states for a set of variables (e.g., the probability of $X = x_1$ is 0.75) . This might be an issue with real models, whose potentials are typically obtained from expert judgements or partially reliable data: sharp values can be unable to express a qualitative expert judgement or a statistical analysis based on scarce or missing data (e.g., which is the number modelling the probability for an option more probable than its negation? And which is the value modelling that the probability of $X = x_1$ is high?).

For addressing these problems, in this dissertation we propose several data structures for representing the quantitative and the qualitative components of an ID. Additionally, several evaluation methods using these data structures are also proposed.

2.1 Contributions

2.1.1 Computational cost reduction

One main contribution of this thesis is the proposal of *binary trees (BTs)* instead of tables for representing and managing the potentials involved in IDs. In doing so, we aim to address the problem of the large generated potentials and reduce the computational complexity of the evaluation. A BT is a tree based structure where leaf nodes labelled with the numerical values of the potential. Each internal node is labelled with a variable and they always have two outgoing branches labelled with subsets of the states of the variable. The advantage of BTs resides in their capability of representing general forms of independence that cannot be structurally encoded by IDs. This is possible because some identical values in the potential can be grouped into a single one. This enhanced capability makes the representation of potentials even more compact. As this forms of independence are quite frequent in large IDs representing real world decision problems, their evaluation should be more efficient. Additionally, if BTs are still extremely large, they can be pruned leading to faster but approximate solutions.

The key concepts for representing potentials as BTs are detailed in Chapter 5: basics definitions of BTs are given with special focus on the forms of independences that can be encoded using BTs. We also propose heuristic algorithms for building (from tables) and pruning BTs representing probability and utility potentials. The ID evaluation using this data structure is explained in Chapter 8. For that, we propose recursive algorithms for doing the basics operations directly on BTs. Then, we explain how some of the main evaluation algorithms can be adapted for working with this potential representation. In the experimental work, we compare BTs with other representations in different aspects: computation time, storage requirements and error. The content related to BTs was published in the international journal IJUFKS [19] and presented at the workshop PGM'12 [17] and at CAEPIA'13 [13].

Some evaluation methods proposed for working with BTs are greedy algorithms that choose at each step the next variable to remove. This decision is taken based on any heuristic that tries to minimize the complexity of the operations involved in the evaluation. However, these heuristics might not be suitable if potentials are represented using BTs. For that reason, in Chapter 9, two new heuristics that consider that potentials are represented using BT are proposed. These heuristics estimate the size of BTs generated when removing a variable. In doing so, we aim to reduce even more the complexity of the evaluation. This work was published in a conference paper at SCAI'13 [16].

In Chapter 12, we also explore different alternatives for reducing the computational cost of the evaluation assuming that the potentials of the ID are represented as tables. We base on the following idea: some of the main evaluation algorithms require performing several combinations and marginalizations on the potentials attached to the ID. Finding an optimal order for these operations is a NP-hard problem [5] and it is an element of crucial importance for the efficiency of the evaluation. The evaluation of an ID can be considered as a combinatorial optimization, that is the problem of finding an optimal order in which combinations and marginalizations are performed.

In this dissertation, different approaches for optimizing the order of the operations involved in the evaluation of IDs are considered. First, we adapt the *symbolic probabilistic inference (SPI)* algorithm [104, 74] for evaluating IDs. Secondly, an optimization of variable elimination (VE). This optimization consists of using a greedy algorithm for minimizing the cost of each operation with the potentials required for the evaluation. This optimization can be seen as an extension of the *binary join trees* of P.P. Shenoy [106] to IDs. In the experimental work, we analyse the behaviour of all these algorithms using a set of IDs from the literature. It is empirically proved that the proposed algorithms can improve the efficiency of the evaluation. Moreover, SPI outperforms VE in many IDs. Preliminary versions of this content were presented at IPMU'14 [20], at the workshop PGM'14 [14] and finally published at the international journal IJAR [15].

2.1.2 Efficient evaluation of asymmetric decision problems

The problem related to the evaluation of IDs modelling asymmetric decision problems can also be solved using BTs. In our approach, the qualitative information about the problem (constraints, due to asymmetries) is represented using BTs and will be called *binary constraint trees (BCTs)*. The structure of these BTs is the same that the one for representing potentials, yet the leaves can only be labelled either with a 0 or a 1. A leaf with a 0 identifies an impossible combination of the states of the variables in the BCT. As the same data structure is used for both, potentials and asymmetries, they can be easily applied in order to reduce the number of scenarios to consider. As a consequence, the efficiency of the evaluation of IDs representing asymmetric decision problems is improved. With this approach, we try to keep the framework without changes, as well as using the standard algorithms as much as possible.

In Chapter 6, the basic definition about BCTs are given. We detail how BCTs can be build from constraint rules, which are logical expressions for defining asymmetries in a more intuitive way. In Chapter 10 we explain how an ID with asymmetries can be evaluated. In particular, an extension of the *variable elimination* algorithm for working with BCTs is proposed. In the experimental work, we show that application of constraints represented as BTs can reduce the complexity of the evaluation in terms of time and memory. This approach was presented at the international conference ECSQARU'13 [18].

2.1.3 Extension to imprecise models

Another main contribution of this dissertation consists of extending the ID formalism to the interval-valued case. In doing so, we address the problem of incapacity of standard IDs to express the imprecision or vagueness in their potentials (usually obtained from experts or from partially reliable data). In particular, we propose replacing the sharp values of potentials involved in an ID with intervals. Such generalization is called *interval-valued potentials*. An ID with this kind of potentials is an *interval-valued ID* and it can be seen as a collection of classical (i.e., precise) IDs consistent with the interval constraints. Some standard approaches to IDs evaluation are generalized in order to cope with the interval-valued case. The proposed algorithms are outer approximations[1] of the exact solutions. Yet, the use of linear programming methods avoid to produce unnecessarily large outer approximations without increasing the computational complexity: this remains the same as with precise potentials for both algorithms. Note that using intervals, the exact evaluation is infeasible: a more general imprecise framework would be required. Yet, this would increase exponentially the complexity of the evaluation.

The formalization of the concept of interval-valued potential is given in Chapter 7. Then, the proposed algorithms for evaluation IDs with intervals are detailed

[1] A method is an *outer approximation* when the obtained solution encompasses the exact one.

in Chapter 11. An experimental comparison against a similar approach for these models is also given, and it shows a clear improvement in terms of both evaluation time and accuracy. This extension of the ID formalism to intervals was initially presented at the conference ECSQARU'15 [11]. Afterwards, it was published in the international journal IJAR [12] with all the methods here proposed and with more accurate descriptions and examples.

2.2 Overview

This dissertation is divided across four parts. Part I is an introductory section composed of four chapters. Chapter 1 resumes the main conclusions achieved with this dissertation, and it has been written in Spanish to fulfil the requirements given by the University of Granada related to the Doctoral theses that aim to obtain the International mention. Chapter 2 provides an introduction to the topic of this thesis and explains the main contributions. The required background about the topic is given in Chapter 3. The notation and basic concepts about IDs is introduced in Chapter 4.

The second part, Part II, introduces the new data structures proposed in this dissertation for representing the quantitative and qualitative information in IDs. Chapters 5 and 6 focus on the use of BTs for representing, respectively, potentials and constraints. Chapter 7 provides the formalization of the concept of interval-valued potential as proposed in this dissertation. Part III focuses on the evaluation of IDs with different data structures: Chapters 8 and 9 details the evaluation with BTs. Chapter 10 explains how an ID with asymmetries can be evaluated using BTs for representing potentials and constraints. In Chapter 11, extensions of some evaluation algorithms to cope with intervals are given. Additionally, efficient algorithms for evaluating IDs where potentials are represented as tables are given in Chapter 12.

Finally, Part IV contains one chapter, Chapter 13. This chapter provides a discussion of the main conclusions of the thesis and states the lines for future work. Also, a list of publications supporting the contributions of this dissertation is provided.

Chapter 3

Fundamentals

3.1 Reasoning under uncertainty

Randomness and uncertain judgements are inherent in most of the real-world problems. While humans have the ability to assimilate and reason with uncertain information (i.e. incomplete, contradictory, or subject to change), this is not so straightforward for computers. *Reasoning under uncertainty* is the branch of artificial intelligence that is concerned with modelling this facet of human skill. In other words, it deals with the problematic of creating automatic systems that take the available information and derive conclusions (i.e. infer some new knowledge). In order to illustrate the process of reasoning under uncertainty, let us consider the following situation.

Example 1 (car start problem [64]) *In the morning, my car will not start. I can hear the starter turn, but nothing happens. There may be several reasons for my problem. I can hear the starter roll, so there must be power in the battery. Therefore, the most probable causes are that there is not fuel or that the spark plugs are dirty. To find out, I first look at the fuel meter. It shows full, so I decide to clean the spark plugs.*

In previous example, a human can easily deduce that the problem is that the spark plus are dirty based on the available information (i.e. there must be power in the battery and there is fuel). For making a computer simulate that kind of reasoning, we need ways of representing the problem and ways of performing inference in this representation.

Several paradigms for reasoning under uncertainty have been suggested, such as fuzzy ruled based systems [112], neural networks [33], etc. In this dissertation we will focus on *probabilistic graphical models* (PGMs) which offer an intuitive representation and are based on a well-established theoretical foundation of graph and probability theory. In Section 3.4 we will define the basic concepts about PGMs. Yet, it is required to introduce first the two theories previously mentioned.

3.2 Graph theory

3.2.1 Basics

Graph theory is the field that studies the mathematical structures which represent the relation between a set of objects (e.g. variables in a problem). These structures are called graphs, which can be defined as follows.

Definition 1 (graph) *A graph is a pair* $\mathcal{G} := \langle \mathbf{X}, \mathbf{L} \rangle$ *where* $\mathbf{X}$ *is a set of nodes* $\mathbf{X} := \{X_1, ..., X_n\}$ *and* L *is a set of links or arcs* $\mathbf{L} := \{l_1, ..., l_l\}$*. A graph gives an abstract representation of a set of objects represented by the nodes in* $\mathbf{X}$ *together with binary relations between distinct nodes, represented by the set of links* $\mathbf{L} \subseteq \{(X_i, X_j) \in \mathbf{X} \times \mathbf{X}\}$.

Each link connects a different pair of nodes from $\mathbf{X}$, and the specification of the link will define its nature. A link that connects two nodes X_i, X_j, can be directed $X_i \rightarrow X_j$ or $X_i \leftarrow X_j$ or undirected $X_i - X_j$. Depending on the nature of the links, we can consider the following types of graphs (Figure 3.1 shows an example of each type):

- *Directed* if all its links are directed.
- *Undirected* if all its links are undirected
- *Partially directed* containing both, directed and undirected links.

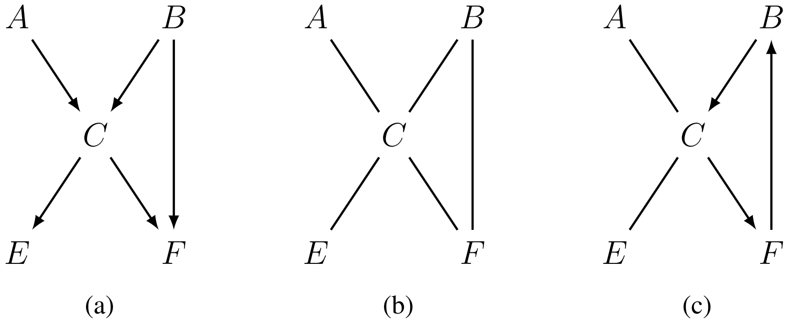


Figure 3.1: Directed graph (a), undirected graph (b) and partially directed graph (c).

If there is a link (directed or undirected) that connects two nodes X_i and X_j, we will say that X_i is a *neighbour* of X_j and vice versa. In case of directed links, the sets of parents (i.e. predecessors) of a node X_i according to $\mathcal{G}$ is denoted $pa(X_i)$. Similarly, its set of children (i.e. successors) is denoted $ch(X_i)$. For example, in Figure 3.1 (a), nodes A and C are neighbours, $pa(C) = \{A, B\}$ and $ch(C) = \{E, F\}$.

We define a *path* between two nodes X_i and X_j as a sequence of nodes $\{X_i, \ldots, X_j\}$ such that there exists a link for each pair of consecutive nodes in the sequence, i.e. they are neighbours. We say that a path is directed if each node in such sequence is a parent of the next one. Otherwise, if each node is a parent or a child we call it *undirected path* (a.k.a active path). In Figure 3.1 (a), $\{A, C, B\}$ is an undirected path while $\{A, C, F\}$ is a directed one.

For the scope of this dissertation, we should introduce the concept of *directed acyclic graph* (DAG) which is the underlying graphical structure of Bayesian networks and influence diagrams. A DAG is a directed graph with no cycles. A cycle in a graph is a directed path $\{X_1, ..., X_k\}$ where $X_1 = X_k$. As an example of DAG, we have the graph shown Figure 3.1 (a). Note that the sequence $\{B, C, F, B\}$ does not define a directed path and hence it is not a cycle. By contrast, the same path in Figure 3.1 (c) is directed and so it is a cycle.

3.2.2 Graphs and d-separation

DAGs can be used for modelling reasoning problems in a graphical and intuitive way. Moreover, a DAG represents the relations of dependence (and independence) between the variables in the problem. The nodes in the DAG correspond to the variables, therefore, from now on we will use both terms interchangeably. The set of possible values or states that a variable can take is called *domain*. In this dissertation, only discrete domains will be considered, i.e. variables can take a finite number of mutually exclusive states. As an example, let us consider the graphical representation of the car star problem as follows.

Example 2 (DAG for the car start problem) *Figure 3.2 depicts the DAG modelling the reasoning problem described in Example 1. It contains the following nodes:* $Fuel\_meter$ *indicating the output of the fuel meter standing and whose domain is* $\{full, half, empty\}$*;* $Fuel?$ *that shows the presence of fuel in the tank whose domain is* $\{yes, no\}$*;* $Start?$ *indicating if the car engiene stars with two possible values* $\{yes, no\}$*;* $Clean\_plugs$ *that indicates if the spark plugs are clean or not.*

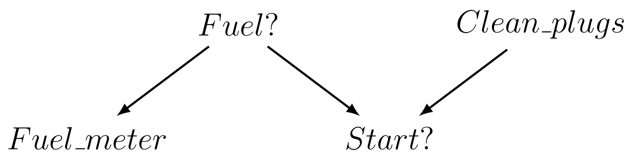
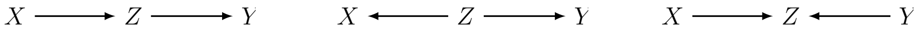


Figure 3.2: DAG modelling the car start problem described in Example 1.

In a DAG representing a reasoning problem, a link from X_i to X_j represents a direct dependence, i.e. X_i depends on X_j and vice versa. In Example 3.2, there is a link from $Fuel?$ to $Fuel\_meter$ which means that the amount of fuel in the tank is reflected in the output of the fuel meter standing. In general, we can say that every variable can be in only one state at one time point, and changes in those states can lead to changes in the remaining variables (not only in the neighbours). We will be interested in reasoning processes such as: "if there is fuel in the tank, and I know that the car does not start, should I clean the spark plugs". When we know the state in which a variable is, we say that there is evidence about that variable (i.e. it observed).

DAGs can be used to follow how a change of certainty in some variables affects to the rest of variables. Here we present the graphical criterion used for this kind of analysis, which is called *d-separation* [32, 53, 103]. When two variables X and Y are d-separated given the variable Z means that, a change of certainty in one variable does not affect to the other one when Z is known, i.e. they are independent. Otherwise, they are d-connected, i.e. they are dependent.

In short, for determining if two variables are d-separated, we should verify if all the possible (undirected) paths between them are blocked. A path is blocked if at least one of its connections is blocked. Figure 3.3 depicts the three possible types of connections in a DAG between a node X and Y connected through Z.



serial *diverging* *converging*

Figure 3.3: Different types of connections in a DAG

In a *serial connection*, also called *head to tail*, if we do not have evidence about Z, a knowledge of X will influence our knowledge on Y through Z, and vice versa. In other words, the flow of information can travel through the connection and therefore it is not blocked. By contrast, if Z is observed the flow cannot travel and so the connection is blocked. The same holds for a *diverging connection*, also called *tail to tail*, where the flow of information can travel unless Z is observed. Finally, in a *converging connection* or *head to head*, the flow can only travel when Z or any of its children are observed. Otherwise, the connection is blocked. Having explained these concepts, we can now give a definition of the d-separation as follows.

Definition 2 (d-separation) *Let* $\mathbf{X}$*,* $\mathbf{Y}$ *and* $\mathbf{Z}$ *be pairwise disjoint* [1] *sets of nodes in a DAG. Let us consider all undirected paths between them. Then* $\mathbf{X}$ *and* $\mathbf{Y}$ *are d-separated given* $\mathbf{Z}$ *if and only if along every of these paths there is an intermediate node* A *such that either:*

(i) the connection is serial or diverging and A *belongs to* $\mathbf{Z}$
or

(ii) the connection is converging and neither A *nor any of its descendants are in* $\mathbf{Z}$*.*

otherwise $\mathbf{X}$ *and* $\mathbf{Y}$ *are d-connected given* $\mathbf{Z}$*.*

As an example, let us consider the DAG shown in Figure 3.2: if we know the actual amount of fuel in the tank, the output of the meter does not give us any new information about the probability of the car to start: the path $\{Fuel\_meter, Fuel?, Start?\}$ contains a diverging connection with an observed node and hence the path it is blocked (case *i*). As there is not any other path connecting both variables, we conclude that $Fuel\_meter$ and $Start?$ are independent given $Fuel?$. On the other hand, if the car does not start, the amount of fuel in the tank will tell us if we have to clean the spark plugs: the path $\{Fuel?, Start?, Clean\_plugs\}$ contains a converging connection with an observed node so the path is not blocked (case *ii*). Therefore, we can say that $Fuel?$ and $Clean\_plugs$ are d-connected given $Start?$.

3.3 Probability Theory

In previous section, it was explained how the presence of dependency relations between variables in a reasoning problem under uncertainty can be represented using graphs. The next step is quantifying these dependencies. For example, in the car start problem, by analysing the graph we know that there is a dependency

[1] A family of sets is pairwise disjoint or mutually disjoint if every two different sets in the family are disjoint.

between the cleanness of the spark plugs and the fact that the car will not start. However, it could happen that 6 out of 10 times the car will start despite the dirty in the spark plugs. The quantification of the dependency relations is done by means of the (Bayesian) probability theory, whose basic concept are here explained.

3.3.1 Basics

When using the term “probability” in day we refer to a degree of confidence that an event of an uncertain nature may occur. An event might be the different outcomes of throwing a die, the outcome of a horse race, the weather configurations, or the possible failures of a piece of machinery. Thus, an event is an outcome of an experiment to which a probability is assigned. Probability theory deals with the formal foundations for assigning probabilities to uncertain events and the rules they should obey.

Formally, we define events by assuming that there is a set of all possible outcomes, called *event space* and denoted by Ω. For example, if we consider the experiment of throwing a die, the event space is $\Omega = \{1, 2, 3, 4, 5, 6\}$. In addition, we assume that there is a set of *measurable events* $\mathcal{S}$ to which we are willing to assign probabilities, i.e. a σ-algebra[2] on Ω. An event is any element of $\mathcal{S}$. For example, in the die experiment, the event $\{5\}$ represents the case in which the die shows the number 5. Another event can be $\{5\} \cup \{6\}$ which represent the case in which we obtain either a 5 or a 6.

The probability of an event α, denoted $P(\alpha)$, is a number quantifying the degree of confidence that α will occur. For example, the probability of getting a 5 is $P(\{5\}) = \frac{1}{6}$. Probabilities obey the following basic axioms:

Axiom 1 *For any event $\alpha \in \mathcal{S}$, its probability belongs to the interval $[0, 1]$. If*

[2] A σ-algebra on Ω is any sub-set $\mathcal{S}$ of the power set of Ω satisfying the following conditions: it contains the *empty event* $\emptyset$ and the *trivial event* Ω; it is closed under union, i.e. if $\alpha, \beta \in \mathcal{S}$, then $(\alpha \cup \beta) \in \mathcal{S}$; It is closed under complementation, i.e. if $\alpha \in \mathcal{S}$, then $(\Omega - \alpha) \in \mathcal{S}$.

$P(\alpha) = 1$ *we are certain that one of the outcomes in* α *occurs. By contrast, if* $P(\alpha) = 0$ *means that* α *is an imposible event.*

Axiom 2 *For any two events* $\alpha, \beta \in \mathcal{S}$ *such that they are mutually exclusive, the probability that either* α *or* β *occurs is:*

$$P(\alpha \cup \beta) = P(\alpha) + P(\beta)$$

From previous axioms, we can deduce that trivial event Ω, allowing all possible outcomes, has the maximal possible probability of 1, i.e. $P(\Omega) = 1$. For example, in the die experiment, it holds that $P(\{1\} \cup \{2\} \cup \{3\} \cup \{4\} \cup \{5\} \cup \{6\}) = 1$. By contrast, the empty event is an impossible event, i.e. $P(\emptyset) = 0$.

For any two events α and β, the probability that both α and β occur is called the *joint probability* of such events and denoted $P(\alpha \cap \beta)$ or simply $P(\alpha, \beta)$. Intuitively, the joint probability is the proportion of events in the event space Ω satisfying both α and β. Note that if both events are mutually exclusive it holds that $P(\alpha, \beta) = 0$. For example, in the die experiment, let us consider the three following events $\alpha = \{3\} \cup \{4\}$, $\beta = \{1\} \cup \{2\} \cup \{3\}$ and $\delta = \{4\} \cup \{5\} \cup \{6\}$. Then we can obtain that $P(\alpha, \beta) = P(\{3\}) = \frac{1}{6}$ while $P(\beta, \delta) = P(\emptyset) = 0$.

Rather than giving the probability of an event, in general we will be more interested in expressing its probability conditioned by other known factors (i.e. *conditional probability*). In the die example, we can say that the probability of getting a 5 is $\frac{1}{6}$ given that the die is fair. By contrast, the probability of a die turning up a 5 assuming that it is not fair could be, for instance, $\frac{2}{6}$.

Definition 3 (conditional probability) *The conditional probability of an event* α *given* β *can be written as* $P(\alpha|\beta) = p$*. This means that if* β *is true and everything else known is irrelevant for* α*, then the probability of* α *is* p*. The conditional probability of* α *given* β *can be computed as follows.*

$$P(\alpha|\beta) = \frac{P(\alpha, \beta)}{P(\beta)} \tag{3.1}$$

Intuitively, $P(\alpha|\beta)$ is the relative proportion of outcomes satisfying α among those that satisfy β. (Note that the conditional probability is not defined when β is an impossible event). From the definition of conditional probability, we can obtain the third axiom of probability calculus (a.k.a *fundamental rule*).

Axiom 3 *For any two events α and β, the joint probability of α and β is*

$$P(\alpha, \beta) = P(\alpha|\beta) \cdot P(\beta) = P(\beta|\alpha) \cdot P(\alpha) \tag{3.2}$$

From (3.1) and (3.2) we can obtain the *Bayes' rule* which is important since it allows us to compute the conditional probability $P(\alpha|\beta)$ from the "inverse" conditional probability $P(\beta|\alpha)$.

$$P(\alpha|\beta) = \frac{P(\beta|\alpha) \cdot P(\alpha)}{P(\beta)} \tag{3.3}$$

3.3.2 Probabilities for variables

By now, our discussion on probabilities has dealt with events. However, when modelling reasoning problems, it would be more natural to consider variables taking finite number of mutually exclusive states. Herein we present the concepts about probabilities in terms of variables.

More precisely, we will consider *random* (or *chance*) variables: those variables representing an entity of uncertain nature. That is, its value is subject to variations due to chance. For example, a random variable could represent the experiment of rolling a die with the possible values $\{1, 2, 3, 4, 5, 6\}$. Thus, we can say that the events are clustered around variables.

We will usually use upper-case roman letters to denote random variables and lower-case letters for its values (or states). For example, if X is a random variable, x will denote a generic state of X. The set of possible values that X can take is called *domain* and denoted Ω_X. In this dissertation, only discrete domains will be considered, i.e. variables can take a finite number of mutually exclusive states. Similarly, we use bold-face upper-case roman letter to denote sets of variables, e.g. $\mathbf{X} := \{X_1, \ldots, X_m\}$ is a set of m variables. The domain of $\mathbf{X}$ is defined as $\Omega_{\mathbf{X}} = \Omega_{X_1} \times, \Omega_{X_2}, \times \cdots \times \Omega_{X_m}$. Its elements are called *configurations* and denoted using bold-face lower-case letters, e.g. $\mathbf{x} \in \Omega_{\mathbf{X}}$.

The uncertainty of a random variable is quantified by means of a *probability distribution*, which is a function defining the probability of each possible event described by the random variable. Formally, a probability distribution can be defined as follows.

Definition 4 (probability distribution) *Let X be a (random) variable with domain Ω_X. Then a probability distribution over X, denoted $P(X)$, is a mapping $P : \Omega_X \to [0, 1]$ such that*

$$\sum_{x \in\ \Omega_X} P(X = x) = 1$$

where $P(X = x)$ is the probability of X being x.

Note that if the variable is obvious from the context, we will simply write $P(x)$ instead of $P(X = x)$. A probability distribution can also be seen as a vector of n probabilities summing 1 where n is the number of states of the variable:

$$P(X) = \left[\begin{array}{c} P(x_1) \\ P(x_2) \\ \vdots \\ P(x_n) \end{array}\right] \begin{array}{c} x_1 \\ x_2 \\ \\ x_n \end{array}$$

The probability distribution of only one variable is often called *marginal distribution*. However, in many situations we are interested in questions that involve the values of several random variables. For example, we might be interested in the event $Fuel? = empty$ and $Clean\_plugs = false$. To discuss such events, we need to consider the joint distribution over these two random variables. In general, the joint probability over a set of variables is defined as follows.

Definition 5 (joint probability distribution) *Let* $\mathbf{X} := \{X_1, \ldots, X_k\}$ *be a set of* k *variables with domain* $\Omega_{\mathbf{X}}$. *Then a probability distribution over* $\mathbf{X}$, *denoted either* $P(\mathbf{X})$ *or* $P(X_1, X_2, \ldots, X_k)$, *is a mapping* $P : \Omega_{\mathbf{X}} \to [0, 1]$ *such that*

$$\sum_{\mathbf{x} \in\ \Omega_{\mathbf{X}}} P(\mathbf{x}) = 1$$

Let us consider the particular case two variables X and Y takes n and m states respectively, then the joint probability $P(X, Y)$ can be represented as a table of $n \cdot m$ entries as depicted below. Note that the sum of all entries should be 1.

$$P(X,Y) = \begin{array}{cc} \begin{array}{cccc} y_1 & y_2 & & y_m \end{array} & \\ \begin{bmatrix} P(x_1,y_1) & P(x_1,y_2) & \cdots & P(x_1,y_m) \\ P(x_2,y_1) & P(x_2,y_2) & \cdots & P(x_2,y_m) \\ \vdots & \vdots & \ddots & \vdots \\ P(x_n,y_1) & P(x_n,y_2) & \cdots & P(x_n,y_m) \end{bmatrix} & \begin{array}{c} x_1 \\ x_2 \\ \\ x_n \end{array} \end{array} \tag{3.4}$$

The joint distribution of two random variables has to be consistent with the marginal distribution, in that we can state the *rule of total probability* as follows.

Proposition 1 (rule of total probability) *Let* $P(X, Y)$ *be a joint probability distribution for two variables* X *and* Y. *As* Ω_X *and* Ω_Y *are exhaustive sets of mutually exclusive states of* X *and* Y, *it holds that:*

$$P(x) = P(x, y_1) + P(x, y_2) + \ldots + P(x, y_m) = \sum_{y \in \Omega_Y} P(x, y) \tag{3.5}$$

for each $x \in \Omega_X$.

In a more compact notation, we may write this rule as $P(X) = \sum_Y P(X, Y)$. This operation is often referred as to *sum-marginalization* or simply *marginalization*. That is, in previous expression we may say that we "sum-marginalize out variable Y from $P(X, Y)$". Considering the representation shown in (3.4), the rule of total probability implies that, for computing $P(X)$, we have to sum all the entries in the same row.

The notion of conditional probability can also be considered for random variables. Thus, we need to introduce the concept of *conditional probability distribution* over some variables conditioned over the states of some others.

Definition 6 (conditional probability distribution) *Let* $\mathbf{X}$ *and* $\mathbf{Y}$ *be two sets of disjoint variables with domains* $\Omega_{\mathbf{X}}$ *and* $\Omega_{\mathbf{Y}}$*. A conditional probability distribution (CPD) of* $\mathbf{X}$ *given* $\mathbf{Y}$*, denoted* $P(\mathbf{X}|\mathbf{Y})$ *is a mapping* $P : \Omega_{\mathbf{X}} \times \Omega_{\mathbf{Y}} \rightarrow [0, 1]$ *such that*

$$\sum_{\mathbf{x} \in \, \Omega_{\mathbf{X}}} P(\mathbf{x}|\mathbf{y}) = 1$$

for each $\mathbf{y} \in \Omega_{\mathbf{Y}}$*.*

Intuitively, $P(\mathbf{X}|\mathbf{Y})$ assigns a probability distribution over $\mathbf{X}$ to each configuration in $\Omega_{\mathbf{Y}}$. The set of variables on the left of the conditioning bar (i.e. in the head) are often called *conditioned variables*. By contrast, those on the right (i.e. in the tail) are called *conditioning variables*.

If the variables X and Y takes n and m states respectively, then $P(X|Y)$ can be represented as a table of $n \cdot m$ entries as depicted below. Note that, in the following representation, the sum of all the entries in a column should be 1.

$$\begin{array}{c} \\ P(X|Y) = \end{array} \begin{array}{cccc} y_1 & y_2 & & y_m \\ \big[P(X|y_1) & P(X|y_2) & \cdots & P(X|y_m)\big] \end{array} \quad =$$

$$= \begin{array}{ccccc} y_1 & y_2 & & y_m & \\ P(x_1|y_1) & P(x_1|y_2) & \cdots & P(x_1|y_m) & x_1 \\ P(x_2|y_1) & P(x_2|y_2) & \cdots & P(x_2|y_m) & x_2 \\ \vdots & \vdots & \ddots & \vdots & \\ P(x_n|y_1) & P(x_n|y_2) & \cdots & P(x_n|y_m) & x_n \end{array} \tag{3.6}$$

When applied to variables, the fundamental rule (see (3.2)) can be written as follows. Note that operations with probability distributions, such as multiplication or division, are made element-wise.

$$P(X, Y) = P(X|Y) \cdot P(Y) = P(Y|X) \cdot P(X) \tag{3.7}$$

And therefore, the set of conditional probability distribution $P(X|Y)$ can be compute as shown below.

$$P(X|Y) = \frac{P(X, Y)}{P(Y)} \tag{3.8}$$

Finally, from (3.7) and (3.8) we can obtain the Bayes' rule for probability distributions:

$$P(X|Y) = \frac{P(Y|X) \cdot P(X)}{P(Y)} \tag{3.9}$$

3.3.3 Marginal and conditional independence

In Section 3.2.2 we already introduced the concept of *independence* between variables and a graphical criterion for its detection. Intuitively, two variables are independent if a change of certainty in one variable does not change the certainty for the other one. This is reflected in the probability distributions of such variables in the sense that, if two variables X and Y are independent we expect $P(X|Y)$ to be equal to $P(X)$. The independence between two variables (or sets of variables) is often called *marginal independence* and can be defined as follows [70].

Definition 7 (marginal independence) *Let* $\mathbf{X}$ *and* $\mathbf{Y}$ *be two sets of variables. We say that* $\mathbf{X}$ *and* $\mathbf{Y}$ *are (marginal) independent and denote* $I(\mathbf{X} \perp \mathbf{Y})$ *if and only if:*

$$P(\mathbf{x}|\mathbf{y}) = P(\mathbf{x})$$

for each possible value of $(\mathbf{x}, \mathbf{y}) \in \Omega_{\mathbf{X}} \times \Omega_{\mathbf{Y}}$ *such as* $P(\mathbf{y}) > 0$.

Previous definition implies that, given a configuration of the conditioned variables, the probability is the same regardless of the configuration of the conditioning variables. As an example, let x be a state of X and let Y be a variable taking m values, if X and Y are independent then it holds that $P(x \mid y_1) = P(x \mid y_2) = \ldots = P(x \mid y_m)$. If we consider the table representation shown in (3.6), the independence of X and Y implies that all the values in a row are the same. A more complex example of independence is shown below.

Example 3 *Let as consider the following probability distributions over the variables X and Y:*

$$P(X|Y) = \begin{array}{cc} \begin{array}{ccc} y_1 & y_2 & y_3 \end{array} & \\ \begin{bmatrix} 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 \\ 0.1 & 0.1 & 0.1 \\ 0.4 & 0.4 & 0.4 \end{bmatrix} & \begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \end{array} \qquad P(Y) = \begin{bmatrix} 0.5 \\ 0.2 \\ 0.3 \end{bmatrix} \begin{array}{c} y_1 \\ y_2 \\ y_3 \end{array}$$

The marginal probability of X can be calculated by sum-marginalizing out variable Y from the joint probability distribution:

$$P(X) = \sum_Y P(X|Y) \cdot P(Y) = \begin{bmatrix} 0.25 \\ 0.25 \\ 0.1 \\ 0.4 \end{bmatrix} \begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \end{array}$$

It holds that $P(x|y) = P(x)$ for any each possible value of $(x, y) \in \Omega_X \times \Omega_Y$. Thus, X and Y are independent, i.e. $I(X \perp Y)$.

Even though marginal independence is a useful property, it is not a frequent situation finding two marginal independent variables. A more usual case is when two variables are independent given some others. In that case we say that there is a *conditional independence* (CI) [70], which can be defined as follows.

Definition 8 (conditional independence) *Let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ three sets of variables. Then $\mathbf{X}$ and $\mathbf{Y}$ are conditional independent given $\mathbf{Z}$, denoted $I(\mathbf{X} \perp \mathbf{Y} | \mathbf{Z})$ if and only if:*

$$P(\mathbf{x}|\mathbf{y}, \mathbf{z}) = P(\mathbf{x}|\mathbf{z})$$

for each possible value of $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \Omega_{\mathbf{X}} \times \Omega_{\mathbf{Y}} \times \Omega_{\mathbf{Z}}$, such as $P(\mathbf{y}, \mathbf{z}) > 0$.

As stated for marginal independence, previous definition also implies the presence of some identical values. Now, if variables X and Y are independent given Z, then it holds $P(x \mid y_1, z) = P(x \mid y_2, z) = \ldots = P(x \mid y_m, z)$ for each possible value of $(x, z) \in \Omega_X \times \Omega_Z$.

Example 4 *Let as consider the following probability distributions over the variables X, Y and Z:*

$$P(X|Y,Z) = \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & & \\ 0.25 & 0.25 & 0.1 & 0.4 & y_1 & \\ 0.25 & 0.25 & 0.1 & 0.4 & y_2 & z_1 \\ 0.25 & 0.25 & 0.1 & 0.4 & y_3 & \\ 0.0 & 0.5 & 0.3 & 0.2 & y_1 & \\ 0.0 & 0.5 & 0.3 & 0.2 & y_2 & z_1 \\ 0.0 & 0.5 & 0.3 & 0.2 & y_3 & \end{array} \qquad P(Y) = \begin{array}{cc} 0.5 & y_1 \\ 0.3 & y_2 \\ 0.2 & y_3 \end{array}$$

$P(X|Z)$ can be calculated by multiplying[3] previous distributions and marginalizing out variable Y:

$$P(X|Z) = \sum_{Y} P(X|Y,Z) \cdot P(Y) = \begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & \\ 0.25 & 0.25 & 0.1 & 0.4 & z_1 \\ 0.0 & 0.5 & 0.3 & 0.2 & z_2 \end{array}$$

[3] the multiplication of two probability distributions is the element-wise product.

It holds that $P(x|y,z) = P(x|z)$ *for each value of* $(x,y,z) \in \Omega_X \times \Omega_Y \times \Omega_Z$*. Thus,* X *and* Y *are conditionally independent given* Z*, i.e.* $I(X \perp Y|Z)$*.*

Note that the marginal independence can be seen as a particular case of the conditional one: if we have that $\mathbf{Z} = \emptyset$, the conditional independence is reduced to a marginal one. Thus instead of writing $I(\mathbf{X} \perp \mathbf{Y}|\emptyset)$ we write $I(\mathbf{X} \perp \mathbf{Y})$.

CI can be used to facilitate the acquisition, representation, and inference of probabilistic knowledge. In particular, in Section 3.4, we will see how probabilistic graphical models can capture such independencies for reducing the representation of the joint probability distribution over a set of variables.

3.3.4 More general forms of independence

It is well-known that the notion of conditional independence is too restrictive to capture independencies that only hold for certain subsets of the variable domains (i.e. in certain partitions or subsets of tuples in the probability distribution). Herein we review some other types of independencies which are more general than CIs, namely *context-specific independence (CSI) [7] , partial conditional independence (PCI)* [92, 75] and *contextual weak independence (CWI)* [110, 10].

3.3.4.1 Context-specific independencies

Independencies that only hold for certain contexts, i.e. given a specific assignment of values to some variables, are called *context-specific independencies (CSIs)*. These were first introduced by Boutilier [7] and can be defined as follows.

Definition 9 (context-specific independence) *Let* $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ *and* $\mathbf{C}$ *be pairwise disjoint sets of variables, then* $\mathbf{X}$ *and* $\mathbf{Y}$*, are contextually independent given* $\mathbf{Z}$ *and the context* $\mathbf{C} = \mathbf{c}$*, denoted* $I(\mathbf{X} \perp \mathbf{Y}|\mathbf{Z}, \mathbf{c})$*, if:*

$$P(\mathbf{x}|\mathbf{y}, \mathbf{z}, \mathbf{c}) = P(\mathbf{x}|, \mathbf{z}, \mathbf{c})$$

for each possible value of $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \Omega_{\mathbf{X}} \times \Omega_{\mathbf{Y}} \times \Omega_{\mathbf{Z}}$*, such as* $P(\mathbf{y}, \mathbf{z}, \mathbf{c}) > 0$.

Previous definition implies that, if variables X and Y are independent given Z and the context $C = c$, then it holds that $P(x \mid y_1, z, c) = P(x \mid y_2, z, c) = \ldots = P(x \mid y_m, z, c)$ for each possible value of $(x, z) \in \Omega_X \times \Omega_Z$.

Example 5 *In order to introduce this new kind of independence, let us consider the following probability distributions:*

$$P(X|Y,Z) = \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & & \\ 0.25 & 0.25 & 0.1 & 0.4 & y_1 & \\ 0.25 & 0.25 & 0.1 & 0.4 & y_2 & z_1 \\ 0.25 & 0.25 & 0.1 & 0.4 & y_3 & \\ 0.1 & 0.5 & 0.2 & 0.2 & y_1 & \\ 0.1 & 0.5 & 0.2 & 0.2 & y_2 & z_2 \\ 0.15 & 0.35 & 0.3 & 0.2 & y_3 & \end{array} \qquad P(Y) = \begin{array}{cc} 0.5 & y_1 \\ 0.3 & y_2 \\ 0.2 & y_3 \end{array}$$

The distribution $P(X|Z)$ can be calculated by multiplying previous distributions and sum-marginalizing out variable Y:

$$P(X|Z) = \sum_Y P(X|Y,Z) \cdot P(Y) = \begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & \\ 0.25 & 0.25 & 0.1 & 0.4 & z_1 \\ 0.11 & 0.47 & 0.22 & 0.2 & z_1 \end{array}$$

We can observe that X and Y are not conditionally independent given Z: it is not true that $P(x|y,z) = P(x|z)$ for each $(x,y,z) \in \Omega_X \times \Omega_Y \times \Omega_Z$ (see Definition 8). However, such condition is true if we only consider the configurations where Z takes the values z_1. Thus, we say that X and Y are contextually independent given the context $Z = z_1$ i.e. $I(X \bot Y | Z = z_1)$. By contrast, they are not contextually independent given the context $Z = z_2$.

Note that CSI is a more general form of independence: a CI is basically a CSI that holds for all the contexts. In Example 4 (page 37) variables X and Y are conditional independent given Z, i.e. $I(X \bot Y | Z)$. We can also say that X and Y are contextually independent given all the states of Z, i.e. $I(X \bot Y | Z = z_1)$ and $I(X \bot Y | Z = z_1)$.

3.3.4.2 Partial conditional independencies

Partial conditional independence (PCI) is a generalization of CSI: a PCI is basically a CSI that holds if a certain subset of the variable domains are considered. This type of independence, which is defined below, was introduced by Pensar [92] and by Lintusaari [75].

Definition 10 (partial conditional independence) *Let* $\mathbf{X}, \mathbf{Y}$ *and* $\mathbf{C}$ *three disjoint sets of variables. We say that* $\mathbf{X}$ *and* $\mathbf{Y}$ *are partial conditional independent in the domain* $\Omega^*_{\mathbf{Y}} \subseteq \Omega_{\mathbf{Y}}$ *given the context* $\mathbf{C} = \mathbf{c}$ *, denoted* $I(\mathbf{X} \perp \mathbf{Y} | \Omega^*_{\mathbf{Y}}, \mathbf{c})$*, if:*

$$P(\mathbf{x}|\mathbf{y}, \mathbf{c}) = P(\mathbf{x}|\mathbf{y}', \mathbf{c})$$

holds for all $(\mathbf{x}, \mathbf{y}), (\mathbf{x}, \mathbf{y}') \in \Omega_{\mathbf{X}} \times \Omega^*_{\mathbf{Y}}$ *whenever* $P(\mathbf{y}, \mathbf{c}) = P(\mathbf{y}', \mathbf{c})$

Note that if $\mathbf{C} = \emptyset$. then $I(\mathbf{X} \perp \mathbf{Y} | \Omega^*_{\mathbf{Y}})$ is simply called *partial independence (PCI)*. Here below an example of this type of independence is given.

Example 6 *Let us consider the probability distribution* $P(X|Y, Z)$ *depicted in Example 5, where* X *and* Y *are contextually independent give* $Z = z_1$ *but not* $Z = z_2$*. If we focus on the probability values consistent with* $Z = z_2$ *we can observe that:*

$$\begin{aligned}
P(x_1|y_1, z_2) = P(x_1|y_2, z_2) \neq P(x_1|y_3, z_2) \\
P(x_2|y_1, z_2) = P(x_2|y_2, z_2) \neq P(x_2|y_3, z_2) \\
P(x_3|y_1, z_2) = P(x_3|y_2, z_2) \neq P(x_3|y_3, z_2) \\
P(x_4|y_1, z_2) = P(x_4|y_2, z_2) = P(x_4|y_3, z_2)
\end{aligned}$$

If we restrict the domain of Y *to* $\{y_1, y_2\}$*, the conditions of independence given the context* $Z = z_2$ *are satisfied. Therefore,* X *and* Y *are partially conditional independent in the domain* $\{y_1, y_2\}$ *and given the context* $Z = z_2$*, i.e.* $I(X \perp Y| \{y_1, y_2\}, z_2)$*.*

3.3.4.3 Contextual-weak independencies

Another form of independence generalizing CSI is *contextual weak independence (CWI)*, introduced by Wong and Butz [110, 10]. With CWIs, we might say, for instance, that variables X and Y are weakly independents given the context $C = c$. Unlike CSIs, in this case the partition in which the independence holds is not explicitly defined by the context $C = c$. Formally, it can be defined as follows.

Definition 11 (contextual-weak independence) *Let* $\mathbf{X}, \mathbf{Y}$ *and* $\mathbf{C}$, *be pairwise disjoint sets of variables. Given a distribution* $P(\mathbf{X}|\mathbf{Y}, \mathbf{C})$*, let us define* T *as the set of tuples or configurations whose probability is higher that 0, i.e.* $T := \{t = (\mathbf{x}, \mathbf{y}, \mathbf{c}) | P(\mathbf{x}|\mathbf{y}, \mathbf{c}) > 0\}$. *Let* $\theta(\mathbf{XC} = \mathbf{c})$ *and* $\theta(\mathbf{C} = \mathbf{cY})$ *be partitions* [4] *of* T. *We say that* $\mathbf{X}$ *and* $\mathbf{Y}$ *are weakly independent given the context* $\mathbf{C} = \mathbf{c}$ *, denoted* $WI(\mathbf{X} \perp \mathbf{Y} | \mathbf{c})$ *if the following two conditions hold:*

(i) $\theta(\mathbf{XC} = \mathbf{c}) \circ \theta(\mathbf{C} = \mathbf{cY}) = \theta(\mathbf{C} = \mathbf{cY}) \circ \theta(\mathbf{XC} = \mathbf{c})$ *where* $\circ$ *denotes the composition*[5] *operator, and*

(ii) there exists an equivalence class π *in* $\theta(\mathbf{XC} = \mathbf{c}) \circ \theta(\mathbf{C} = \mathbf{cY})$ *such that: for any given* $(\mathbf{x}, \mathbf{y}) \in \Omega^{\pi}_{\mathbf{X}} \times \Omega^{\pi}_{\mathbf{Y}}$*:*

$$P(\mathbf{x}|\ \mathbf{y}, \mathbf{c}) = P(\mathbf{x}|\ \mathbf{c})$$

where $\Omega^{\pi}_{\mathbf{X}}$ *and* $\Omega^{\pi}_{\mathbf{Y}}$ *are the sets of values of* $\mathbf{X}$ *and* $\mathbf{Y}$ *appearing in* π.

Condition *(i)* states that the composite relation $\theta(\mathbf{XC} = \mathbf{c}) \circ \theta(\mathbf{C} = \mathbf{cY})$ is an equivalence relation. This is a necessary condition for the conditional weak independence to hold. Condition *(ii)* says that $\mathbf{X}$ and $\mathbf{Y}$ are independent in context

[4] $\theta(\mathbf{XC} = \mathbf{c})$ is the *equivalence relation* induced by $\mathbf{X}$ and by the configuration $\mathbf{C} = \mathbf{c}$. Each equivalence class $\pi \in \theta(\mathbf{XC} = \mathbf{c})$ is a subset of tuples of T with the same configuration for $\mathbf{X}$ and with $\mathbf{C} = \mathbf{c}$.

[5] Given two equivalence relations θ_1 and θ_2, the binary operator $\circ$ called *composition* is defined by: for $t_i, t_k \in T$, $t_i(\theta_1 \circ \theta_2)t_k$ if for some $t_j \in T$ it holds that $t_i \theta_1 t_j$ and $t_j \theta_2 t_k$.

$\mathbf{C} = \mathbf{c}$ if we only consider the tuples present in any of the equivalence class of the composite relation.

Example 7 *Let us consider the CPD shown below. We aim to proof that* D *and* B *are weakly independent given the context* $A = a_1$*, i.e.* $WI(D \perp B | A = a_1)$*.*

$$P(D|B,A) = \begin{array}{c c c c c c c} d_1 & d_2 & d_3 & d_4 & & & \\ 0.3 & 0.7 & 0 & 0 & b_1 & & \\ 0.3 & 0.7 & 0 & 0 & b_2 & a_1 & \\ 0 & 0 & 0.1 & 0.9 & b_3 & & \\ 0 & 0 & 0.1 & 0.9 & b_4 & & \\ 0.6 & 0.4 & 0 & 0 & b_1 & & \\ 0.8 & 0.2 & 0 & 0 & b_2 & a_2 & \\ 0 & 0 & 0.2 & 0.8 & b_3 & & \\ 0 & 0 & 0.3 & 0.7 & b_4 & & \end{array}$$

If only those configurations with a probability value higher than 0 *are considered,* $P(D|B,A)$ *can be depicted as follows.*

	D	B	A	$P(D\|B,A)$
t_1	d_1	b_1	a_1	0.3
t_2	d_2	b_1	a_1	0.7
t_3	d_1	b_2	a_1	0.3
t_4	d_2	b_2	a_1	0.7
t_5	d_3	b_3	a_1	0.1
t_6	d_4	b_3	a_1	0.9
t_7	d_3	b_4	a_1	0.1
t_8	d_4	b_4	a_1	0.9

	D	B	A	$P(D\|B,A)$
t_9	d_1	b_1	a_2	0.6
t_{10}	d_2	b_1	a_2	0.4
t_{11}	d_1	b_2	a_2	0.8
t_{12}	d_2	b_2	a_2	0.2
t_{13}	d_3	b_3	a_2	0.2
t_{14}	d_4	b_3	a_2	0.8
t_{15}	d_3	b_4	a_2	0.3
t_{16}	d_4	b_4	a_2	0.7

Let us now consider the following equivalence relations:

$$\theta_1(DA = a_1) = \{\{t_1, t_3\}, \{t_2, t_4\}, \{t_5, t_7\}, \{t_6, t_8\}\}$$

$$\theta_2(BA = a_1) = \{\{t_1, t_2\}, \{t_3, t_4\}, \{t_5, t_6\}, \{t_7, t_8\}\}$$

Condition (i) in Definition 11 is satisfyed as the composition of θ_1 and θ_2 is also an equivalence relation as it holds that:

$$\theta_1 \circ \theta_2 = \{\pi_1 = \{t_1, t_2, t_3, t_4\}, \pi_2 = \{t_5, t_6, t_7, t_8\}\} = \theta_2 \circ \theta_3 \tag{3.10}$$

Given the equivalence class π_1 obtained in (3.10)*, we can compute the following domains for variables D and B:*

$$\Omega_D^{\pi_1} = \{d_1, d_2\} \quad \Omega_B^{\pi_1} = \{b_1, b_2\} \tag{3.11}$$

If we consider the values of $P(D|B, A)$ consistent with these domains, it holds that:

$$P(d_1|b_1, a_1) = 0.3 = P(d_1|b_2, a_1)$$
$$P(d_2|b_1, a_1) = 0.7 = P(d_2|b_2, a_1)$$

in other words, D and B are independent given $A = a_1$ with respect to the new domains. Thus condition (ii) is satisfied and we can say that D and B are weakly independent given the context $A = a_1$, i.e. $WI(D \perp B|A = a_1)$. Now we can similarly proceed and compute the new domains in the equivalence class π_2:

$$\Omega_D^{\pi_2} = \{d_3, d_4\} \quad \Omega_B^{\pi_2} = \{b_3, b_4\} \tag{3.12}$$

Now we observe that:

$$P(d_3|b_3, a_1) = 0.1 = P(d_3|b_4, a_1)$$
$$P(d_4|b_3, a_1) = 0.9 = P(d_4|b_4, a_1)$$

Thus, variables D and B are also independent given $A = a_1$ with respect to the new domains in equivalence class π_2.

If it holds that $\mathbf{C} = \emptyset$, it will be simply called *weak independence* (i.e. non contextual) and denoted $WI(\mathbf{X} \perp \mathbf{Y})$.

3.4 Probabilistic graphical models

A *probabilistic graphical mode*l (PGM), also known as *probabilistic network*, is a visual representation of a reasoning problem under uncertainty used to infer new knowledge. The representational components of a PGM are a *qualitative* and a *quantitative*. The former encodes a set of (conditional) dependence and independence relations among a set of variables. The quantitative component, on the other hand, quantifies the strength of such dependencies.

More formally, a PGM contains three elements $\langle \mathbf{X}, P, \mathcal{G} \rangle$ where $\mathbf{X}$ is the set of variables in the problem with a joint probability distribution $P(\mathbf{X})$ and $\mathcal{G}$ is a graph that represents the dependence (and independence) relations between the variables. Note that $P(\mathbf{X})$ and $\mathcal{G}$ are related in the sense that all the independencies between variables expressed in $\mathcal{G}$ due to the d-separation criterion are also conditional independencies [6] in the probability distribution P.

There are several types of PGMs such as *Bayesian networks* [89, 91], *chain graphs* [100], *Markov decision models* [49], *Influence diagrams* [87, 56], etc. The latter are fundamental for the scope of this dissertation and hence they are described more in detail in Section 4.3. Bayesian networks, on the other hand, are not directly connected with the topic of this thesis. Yet, in the following section we briefly introduce them for two reasons: they are one of the types of PGMs more commonly used and an influence diagram can be seen as an augmented bayesian network with additional kinds of nodes.

3.4.1 Bayesian networks

A *Bayesian network* (BN) [89, 91] is a class of PGM representing a joint probability distribution over a finite set of random variables. The nodes in the network represent the variables in a reasoning problem, and the links represent the (condi-

[6] For more details about d-separation and conditional independence see Sections 3.2.2 and 3.3.3 respectively.

tional) dependencies and independencies among the variables. More formally, a BN can be defined as follows.

Definition 12 (Bayesian network) *A Bayesian network (BN) is a tuple* $\langle \mathbf{X}, \mathbf{P}, \mathcal{G} \rangle$ *where:*

- $\mathbf{X}$ *is a set of discrete random variables.*
- $\mathcal{G}$ *is a DAG where each node represents a variable in* $\mathbf{X}$*.*
- $\mathbf{P}$ *is a set of CPDs, containing one distribution* $P(X \ |pa(X))$ *for each* $X \in \mathbf{X}$ *where* $pa(X)$ *is the set of parents*[7] *of* X *according to* $\mathcal{G}$*.*

The set $\mathbf{P}$ defines a joint probability distribution through the factorisation (a.k.a the *chain rule* for Bayesian networks):

$$P(\mathbf{X}) = \prod_{X \in \mathbf{X}} P(X|pa(X)) \tag{3.13}$$

From previous equation we can observe that, in a BN, each variable $X \in \mathbf{X}$ is conditional independent of its non-decendants given its parents. Moreover, BNs admits d-seaparation as presented in Definition 2, page 27. This is the power of BNs, which provide space saving in the representation of the joint probability.

Example 8 (BN for the car start problem) *Figure 3.4 depicts the graph* $\mathcal{G}$ *of a BN modelling the reasoning problem described in Example 1. The set of random variables is* $\mathbf{X} = \{F, C, M, S\}$*, while the set of probability distributions is* $\mathbf{P} = \{P(F), P(C), P(M|F), P(S|F, C)\}$*. The numerical values of these distributions are reported here below in a table form, with the corresponding states depicted in grey.*

[7] If a node X has no parents (i.e. $pa(X) = \emptyset$), then the distribution associated to X in $\mathbf{P}$ is the marginal distribution $P(X)$.

$$P(F) = \begin{array}{cc} \begin{bmatrix} 0.98 \\ 0.02 \end{bmatrix} & \begin{matrix} f \\ e \end{matrix} \end{array}, \; P(C) = \begin{array}{cc} \begin{bmatrix} 0.96 \\ 0.04 \end{bmatrix} & \begin{matrix} y \\ n \end{matrix} \end{array},$$

$$P(M|F) = \begin{array}{cc} \begin{matrix} f & \quad e \end{matrix} & \\ \begin{bmatrix} 0.39 & 0.001 \\ 0.60 & 0.001 \\ 0.01 & 0.998 \end{bmatrix} & \begin{matrix} f \\ h \\ e \end{matrix} \end{array} \quad P(S|F,C) = \begin{array}{cc} \begin{matrix} \quad y & & \quad n & \\ f & e & f & e \end{matrix} & \\ \begin{bmatrix} 0.99 & 1.0 & 0.99 & 1.0 \\ 0.01 & 0.0 & 0.01 & 0.0 \end{bmatrix} & \begin{matrix} y \\ n \end{matrix} \end{array}$$

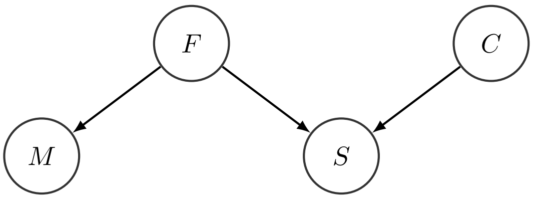


Figure 3.4: Graph of a BN modelling the car start problem. We have used the abbreviations F ($Fuel?$), C ($Clean\_Plugs$), S ($Start?$), and M ($Fuel\_Meter$).

Considering previous example, a table representing the joint probability distribution $P(F, C, M, S)$ requires 48 entries (i.e. $2 \cdot 2 \cdot 3 \cdot 2$) while only 18 are required due to the factorisation.

Making probabilistic inference in BNs (a.k.a *belief updating*) consists of the computation of the posterior probability distribution for a set of variables of interest given some evidence over some other variables. In Example 8, we might be interested, for instance, in computing $P(C|S = n, M = f)$, i.e., the probability for the plugs to be clean knowing that the car does not start and the fuel meter indicates that the tank is full. During the last three decades, probabilistic inference has attracted the interest of researchers in the field of BNs, leading to the development of both, exact [116, 42, 79] and approximate algorithms [61, 90, 38].

Chapter 4

Probabilistic Graphical Models for Decision Reasoning

Since their introduction in the mid 1970s, *influence diagrams (IDs)* [87, 56] have become a popular and standard modelling tool for decision-making under uncertainty. In this chapter, we introduce this kind of probabilistic graphical model: Section 4.1 presents some basic concepts about decision theory, which provides a formal foundation for IDs; Section 4.2 describes *decision trees*, which are another tool for representing and solving decision problems. Section 4.3 introduces the notation and basics concepts about IDs; finally, Section 4.5 describes some algorithms present in the literature for evaluating IDs, namely *variable elimination* [116, 64], *arc reversal* [102] and *lazy evaluation* [79].

4.1 Decision theory

In the task of decision making under uncertainty, the decision maker has to choose between a set of possible actions (take a decision). Each of them can lead to one of several outcomes (or consequences), which the decision maker can prefer to different degrees. However, the outcome of each action might not be known with certainty. Thus, both the probabilities of each outcome and the preferences must be taken into account. In order to introduce the *decision problems under uncertainty*, let us first consider the following example.

Example 9 (Investor's decision problem [70]) *An investor has two decide between two investment opportunities: he can spend his money in a high-technology company (t), where he can make a profit of* $\$4$ *million with 20 percent probability and* $\$0$ *with 80 percent probability; or he can invest in pork belly futures (p), where he can make* $\$3$ *million with 25 percent probability and* $\$0$ *with 75 percent probability.*

In the previous example, the possible actions correspond to the kind of inversion, i.e. t or p. The uncertain outcomes are to obtain a profit or not. We assume that this profit determines the preference for each action. The expected utility (or profit) of each action can be computed as follows:

$$EU(t) = 0.2 \cdot 4 + 0.8 \cdot 0 = 0.8$$
$$EU(p) = 0.25 \cdot 3 + 0.75 \cdot 0 = 0.75$$

As it holds that $EU(t) > EU(p)$, then we can say that the best option is to invest in a high-technology company (t). More formally, we can define a decision problem under uncertainty as follows:

Definition 13 *A decision problem under uncertainty is defined by the following elements:*

- *a set of outcomes* $\mathcal{O} := \{o_1, \ldots, o_n\}$*;*
- *a set of possible actions that the decision maker can take,* $\mathcal{A} := \{a_1, \ldots, a_k\}$*;*
- *a probability distribution* $P : \mathcal{O} \times \mathcal{A} \to [0, 1]$ *such that* $\sum_{o \in \mathcal{O}} P(o|a) = 1$ *for each* $a \in \mathcal{A}$*.*
- *a utility function* $U : \mathcal{O} \times \mathcal{A} \to \mathbb{R}$ *that represents the user preferences.*

Solving a decision problem is done by following the *principle of maximum expected utility (MEU)* [83]: if we have to choose between a set of possible actions, we should choose the one with the maximum expected utility. More formally:

Definition 14 (MEU principle) *The principle of maximum expected utility asserts that, in a decision problem under uncertainty, we should choose the action* $a \in \mathcal{A}$ *that maximizes the expected utility:*

$$EU(a) = \sum_{o \in \mathcal{O}} P(o|a) \cdot U(o, a) \tag{4.1}$$

Note that in previous definitions, we consider that only one decision is taken. However, in real problems, we usually have to take a sequence of decisions. Decision trees and influence diagrams, which are explained in the following sections, can also represent this kind of more complex decision problems.

4.2 Decision trees

Decision trees [96], which are the classical framework for representing and solving decision problems under uncertainty. Even though this thesis is not focused on decision trees, we should first briefly introduce this class of PGM for a better understanding of influence diagrams, which are a compact representation of decision trees. Let us first consider the oil wildcatter's problem detailed in Example 10.

Example 10 (the oil wildcatter's decision problem [96, 105]) *An oil wildcatter must decide whether to drill* ($D = d$) *or not* ($D = nd$)*. He is uncertain whether the amount of oil* (O) *in the place is soaking* (s)*, wet* (w) *or empty* (e)*. The profits are* $200M€$ *in case of being soaking and* $50M€$ *in case of being wet. By contrast, if there is not oil, he will lose* $70M€$*. The wildcatter can make seismic tests* (S) *that will give a closed reflection pattern* (c) *indicating much oil, an open pattern* (o) *indicating for some oil, or a diffuse pattern* (d) *denoting almost no hope for oil. The tests imply a cost of* $10M€$ *and they are not completely reliable, thus he also has to decide if it is worthy to do the test* ($T = t$) *or not* ($T = nt$)*. The uncertainty in the problem is modelled with the following probability distributions:*

$$P(O) = \begin{array}{cc} \begin{bmatrix} 0.5 \\ 0.3 \\ 0.2 \end{bmatrix} & \begin{array}{c} e \\ w \\ s \end{array} \end{array} , \; P(S|O,T) = \begin{array}{cc} \begin{array}{cccccc} & t & & & nt & \\ e & w & s & e & w & s \end{array} & \\ \begin{bmatrix} 0.1 & 0.3 & 0.5 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0.3 & 0.4 & 0.4 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0.6 & 0.3 & 0.1 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} & \begin{array}{c} c \\ o \\ d \end{array} \end{array}$$

The oil wildcatter's problem can be represented and solved with a decision tree as shown in Figure 4.1. Basically, a decision tree is a tree-like structure in which the internal nodes represent the *variables* in the problem. Their outgoing arcs correspond with the possible *states* that a variable can take. Variables are denoted with upper-case letters while their states are denoted using lower-case letters. We can distinguish two types of internal nodes: *decision* nodes, depicted as squares, which correspond with the variables that the decision maker can control; and *chance* nodes, represented by circles, that correspond with the random variables in the problem. For example, in the decision tree representing the oil wildcatter's problem, decision nodes are T and D while the chance nodes are S and O. In case of an outgoing arc from a chance node Y labelled with the state y, it is also labelled with the conditional probability $P(y|left(Y))$ where $left(Y)$ are the states labelling the arcs from the root node to Y. Note that these probabilities might not be available in the initial definition of the problem, and hence the will be need to be computed when building the tree.

In this kind of representation, a path from the root to a leaf is a *scenario*. e.g. in Figure 4.1 , $\{nt, d, nd\}$ (upper-most path) is a scenario for the sequence of variables $\{T, S, D\}$ while $\{t, c, d, e\}$ (bottom-most path) is another scenario for the sequence $\{T, S, D, O\}$.

For evaluating a decision tree, leaves nodes are initially labelled with the utility of each scenario. Then, the evaluation is done from right to left by calculating the expected utility associated to each node as explained in Proposition 2.

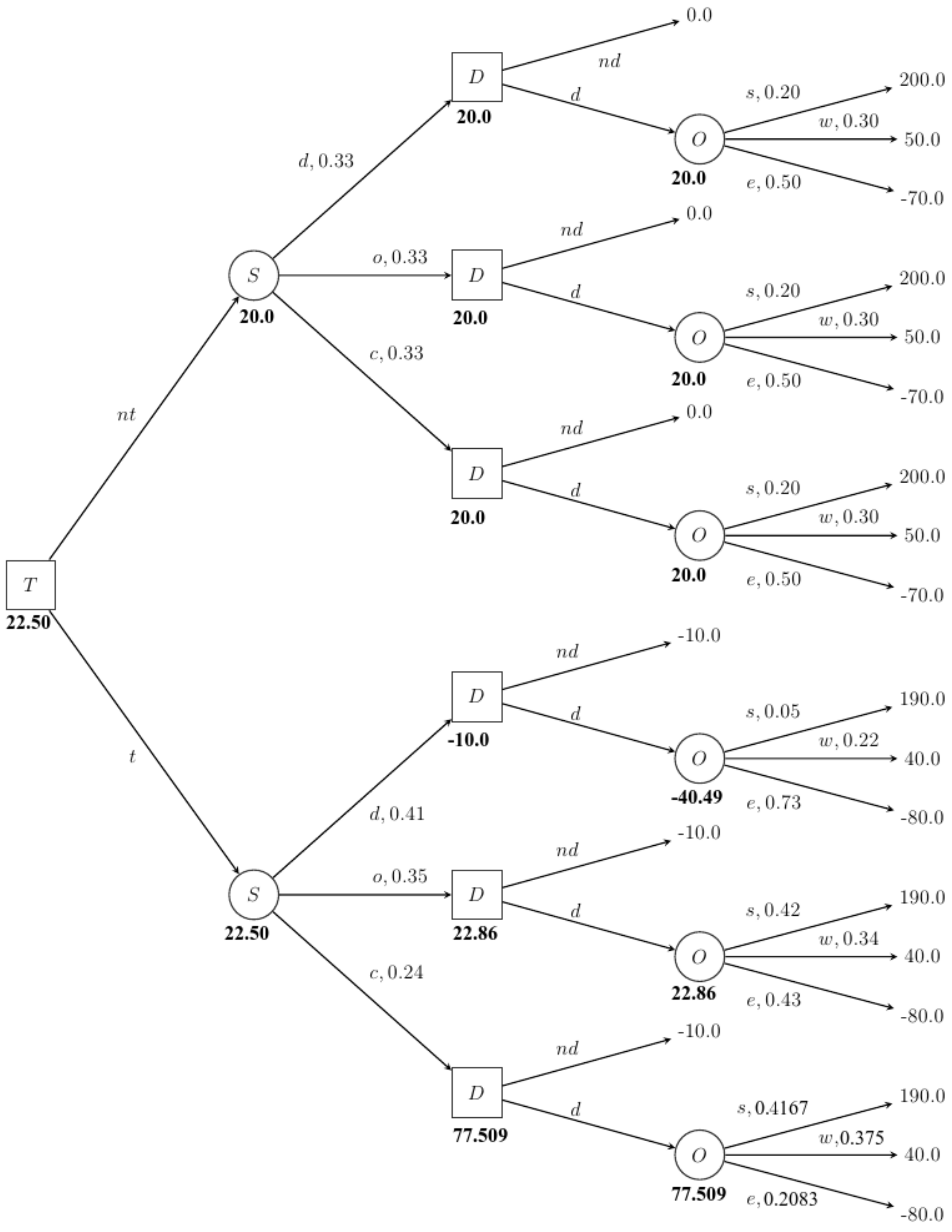


Figure 4.1: Decision tree representing the oil wildcatter's problem described in Example 10.

Proposition 2 (expected utility associated to a node in a decision tree) *Let Y be a node in a decision tree taking values in Ω_Y, let $left(Y)$ be the states labelling the arcs from the root node to Y, and let $U(y, left(Y))$ the utilities associated to a node on the right reached through the arc labelled with y. Then, the utility associated to Y and denoted $U(left(Y))$ can be computed as follows:*

- *If Y is a chance node, then:*

  $$U(left(Y)) := \sum_{y \in \Omega_Y} U(y, left(Y)) \cdot P(y|left(Y))$$

  where $P(y|left(Y))$ is the posterior probability labelling the arc associated to the state y.

- *otherwise (it is a decision node):*

  $$U(left(Y)) := \max_{y \in \Omega_Y} U(y, left(Y))$$

Note that if Y is a decision node, the optimal alternative (given $left(Y)$) is the state y of Y with a maximal $U(y, left(Y))$. In Figure 4.1, the utilities associated to each internal node are shown in bold. To illustrate these computations, let us consider the chance node O in the bottom-most branch whose expected utility is:

$$U(t, c, d) = 0.4167 \cdot 190.0 + 0.375 \cdot 40.0 + 0.2083 \cdot (-80.0) = 77.509$$

For the decision node D in the same branch, its utility is:

$$U(t, c) = \max\{-10.0,\ 77.509\} = 77.509$$

4.3 Influence diagrams

4.3.1 Definitions and notation

Influence diagrams (IDs) [87, 56] are a class of probabilistic graphical models designed to formalize sequential decision problems with uncertainty. Compared to decision trees, IDs offer a compact encoding of the independence relations between variables, which prevents an exponential growth in the problem representation as in the case of decision trees.

4.3.1.1 Syntax

Let us first define the basic notation. Like for decision trees, we use upper-case letters for variables and lower-case for their possible values (or *states*). Given a variable X, x is an element of the *domain* of X, which we denote as Ω_X. We assume that all the variables are discrete. Given a set of n variables $\mathbf{X} := \{X_1, \ldots, X_n\}$, and a multi-valued index $J \subseteq \{1, \ldots, n\}$, $\mathbf{X}_J$ is the joint variable including any X_i such that $i \in J$. Thus, $\Omega_{\mathbf{X}_J} = \times_{i \in J} \Omega_{X_i}$, where $\times$ is the Cartesian product. The elements of $\Omega_{\mathbf{X}_J}$ are called *configurations* and denoted $\mathbf{x}_J$. Given two sets of variables $\mathbf{X}_I$ and $\mathbf{X}_J$, their union is denoted $\mathbf{X}_I \cup \mathbf{X}_J$ or simply $\mathbf{X}_{I \cup J}$. Analogously, their intersection is denoted $\mathbf{X}_I \cap \mathbf{X}_J$ or $\mathbf{X}_{I \cap J}$. The notation $\mathbf{x}_I \sim \mathbf{x}_J$ is used to express *consistency*, i.e., to denote the fact that both configurations have the same values on $\mathbf{X}_{I \cap J}$.

An ID over a set of chance variables $\mathcal{U}_C$ and a set of decisions $\mathcal{U}_D$ is made of a qualitative and a quantitative part. The qualitative part is an acyclic directed graph $\mathcal{G}$ with three types of nodes. *Chance* nodes are depicted as circles and represent the chance or random variables in the problem, i.e., those in $\mathcal{U}_C$. *Decision* nodes are depicted as squares and associated to decision variables, i.e., those in $\mathcal{U}_D$. Decisions variables correspond with the actions which the decision maker can control. Utility nodes are depicted as diamonds and represent the user maker preferences. The terms node and variable are used interchangeably for both chance and decision variables. Utility nodes are not associated to variables. Yet, these nodes are jointly denoted as $\mathcal{U}_V$. Note that we consider that the decision and chance variables are discrete while the utility nodes have no states.

In the description of an ID, it is more convenient to use the terms of predecessors and successors (instead of parents and children). The set of *predecessors* of a node X is the set of all its ancestors, i.e. nodes along directed paths into X. In particular, the set of *direct predecessors* of a node X, denoted $pa(X)$, is the set of parents of a node X according to $\mathcal{G}$ while the set of *indirect predecessors* is the set formed by removing from its predecessors all its direct predecessors. In a similar way, the set of all the descendants of a node X are called successors. The set of children of X are called *direct successors* and denoted $ch(X)$. The remaining successors are called *indirect successors*. Some evaluation algorithms require IDs to be *regular* (see Proposition 3). Here we consider only such kind of IDs.

Proposition 3 (regularity) *An ID is regular if it satisfies the following conditions:*

1. *The graph $\mathcal{G}$ has no directed cycles.*
2. *The utility nodes have no direct successors.*
3. *The graph $\mathcal{G}$ contains at least a directed path connecting all decisions nodes.*

Once the qualitative part of an ID has been considered, we can describe the the quantitative one which is made of a set of *probability potentials* (PPs) and of a set of *utility potentials* (UPs). PPs represent the uncertainty whereas UPs represent the user preferences. A PP over the set of variables $\mathbf{X}_I$, denoted as $\phi(\mathbf{X}_I)$, is a map $\phi : \Omega_{\mathbf{X}_I} \to [0, 1]$. Similarly, a UP over $\mathbf{X}_K$, denoted as $\psi(\mathbf{X}_K)$, is a map $\psi : \Omega_{\mathbf{X}_K} \to \mathbb{R}$. The operator *dom* returns the variables in the domain of a potential, e.g. $dom(\phi(\mathbf{X}_I)) = \mathbf{X}_I$. For each chance node, a PP over the corresponding variable and its direct predecessors is defined, while, for each utility node, a UP potential over the parents must be assessed. In the algorithms explained in this dissertation, all the PPs are also CPDs[1]. Thus we will include the *conditioning bar* in the notation of PPs to make explicit which are the *conditioned* variables (i.e. in the head or on the left) and which are *conditioning* ones (i.e. in the tail or on the right). For example, in the PP $\phi(\mathbf{X}_I|\mathbf{X}_J)$, the conditioned variables are those in $\mathbf{X}_I$ while the conditioning ones are those in $\mathbf{X}_J$. Overall, the formal definition of an ID is the following.

[1] A *conditional probability distribution* (CPD) over two disjoint sets of variables $\mathbf{X}_I$ and $\mathbf{X}_J$, denoted $P(\mathbf{X}_I|\mathbf{X}_J)$, is a particular type of PP such that $\sum_{\mathbf{x}_I \in \Omega_{\mathbf{X}_I}} P(\mathbf{x}_I|\mathbf{x}_J) = 1$ for each $\mathbf{x}_J \in \Omega_{\mathbf{X}_J}$. For more details see Definition 6 at page 34.

Definition 15 (influence diagram) *An influence diagram (ID) is a tuple $\langle \mathcal{G}, \mathcal{U}_C, \mathcal{U}_D, \mathcal{U}_V, \Phi, \Psi \rangle$, where $\mathcal{G}$ is an acyclic directed graph over nodes $\mathcal{U}_C \cup \mathcal{U}_D \cup \mathcal{U}_V$, while $\Phi = \{\phi(X|pa(X))\}_{X \in \mathcal{U}_C}$ and $\Psi = \{\psi(pa(U))\}_{U \in \mathcal{U}_V}$ are collections of, respectively, PPs and UPs.*

Example 11 (the oil wildcatter's ID [96, 105]) *Figure 4.2 depicts the graph of an ID modelling the decision problem described in Example 10. The set of chance variables is $\mathcal{U}_C = \{S, O\}$, while the set of decisions is $\mathcal{U}_D = \{T, D\}$. The utility nodes P and C describe the profit possibly obtained from the presence of oil and the cost of the tests. The sets of potentials are $\Phi = \{\phi(O), \phi(S|O,T)\}$, and $\Psi = \{\psi(T), \psi(O,D)\}$. The numerical values of these potentials are reported here below in a table form, with the corresponding states depicted in grey.*

$$\phi(O) = \begin{array}{c} \\ \left[\begin{array}{c} 0.5 \\ 0.3 \\ 0.2 \end{array}\right] \end{array} \begin{array}{c} e \\ w \\ s \end{array}, \; \psi(T) = \left[\begin{array}{c} -10 \\ 0 \end{array}\right] \begin{array}{c} t \\ nt \end{array}, \; \psi(O,D) = \begin{array}{cc} d & nd \\ \multicolumn{2}{c}{} \end{array} \left[\begin{array}{cc} -70 & 0 \\ 50 & 0 \\ 200 & 0 \end{array}\right] \begin{array}{c} e \\ w \\ s \end{array}$$

$$\phi(S|O,T) = \begin{array}{cccccc|c} \multicolumn{3}{c}{t} & \multicolumn{3}{c}{nt} & \\ e & w & s & e & w & s & \\ 0.1 & 0.3 & 0.5 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & c \\ 0.3 & 0.4 & 0.4 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & o \\ 0.6 & 0.3 & 0.1 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & d \end{array}$$

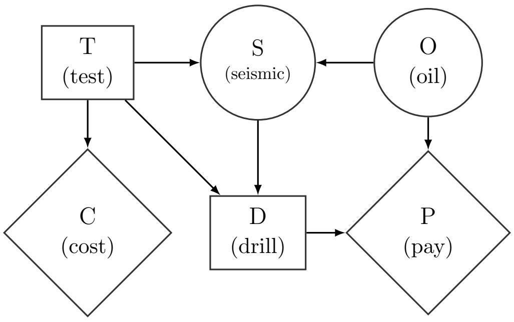


Figure 4.2: Graph of an ID modelling the oil wildcatter's decision problem

4.3.1.2 Semantics

Because of the acyclicity of $\mathcal{G}$, the path connecting all decision nodes defines a topological ordering[2] over the decision variables. Without lack of generality, the indexes of the decision nodes, say $\mathcal{U}_D := \{D_1, \ldots, D_n\}$, can be assumed to reflect such order, i.e., $D_1 \prec \ldots \prec D_n$, with the symbol $\prec$ denoting topological precedence. We partition the chance variables $\mathcal{U}_C$ in $n + 1$ disjoint sets $\{\mathcal{I}_i\}_{i=0}^n$. For each $i = 0, \ldots, n - 1$, $\mathcal{I}_i$ includes the chance nodes directly preceding D_{i+1} but not preceding D_i. If a chance variable is a direct predecessor of more than a decision node, it belongs to the set associated to the decision variable with the smallest index. The remaining chance variables, i.e., those not having decision nodes among their direct successors, belong to $\mathcal{I}_n$. This forms a partition of $\mathcal{U}_C$, i.e., $\cup_{i=0}^n \mathcal{I}_i = \mathcal{U}_C$ and $\mathcal{I}_i \cap \mathcal{I}_j = \emptyset$ for each $i, j = 0, 1, \ldots, n$, $i \neq j$. Overall, regular IDs are characterized by the following partial order of the variables:

$$\mathcal{I}_0 \prec D_1 \prec \mathcal{I}_1 \prec \cdots \prec D_n \prec \mathcal{I}_n$$

Such order reflects a temporal interpretation: the chance variables in $\mathcal{I}_i$ are observed by the decision maker before decision D_{i+1} is taken. Thus $\mathcal{I}_0$ is the set of chance nodes observed before any decision is taken while $\mathcal{I}_n$ are observed after the last decision is taken (or never observed). The ordering over $\mathcal{U}_D$ reflects the order in which the different decisions are taken. For example, in the oil wildcatter's ID, the partial order is $T \prec \{S\} \prec D \prec \{O\}$. This implies that, when deciding whether to drill or not (decision D), the decision maker knows if the test has been done and its result (variables T and S). Note that an indirect predecessor of a decision D_i might belong to $\mathcal{I}_n$, instead of $\mathcal{I}_j$ with $j < i$. This is the case of the ID shown in Figure 4.3, where chance node C is an indirect predecessor of D_2 but it belongs to $\mathcal{I}_2$.

The chance variables are observed at the time of decision, but not (necessarily) at the time of analysis. We will assume that chance variables are always unobserved from the point of view of the analyst.

[2] A *topological ordering* of a directed graph is a linear ordering of its nodes such that for every directed arc (X_i, X_j) from, X_i comes before X_j in the ordering.

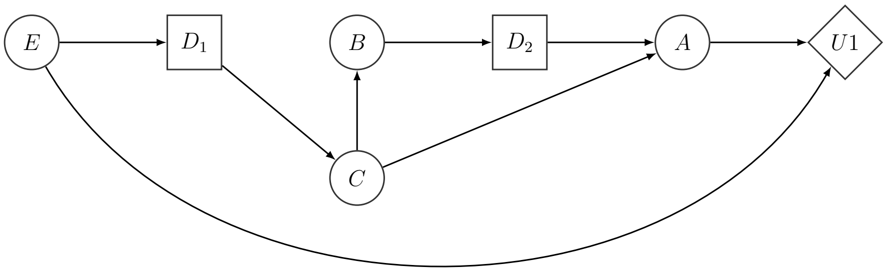


Figure 4.3: Example of an ID where a predecessor of a decision belongs to $\mathcal{I}_n$. The partial order is $\{E\} \prec D_1 \prec \{B\} \prec D_2 \prec \{A, C\}$.

The arcs in $\mathcal{G}$ have different meaning depending on the target. *Conditional arcs* are those into chance and utility nodes and represent probabilistic and functional dependence respectively. Thus, direct predecessors of utility and chance nodes are called *conditional predecessors*. By contrast, arcs into decision nodes are called *informational arcs* and imply a time precedence. Thus direct predecessors of decision nodes are called *informational predecessors*.

When representing a decision problem with an ID, the *non-forgetting assumption* [102, 56] is required (perfect recall): previous decisions and observations are known at each decision. For example, in the ID shown in Figure 4.3, variables E and D_1 are known when deciding about D_2 (B is also known since it is its direct predecessor). This perfect recall has the following implication:

Proposition 4 (non-forgetting assumption [102, 56]) *If decision node D_i precedes decision node D_j in a regular ID, then D_i and all of its informational predecessors should be informational predecessors of D_j.*

Informational arcs that satisfy Proposition 4 are called *non-forgetting arcs* and they are usually assumed implicit to reduce complexity of the graphical display. When all the non forgetting arcs are present, it holds that $pa(D_i) = \mathcal{I}_0 \cup \{D_1\} \cup \cdots \cup \mathcal{I}_{i-1}$. Figure 4.4 shows this ID with the non-forgetting arcs depicted as dashed arcs.

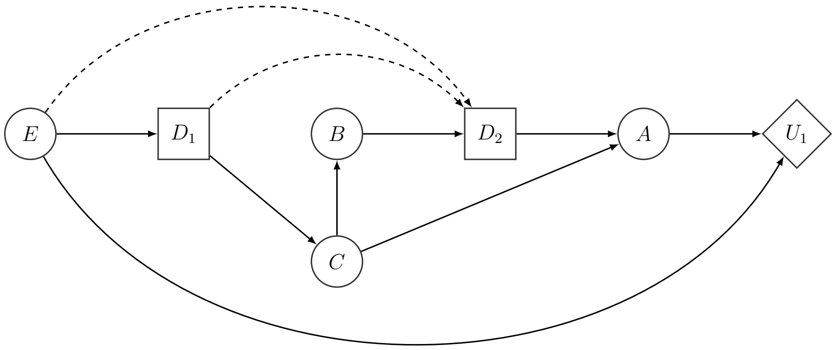


Figure 4.4: ID in Figure 4.3 with non-forgetting arcs shown explicitly.

4.3.2 Evaluation

As we have seen, an ID offers a compact representation of a decision problem under uncertainty. Its graphical display can be useful to understand and to reason about the problem. Furthermore, we will be interested in inferring new knowledge for helping the decision maker. In particular, we will have to identify the best alternatives for each decision given previous observations and decisions (*optimal policy*). This process, which is called *evaluation*, is performed according to the MEU principle (see Definition 14), which asserts that if we have to choose between a set of possible actions, we should choose the one with the maximum expected utility. Herein we formalize these concepts, but we should first review the basics operations with potentials required during the evaluation.

4.3.2.1 Operations with potentials

To evaluate an ID we need to define some simple operations, namely combination, division, marginalization and restriction.

The *combination* operation represents aggregation of knowledge. Given two potentials (no matter whether UPs or PPs) ϕ and ψ, their combination gives as a result a new potential whose domain is $dom(\phi) \cup dom(\psi)$. In the evaluation of an ID, depending on the potentials involved, we can distinguish two types of combination: *multiplication* and *addition*. Two PPs are combined using the multiplication while two UPs are combined with the addition instead (we assume an additive model). When combining a PP with a UP, we will proceed as in the case of two PPs. Thus, the combination can be defined as follows:

Definition 16 (combination) *The combination (multiplication) $\phi \cdot \psi$ of a PP $\phi(\mathbf{X}_I|\mathbf{X}_J)$ with a UP $\psi(\mathbf{X}_K)$ is a UP over $\mathbf{X}_L := \mathbf{X}_{I\cup J\cup K}$ defined by element-wise products, i.e.,*

$$(\phi \cdot \psi)(\mathbf{x}_{I\cup J\cup K}) := \phi(\mathbf{x}_I|\mathbf{x}_J) \cdot \psi(\mathbf{x}_K), \tag{4.2}$$

for each $\mathbf{x}_{I\cup J\cup K} \in \Omega_{\mathbf{X}_{I\cup J\cup K}}$, with $\mathbf{x}_I, \mathbf{x}_J, \mathbf{x}_K \sim \mathbf{x}_{I\cup J\cup K}$. Finally, the combination $\phi \cdot \phi'$ of two PPs, say $\phi(\mathbf{X}_I|\mathbf{X}_J)$ and $\phi'(\mathbf{X}_K|\mathbf{X}_L)$, is a PP over $\mathbf{X}_{I\cup K}$ given $\mathbf{X}_{(J\cup L)\setminus(I\cup K)}$ defined by element-wise products, i.e.,

$$(\phi \cdot \phi')(\mathbf{x}_{I\cup K}|\mathbf{x}_{(J\cup L)\setminus(I\cup K)}) := \phi(\mathbf{x}_I|\mathbf{x}_J) \cdot \phi(\mathbf{x}_K|\mathbf{x}_L) \tag{4.3}$$

for each $\mathbf{x}_{I\cup K} \in \Omega_{\mathbf{X}_{I\cup K}}$ and $\mathbf{x}_{(J\cup L)\setminus(I\cup K)} \in \Omega_{\mathbf{X}_{(J\cup L)\setminus(I\cup K)}}$, with $\mathbf{x}_I, \mathbf{x}_J, \mathbf{x}_K, \mathbf{x}_L \sim \mathbf{x}_{I\cup K}, \mathbf{x}_{(J\cup L)\setminus(I\cup K)}$. The combination (addition) $\psi + \psi'$ of two UPs, say $\psi(\mathbf{X}_I)$ and $\psi'(\mathbf{X}_J)$, is a UP over $\mathbf{X}_{I\cup J}$ obtained by element-wise sums, i.e.,

$$(\psi + \psi')(\mathbf{x}_{I\cup J}) := \psi(\mathbf{x}_I) + \psi'(\mathbf{x}_J) \tag{4.4}$$

for each $\mathbf{x}_{I\cup J} \in \Omega_{\mathbf{X}_{I\cup J}}$, with $\mathbf{x}_I, \mathbf{x}_J \sim \mathbf{x}_{I\cup J}$.

The multiplication of two potentials is denoted by a dot, which is often hidden. The symbol Π is used for denoting the multiplication of a collection of potentials, e.g. $\prod_{\phi\in\Phi} \phi$. The *commutative law* applies to multiplication i.e. $\phi_1\phi_2 = \phi_2\phi_1$, and so does the *associative law* i.e. $(\phi_1\phi_2)\phi_3 = \phi_1(\phi_2\phi_3)$. These laws also apply to addition, i.e. $\psi_1 + \psi_2 = \psi_2 + \psi_1$ and $(\psi_1 + \psi_2) + \psi_3 = \psi_1 + (\psi_2 + \psi_3)$. The distributive law applies to addition as well, i.e $\phi \cdot (\psi_1 + \psi_2) = \phi \cdot \psi_1 + \phi \cdot \psi_2$. The symbol Σ is used for denoting the addition of a collection of UPs, e.g. $\sum_{\psi\in\Psi} \psi$.

Example 12 (combination) *Let us consider the potentials depicted in Example 11 associated to the oil wildcatter's ID. Then, three examples of the combination operation with these potentials are shown here below:*

$$\psi_1(T,O,D) = \psi(T) + \psi(O,D) = \begin{array}{c} \begin{array}{cccc} t & & nt & \\ d & nd & d & nd \end{array} \\ \left[\begin{array}{rrrr} -80.0 & -10.0 & -70.0 & 0.0 \\ 40.0 & -10.0 & 50.0 & 0.0 \\ 190.0 & -10.0 & 200 & 0.0 \end{array}\right] \begin{array}{c} e \\ w \\ s \end{array} \end{array} \tag{4.5}$$

$$\phi_1(S,O|T) = \phi(O) \cdot \phi(S|O,T) = \begin{array}{c} \begin{array}{cccccc} & t & & & nt & \\ e & w & s & e & w & s \end{array} \\ \left[\begin{array}{rrrccc} 0.05 & 0.09 & 0.1 & \frac{1}{6} & \frac{1}{10} & \frac{1}{15} \\ 0.15 & 0.12 & 0.08 & \frac{1}{6} & \frac{1}{10} & \frac{1}{15} \\ 0.3 & 0.09 & 0.02 & \frac{1}{6} & \frac{1}{10} & \frac{1}{15} \end{array}\right] \begin{array}{c} c \\ o \\ d \end{array} \end{array} \tag{4.6}$$

$$\psi_2(S,O,T,D) = \phi_1(S,O|T) \cdot \psi(O,D) = \begin{array}{c} \begin{array}{cccc} t & & nt & \\ d & nd & d & nd \end{array} \\ \left[\begin{array}{rrrr} -3.5 & 0.0 & -11.667 & 0.0 \\ 4.5 & 0.0 & 5.0 & 0.0 \\ 20.0 & 0.0 & 13.333 & 0.0 \\ -10.5 & 0.0 & -11.667 & 0.0 \\ 6.0 & 0.0 & 5.0 & 0.0 \\ 16.0 & 0.0 & 13.333 & 0.0 \\ -21.0 & 0.0 & -11.667 & 0.0 \\ 4.5 & 0.0 & 5.0 & 0.0 \\ 4.0 & 0.0 & 13.333 & 0.0 \end{array}\right] \begin{array}{cc} e & \\ w & c \\ s & \\ e & \\ w & o \\ s & \\ e & \\ w & d \\ s & \end{array} \end{array} \tag{4.7}$$

The *division* or *ratio* of two potentials can also be used during the evaluation of an ID, and it is defined as follows:

Definition 17 (division) *The division between a UP $\psi(\mathbf{X}_I)$ and a PP $\phi(\mathbf{X}_J)$ is a UP ψ/ϕ over $\mathbf{X}_{I\cup J}$ such that, for each $\mathbf{x}_{I\cup J} \in \Omega_{\mathbf{X}_{I\cup J}}$:*

$$(\psi/\phi)(\mathbf{x}_{I\cup J}) \quad := \quad \psi(\mathbf{x}_I)/\phi(\mathbf{x}_J) \tag{4.8}$$

with $\mathbf{x}_I, \mathbf{x}_J \sim \mathbf{x}_{I\cup J}$. With zero denominators, the result is set to $+\infty$ for positive numerators and $-\infty$ for negative ones. When both, numerator and denominator, are zero, the convention $\frac{0}{0} = 0$ is adopted.

The division of two PPs is analogously defined. Note that the laws previously mentioned (commutative, associative and distributive) do not apply to division.

The marginalization (projection) is an operation that removes a variable from the domain of a potential. During the evaluation of an ID, two types of marginalization (projection) can be used: *sum-marginalization* and *max-marginalization*. These operations are defined as follows:

Definition 18 (sum-marginalization) *The sum-marginalization* $\sum_Y \psi$ *of a UP* $\psi(Y, \mathbf{X}_I)$ *is a UP over* $\mathbf{X}_I$ *such that:*

$$\left(\sum_Y \psi\right)(\mathbf{x}_I) \quad := \quad \sum_{y \in \Omega_Y} \psi(y, \mathbf{x}_I) \tag{4.9}$$

$$\tag{4.10}$$

for each $\mathbf{x}_I \in \Omega_{\mathbf{X}_I}$.

The commutative law applies to sum-marginalization, i.e. $\sum_Y \sum_Z \psi = \sum_Z \sum_Y \psi$. The sum-marginalization of a PP is analogously defined. Let $\phi(\mathbf{X}_I|\mathbf{X}_J)$ be a PP, then it holds that

$$\sum_{\mathbf{X}_I} \phi(\mathbf{X}_I|\mathbf{X}_J) = 1_{\mathbf{X}_J}$$

where $1_{\mathbf{X}_J}$ is a *unity potential*, i.e. a potential defined on $\mathbf{X}_J$ assigning the value 1 to each configuration of $\Omega_{\mathbf{X}_J}$.

Definition 19 (max-marginalization) *The max-marginalization* $\max_Y \psi$ *of a UP* $\psi(Y, \mathbf{X}_I)$ *is a UP over* $\mathbf{X}_I$ *such that:*

$$\left(\max_Y \psi\right)(\mathbf{x}_I) \quad := \quad \max_{y \in \Omega_Y} \psi(y, \mathbf{x}_I) \tag{4.11}$$

for each $\mathbf{x}_I \in \Omega_{\mathbf{X}_I}$.

The max-marginalization of a PP is analogously defined. The commutative law also applies to max-marginalization, i.e. $\max_Y \max_Z \psi = \max_Z \max_Y \psi$. However, sum-marginalization and max-marginalization are not in general commutative, i.e. $\sum_Y \max_Z \psi \neq \max_Z \sum_Y \psi$ (see Proof 1).

Example 13 (marginalization and division) *Let us consider the potentials $\psi_2(S,O,T,D)$ and $\phi_1(S,O|T)$ obtained in Example 12 (Equations* (4.7) *and* (4.6) *respectively). Then, the result of sum-marginalizing out the variable O from these potentials is:*

$$\psi_3(S,T,D) = \sum_O \psi_2(S,O,T,D) = \begin{array}{cc} \begin{array}{cccc} t & & nt & \\ d & nd & d & nd \end{array} & \\ \begin{bmatrix} 21.0 & 0.0 & 6.667 & 0.0 \\ 11.5 & 0.0 & 6.667 & 0.0 \\ -12.5 & 0.0 & 6.667 & 0.0 \end{bmatrix} & \begin{array}{c} c \\ o \\ d \end{array} \end{array} \qquad (4.12)$$

$$\phi_2(S|T) = \sum_O \phi_1(S,O|T) = \begin{array}{cc} \begin{array}{cc} t & nt \end{array} & \\ \begin{bmatrix} 0.24 & \frac{1}{3} \\ 0.35 & \frac{1}{3} \\ 0.41 & \frac{1}{3} \end{bmatrix} & \begin{array}{c} c \\ o \\ d \end{array} \end{array} \qquad (4.13)$$

The division of the UP and the PP previously obtained gives as a result the following UP:

$$\psi_4(S,T,D) = \frac{\psi_3(S,T,D)}{\phi_2(S|T)} = \begin{array}{cc} \begin{array}{cccc} t & & nt & \\ d & nd & d & nd \end{array} & \\ \begin{bmatrix} 87.5 & 0.0 & 20.0 & 0.0 \\ 32.857 & 0.0 & 20.0 & 0.0 \\ -30.488 & 0.0 & 20.0 & 0.0 \end{bmatrix} & \begin{array}{c} c \\ o \\ d \end{array} \end{array} \qquad (4.14)$$

Finally, the max-marginalization of decision D from $\psi_4(S,T,D)$ is:

$$\psi_5(S,T) = \max_D \psi_4(S,T,D) = \begin{array}{cc} \begin{array}{cc} t & nt \end{array} & \\ \begin{bmatrix} 87.5 & 20.0 \\ 32.857 & 20.0 \\ 0.0 & 20.0 \end{bmatrix} & \begin{array}{c} c \\ o \\ d \end{array} \end{array} \qquad (4.15)$$

Proof 1 *Let us consider the potential $\phi_1(S,O|T)$ obtained in Example 12 (Equations* (4.6)*). If variable O is first sum-marginalized and then variable S is max-marginalized out, we obtain:*

$$\max_S \sum_O \phi_1(S,O|T) = \begin{bmatrix} 0.41 \\ \frac{1}{3} \end{bmatrix} \begin{matrix} t \\ nt \end{matrix} \tag{4.16}$$

if the order of the operations is changed, we obtain:

$$\sum_O \max_S \phi_1(S,O|T) = \begin{bmatrix} 0.52 \\ \frac{1}{3} \end{bmatrix} \begin{matrix} t \\ nt \end{matrix} \tag{4.17}$$

the potentials obtained in (4.16) *and* (4.17) *are different, and hence we can state that the sum-marginalization and max-marginalization are not in general commutative.*

Another auxiliary usually required for the evaluation is the *restriction*, which is the instantiation of one the variables in the potential. This operation can be defined as follows.

Definition 20 (restriction) *The restriction $\psi^{R(Y=y)}$ of a UP $\psi(Y,\mathbf{X}_I)$ is a UP over $\mathbf{X}_I$ such that:*

$$\left(\psi^{R(Y=y)}\right)(\mathbf{x}_I) \;\; := \;\; \psi(Y=y,\mathbf{x}_I) \tag{4.18}$$

for each $\mathbf{x}_I \in \Omega_{\mathbf{X}_I}$.

The restriction of a PP is analogously defined. The commutative law also applies to restriction, i.e. $\left(\psi^{R(Y=y)}\right)^{R(Z=z)} = \left(\psi^{R(Z=z)}\right)^{R(Y=y)} = \psi^{R(Y=y,Z=z)}$.

Example 14 (restriction) *Let us consider the potential $\psi_4(S,T,D)$ obtained in Example 13 (Equation* (4.13) *). Then, the result of restricting this UP to $D=d$ is:*

$$\psi_6(S,T) = \psi_4^{R(D=d)}(S,T,D) = \begin{array}{cc} \begin{array}{cc} t & nt \end{array} & \\ \begin{bmatrix} 87.5 & 20.0 \\ 32.857 & 20.0 \\ -30.488 & 20.0 \end{bmatrix} & \begin{array}{c} c \\ o \\ d \end{array} \end{array} \tag{4.19}$$

For the scope of this dissertation, we also need to define the restriction to a set of states as follows.

Definition 21 (restriction to a set of states) *Let Y be a variable and let $S_Y \subseteq \Omega_Y$ be a subset of its states with more than one element. Then the restriction $\psi^{R(Y,S_Y)}$ of a UP $\psi(Y,\mathbf{X}_I)$ is a UP over $\{Y\} \cup \mathbf{X}_I$ such that:*

$$\left(\psi^{R(Y,S_Y)}\right)(y,\mathbf{x}_I) \;\; := \;\; \psi(y,\mathbf{x}_I) \tag{4.20}$$

for each $(y,\mathbf{x}_I) \in \Omega_Y \times \Omega_{\mathbf{X}_I}$.

Again, the restriction of a PP is analogously defined and the commutative law applies to the restriction. Note that, in previous definition, we consider that the set S_Y contains more than one element. Otherwise, the basic restriction operation (i.e., to one state) given in Definition 20 will be considered.

Example 15 (restriction to a set of states) *Let us consider the potential $\psi_4(S,T,D)$ obtained in Example 13 (Equation (4.13)). Then, the result of restricting this UP to the set $S_S = \{c,o\}$ is:*

$$\psi_7(S,T,D) = \psi_4^{R(S,\{c,o\})}(S,T,D) = \begin{array}{cc} \begin{array}{cccc} \multicolumn{2}{c}{t} & \multicolumn{2}{c}{nt} \\ d & nd & d & nd \end{array} & \\ \begin{bmatrix} 87.5 & 0.0 & 20.0 & 0.0 \\ 32.857 & 0.0 & 20.0 & 0.0 \end{bmatrix} & \begin{array}{c} c \\ o \end{array} \end{array} \tag{4.21}$$

4.3.2.2 Optimal policies and strategies

An ID encodes a joint probability distribution over all the chance variables given all the decisions. The set of all PPs specifies a multiplicative factorization of the joint probability distribution of $\mathcal{U}_C$ given $\mathcal{U}_D$ as represented by the *chain rule*.

Theorem 1 (the chain rule for IDs) *Consider an ID as in Definition 15 which is also regular and satisfies the non-forgetting assumption. Let $\mathcal{U}_C$ and $\mathcal{U}_D$ be the set of chance and decision variables and $pa(X)$ the conditional predecessors of a node X, then the probability distribution representing the uncertainty is:*

$$P(\mathcal{U}_C|\mathcal{U}_D) := \prod_{X \in \mathcal{U}_C} \phi(X|pa(X)) \tag{4.22}$$

Due to the chain rule, we can see that an ID is a compact representation of a joint expected utility function:

$$EU(\mathcal{U}_C,\mathcal{U}_D) := \prod_{X \in \mathcal{U}_C} \phi(X|pa(X)) \sum_{U \in \mathcal{U}_V} \psi(pa(U)) \tag{4.23}$$

A *policy* for a decision variable D_i is a mapping $\delta_{D_i} : \Omega_{pa(D_i)} \to \Omega_{D_i}$ associating a state of D_i (i.e., a decision) to its past observations and decisions. A *strategy* Δ is a collection of policies, one for each decision variable, i.e., $\Delta := \{\delta_{D_1}, \delta_{D_2}, \ldots, \delta_{D_n}\}$. Evaluating IDs consists in the identification of an optimal strategy $\widehat{\Delta}$, which maximizes the expected value of the sum of the UPs. The (optimal) policies of an optimal strategy and the maximum expected utility are defined as follows.

Definition 22 (optimal policy and maximum expected utility [64]) *Consider an ID as in Definition 15 which is also regular and satisfies the non-forgetting assumption. Let the temporal order be described as $\mathcal{I}_0 \prec D_1 \prec \mathcal{I}_1 \prec \cdots \prec D_n \prec \mathcal{I}_n$. For each $i = 1, \ldots, n$, the optimal policy for decision D_i is*

$$
\begin{aligned}
&\widehat{\delta}_{D_i}(\mathcal{I}_0, D_1, \dots, \mathcal{I}_{i-1}) := \\
&\arg\max_{D_i} \sum_{\mathcal{I}_i} \max_{D_{i+1}} \cdots \max_{D_n} \sum_{\mathcal{I}_n} \prod_{X \in \mathcal{U}_C} \phi(X|pa(X)) \sum_{U \in \mathcal{U}_V} \psi(pa(U))
\end{aligned}
\tag{4.24}
$$

The expected utility for D_i (and acting optimally in the future) is:

$$
\begin{aligned}
EU_{D_i}(\mathcal{I}_0, D_1, \dots, \mathcal{I}_{i-1}) &:= \frac{1}{\phi(\mathcal{I}_0, \dots, \mathcal{I}_{i-1}|D_1, \dots, D_{i-1})} \\
\max_{D_i} \sum_{\mathcal{I}_i} \max_{D_{i+1}} \cdots \max_{D_n} \sum_{\mathcal{I}_n} &\prod_{X \in \mathcal{U}_C} \phi(X|pa(X)) \left(\sum_{U \in \mathcal{U}_V} \psi(pa(U)) \right)
\end{aligned}
\tag{4.25}
$$

and the maximum expected utility is

$$
\text{MEU} := \sum_{\mathcal{I}_0} \max_{D_1} \cdots \max_{D_n} \sum_{\mathcal{I}_n} \prod_{X \in \mathcal{U}_C} \phi(X|pa(X)) \sum_{U \in \mathcal{U}_V} \psi(pa(U)), \tag{4.26}
$$

Equation (4.24) returns the value of D_i maximizing the (unnormalized) expected value of the sum of the UPs. At the moment of that decision, all the previous decisions have been already taken and all the chance variables in the past observed. The maximization is indeed achieved with respect to D_i and the subsequent decisions, with the expectation computed with respect to the uncertainty about the chance variables in the future of D_i. The expected utility for D_i in Equation (4.25) can be regarded as the expected utility given the past variables and assuming that the optimal decisions are taken in the future. Similarly, the MEU in Equation (4.26) can be regarded as the expected value of the sum of the utilities when the decision maker takes his/her decisions on the basis of the optimal policies in Equation (4.24). In other words, an ID can be solved by removing from the joint expected utility function (Equation (4.23)) all the variables in reverse order of information precedence given by $\prec$. In particular, chance variables are removed using sum-marginalization (Definition 18) whereas decisions are removed using max-marginalization (Definition 19). Overall, an example of the evaluation of an ID using formulas in Definition 22 is shown here below.

Example 16 (optimal oil wildcatter's policy) *Let us consider the oil wildcatter's ID (Example 11). The joint probability $\phi(S, O|T)$ was given in Equation* (4.6)*. The joint expected utility $EU(O, S, T, D)$ can be computed as follows:*

$$EU(O,S,T,D) = \phi(O) \cdot \phi(S|O,T) \cdot \Big(\psi(T) + \psi(O,D)\Big) =$$

$$= \begin{array}{cccc|cc} \multicolumn{2}{c}{t} & \multicolumn{2}{c|}{nt} & & \\ d & nd & d & nd & & \\ \hline -4.0 & -0.5 & -11.667 & 0.0 & c & \\ -12.0 & -1.5 & -11.667 & 0.0 & o & e \\ -24.0 & -3.0 & -11.667 & 0.0 & d & \\ 3.6 & -0.9 & 5.0 & 0.0 & c & \\ 4.8 & -1.2 & 5.0 & 0.0 & o & w \\ 3.6 & -0.9 & 5.0 & 0.0 & d & \\ 19.0 & -1.0 & 13.333 & 0.0 & c & \\ 15.2 & -0.8 & 13.333 & 0.0 & o & s \\ 3.8 & -0.2 & 13.333 & 0.0 & d & \end{array} \tag{4.27}$$

For computing the optimal policy for decision D, we apply Equation (4.24)*. That is, we have to remove from $EU(O, S, T, D)$ all the variables in the future of D, and restrict to the states in Ω_T that maximize the expected utility :*

$$\widehat{\delta}_D(S,T) = \arg\max_D \sum_O EU(O,S,T,D) =$$

$$= \arg\max_D \begin{array}{cccc|c} \multicolumn{2}{c}{t} & \multicolumn{2}{c|}{nt} & \\ d & nd & d & nd & \\ \hline 18.6 & -2.4 & 6.667 & 0.0 & c \\ 8.0 & -3.5 & 6.667 & 0.0 & o \\ -16.6 & -4.1 & 6.667 & 0.0 & d \end{array} = \begin{array}{cc|c} t & nt & \\ \hline d & d & c \\ d & d & o \\ nd & d & d \end{array} \tag{4.28}$$

and the expected utility from following the optimal policy $\widehat{\delta}_D(S, T)$ can be calculated using Equation (4.25)*:*

$$
\begin{aligned}
EU_D(S,T) = {} & \tfrac{1}{\max_D \sum_O \phi(S,O|T)} \cdot \max_D \sum_O EU(O,S,T,D) = \\
= {} & \frac{1}{\begin{array}{cccc} t & nt & \\ 0.24 & \frac{1}{3} & c \\ 0.35 & \frac{1}{3} & o \\ 0.41 & \frac{1}{3} & d \end{array}} \cdot \begin{array}{ccc} t & nt & \\ 18.6 & 6.667 & c \\ 8.0 & 6.667 & o \\ -4.1 & 6.667 & d \end{array} = \begin{array}{ccc} t & nt & \\ 77.5 & 20.0 & c \\ 22.857 & 20.0 & o \\ -10.0 & 20.0 & d \end{array}
\end{aligned}
\tag{4.29}
$$

We proceed similarly for computing the optimal policy for T:

$$
\begin{aligned}
\widehat{\delta}_T = {} & \arg\max_T \sum_S \max_D \sum_O EU(O,S,T,D) = \\
= {} & \arg\max_T \begin{array}{cc} \left[\begin{matrix} 22.5 \\ 20.0 \end{matrix}\right] & \begin{matrix} t \\ nt \end{matrix} \end{array} = [\ t\ ]
\end{aligned}
\tag{4.30}
$$

Note that $\widehat{\delta}_T$ has not arguments since there is not any variable on its past (T is the first decision and $\mathcal{I}_0 = \emptyset$). Thus, the best alternative for T is always t, i.e. to test. The expected utility from following the optimal policy $\widehat{\delta}_T$ is:

$$
\begin{aligned}
EU_T = {} & \tfrac{1}{\max_T \sum_S \max_D \sum_O \phi(S,O|T)} \cdot \max_T \sum_S \max_D \sum_O EU(O,S,T,D) = \\
= {} & \max_T \sum_S \max_D \sum_O EU(O,S,T,D) = 22.5
\end{aligned}
\tag{4.31}
$$

Another consequence of not having any variable on the past of T is that the EU_T is equal to the MEU: it holds that $\max_T \sum_S \max_D \sum_O \phi(S,O|T) = 1$.

Although IDs avoid the exponential growth in the problem representation, the evaluation using formulas in Definition 22 might be intractable when the problem contains a high number of variables: the computation of $\phi(\mathcal{U}_C|\mathcal{U}_D)$ and $EU(\mathcal{U}_C,\mathcal{U}_D)$ might be too costly as the sizes of these potentials are exponential in the number of variables in the ID. Thus we must look for methods that allow us to deal with smaller potentials. In Section 4.5 we will review some of the evaluation algorithms from the literature that deal with this problem.

4.4 Independence assumptions in IDs

The power of IDs lies in the representation of independence relations which can be exploited in order to avoid the exponential growth in the problem representation and to provide computational savings during the evaluation process: like in BNs, each chance variable in an ID is conditional independent from its non-successors given its direct predecessors. This property is exploited by IDs to reduce the size of the joint probability distribution $P(\mathcal{U}_C|\mathcal{U}_D)$ since it can be expressed as the product of all the PPs (see Theorem 1). In this section we define such relations, explain how they can be detected using the d-separation criterion and how these concepts can be used to reduce the complexity of an ID prior the evaluation (minimalization).

4.4.1 D-separation in IDs

As explained in Section 3.2.2, d-separation is a criterion for detecting independences in probabilistic graphical models (and therefore in IDs). This criterion can be addapted for IDs [103, 64] as follows.

Definition 23 (d-separation for IDs) *Let* $\mathbf{X}$*,* $\mathbf{Y}$ *and* $\mathbf{Z}$ *be pairwise disjoint sets of nodes in a DAG. Let us consider all undirected paths between them that neither contain informational arcs nor utility nodes. Then* $\mathbf{X}$ *and* $\mathbf{Y}$ *are d-separated given* $\mathbf{Z}$ *if and only if along every of these paths there is an intermediate node* A *such that either:*

(i) the connection is serial or diverging and A *belongs to* $\mathbf{Z}$
or

(ii) the connection is converging and neither A nor any of its descendants are in $\mathbf{Z}$.

When $\mathbf{X}$ *and* $\mathbf{Y}$ *are d-separated given* $\mathbf{Z}$, *we write* $I(\mathbf{X} \perp \mathbf{Y} | \mathbf{Z})$ *to indicate that this CI can be derived* [3]. *Otherwise,* $\mathbf{X}$ *and* $\mathbf{Y}$ *are conditionally dependent given* $\mathbf{Z}$.

In previous definition, $\mathbf{Z}$ can be empty. In that case, $\mathbf{X}$ and $\mathbf{Y}$ are d-separated if and only if every path contains at least one converging connection. In that case, we can deduce that X and Y are independent, i.e. $I(X \perp Y) = I(X \perp Y | \emptyset)$.

Note that the d-separation criterion for IDs is almost the same that the one for BNs (see Definition 2 at page 27). The only difference is that informational arcs and utility nodes are ignored. For example, in the ID shown in Figure 4.5, variables C and T are d-separated given B. Also, A is d-separated from D_2, i.e. $I(A \perp D_2)$ since the only possible path is $\{A, B, C, D_2\}$ which contains the converging connection $\{B, C, D_2\}$. Note that the paths through T are not considered because (T, D_2) is an informational arc.

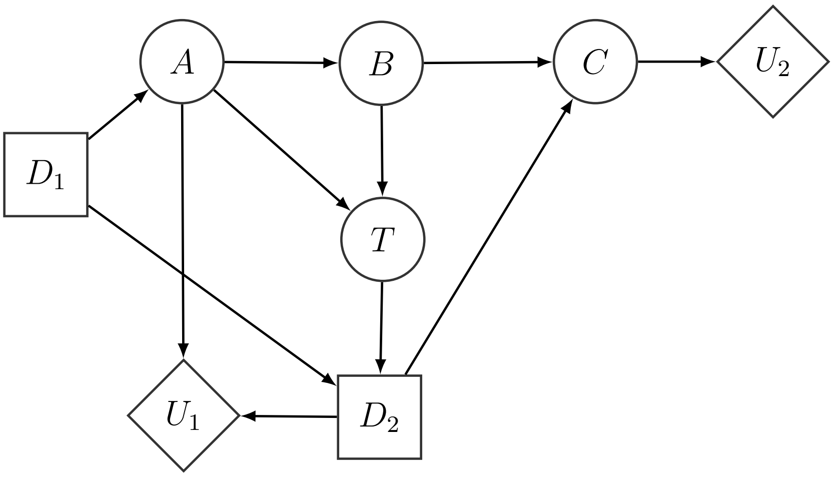


Figure 4.5: Example of an ID obtained from [64, page 226].

[3] For more details about marginal and conditional independence see Section 3.3.3.

For decision variables, the question is whether an action will have impact in the certainty for chance variables. The effects of a decision cannot “go back in time”, i.e. any decision is d-separated from its predecessors. In relation to this idea, Jensen and Nielsen [64, page 226] obtained the following conclusion:

Proposition 5 *Let $A \in \mathcal{I}_i$ and let D_j be a decision variable with $i < j$. Then,*

(i) A and D_j are d-separated and hence

$$P(A|D_j) = P(A)$$

(ii) Let W be any set of variables prior to D_j in the temporal order. Then, A and D_j are d-separated given W and hence

$$P(A|D_j, W) = P(A|W)$$

4.4.2 Minimalization of an ID

Before the evaluation, an ID can be simplified (*minimalization*) by removing *redundant informational arcs* and *barren nodes*. These two transformations map the original ID into an equivalent and less complex one (two IDs are equivalent if they have the same MEU and the same optimal policies).

As stated in Definition 22, the optimal policy $\widehat{\delta}_{D_i}$ is defined over all past observations and decisions, i.e. $\mathcal{I}_0, D_1, \ldots, \mathcal{I}_{i-1}$. However, some of the informational predecessors of D_i might not be requisite for computing $\widehat{\delta}_{D_i}$. Informally speaking, an observation (or decision) is non-requisite for a decision [48, 86], if the outcome of the observation does not impact the choice of the decision option. A more formal definition is:

Definition 24 (non-requisite informational predecessors) *Let X be an informational predecessor of a decision D_i, then X is non-requisite if X is d-separated from all the utility nodes that are (direct or indirect) successors of D_i given the rest of informational predecessors and D_i.*

Informational arcs from non-requisite observations are called *redundant arcs* and can be removed in reverse order. That is, we first identify and remove redundant arcs into D_n, then those into D_{n-1}, etc. Note that after the removal of redundant arcs, some nodes might become barren, which are defined as follows.

Definition 25 (barren node [102]) *A chance or decision node is a barren node if it is sink, in other words, it has no successors or only barren successors.*

For evaluating an ID, barren nodes have no impact on the decisions and therefore they can be directly removed without processing. After the minimalization, the direct predecessors of a decision compose its *relevant past*.

Example 17 (redundant arcs removal) *Let us consider the ID shown in Figure 4.6 including all the non-forgetting arcs. Its partial order is* $\{B\} \prec D_1 \prec \{E, F\} \prec D_2 \prec \{\} \prec D_3 \prec \{G\} \prec D_4 \prec \{A, C, D, H, I, J, K, L\}$.

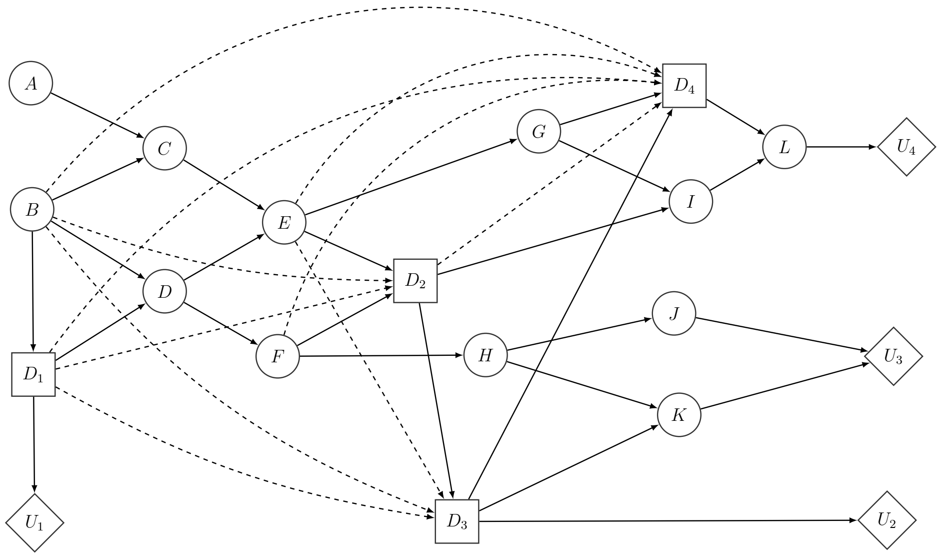


Figure 4.6: Example of an ID obtained from [64, page 141] including all the non-forgetting arcs.

Let us first consider D_4 *whose set of informational predecessors* $pa(D_4)$ *is* $\{B, D_1, E, F, D_2, D_3, G\}$. *Thus we should check if each of these variables are d-separated from* U_4 *given the rest of them (*U_4 *is the only utility node which is a successor of*

D_4). Doing that we get that G and D_2 compose the relevant past of D_4. Note that the paths $G - I - L - U_4$ and $D_2 - I - L - U_4$ are not blocked. The rest of informational predecessors are non-requisite and therefore their corresponding informational arcs into D_4 are redundant (they can be removed). If we proceed in the same way for decisions D_3, D_2, D_1 we obtain the ID depicted in Figure 4.7.

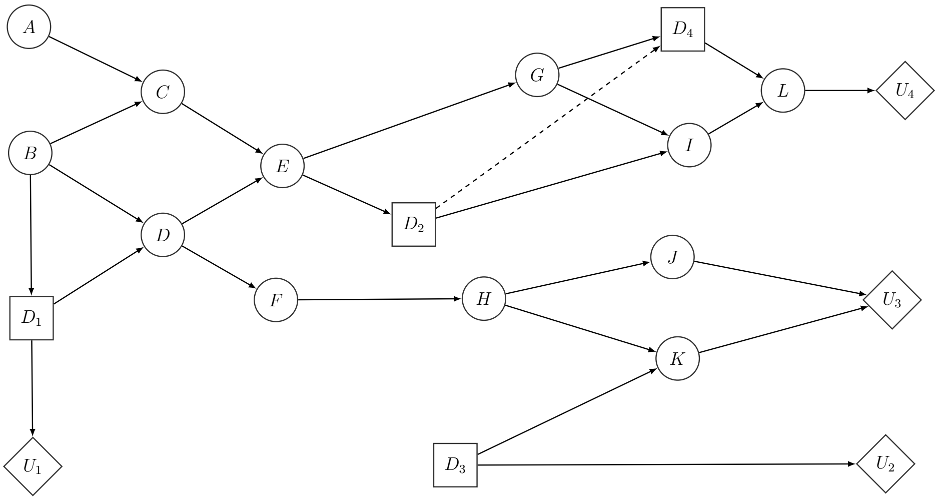


Figure 4.7: ID shown in Figure 4.6 after removing all redundant arcs.

4.5 Influence diagrams evaluation algorithms

4.5.1 Variable elimination

Variable elimination (VE) is a typical approach to inference in graphical models such as Bayesian networks [116]. VE algorithms for IDs [64] are commonly used to solve Equation (4.26): it starts with a set of potentials and it eliminates all the variables one by one. Unlike VE for Bayesian networks, in regular IDs the elimination order is not arbitrary: it should be the inverse of an order consistent with the partial order associated to the ID (called strong elimination order [68]). When the last variable is eliminated, the algorithm returns a potential with no arguments (i.e., a single value) whose value is the MEU in Equation (4.26). Every time a decision variable is eliminated, the corresponding optimal policy is also obtained. The general scheme of VE for IDs is shown in Algorithm 1.

Algorithm 1 Variable Elimination Scheme

```
input : $\Phi, \Psi$ (sets of potentials in the ID), $\{\mathcal{I}_0, D_1, \mathcal{I}_1, \dots, D_n, \mathcal{I}_n\}$ (partitions of nodes in the ID).
for $k \leftarrow n$ to $0$ do
    while $\mathcal{I}_k \neq \emptyset$ do
        Select $X \in \mathcal{I}_k$                                  ▷ Pick a chance variable to eliminate
        $(\Phi, \Psi) \leftarrow \mathrm{ElimVar}(X, \Phi, \Psi)$      ▷ Chance variable elimination (Algorithm 2)
        $\mathcal{I}_k \leftarrow \mathcal{I}_k \backslash \{X\}$
    end while
    if $k > 0$ then
        $(\Phi, \Psi) \leftarrow \mathrm{ElimVar}(D_k, \Phi, \Psi)$    ▷ Decision variable elimination (Algorithm 2)
    end if
end for
```

Another difference with regard to the VE for Bayesian networks is related to the elimination of decision variables. While chance variables are removed by sum, as in Bayesian networks, decision variables are instead eliminated by maximization. Algorithm 2 shows how to remove a single variable, no matter whether chance or decision, from an ID. All the required operations with potentials were defined in Section 4.3.2.1.

In order to remove a variable Y, all the potentials containing such variable in their domains are selected and combined, giving as a result a PP and a UP denoted as ϕ_Y and ψ_Y (lines 1 and 2). Then, in case of a chance variable, Y is separately removed from the PPs and UPs using sum-marginalization (line 4). In case of a decision (line 6), Y can be eliminated by instantiating (restriction) an arbitrary value $y \in \Omega_Y$: when removing a decision, usually there are not PPs containing it and if any, the decision is not affecting the values of such PP since any decision is d-separated from its predecessors (see Proposition 5) and any successor has already been removed. When eliminating a decision variable, the resulting UP is obtained by removing Y using the max-marginalization operation. In addition, the argument that maximizes such UP also gives the corresponding optimal policy (line 7).

Algorithm 2 ElimVar - Elimination of a single variable

```
input :  $Y$ (variable to remove), $\Phi, \Psi$ (sets of current potentials)
output :  $\Phi, \Psi$ (updated sets of current potentials without $Y$)
 1: $(\Phi_Y, \Psi_Y) \leftarrow (\{\phi \in \Phi | Y \in dom(\phi)\}, \{\psi \in \Psi | Y \in dom(\psi)\})$      ▷ Select
 2: $(\phi_Y, \psi_Y) \leftarrow (\prod_{\phi \in \Phi_Y} \phi, \sum_{\psi \in \Psi_Y} \psi)$      ▷ Combine
 3: if $Y \in \mathcal{U}_C$ then
 4:     $(\phi'_Y, \psi'_Y) \leftarrow (\sum_Y \phi_Y, \frac{\sum_Y \phi_Y \cdot \psi_Y}{\sum_Y \phi_Y})$      ▷ Remove by sum (chance vars)
 5: else
 6:     $(\phi'_Y, \psi'_Y) \leftarrow (\phi_Y^{R(Y=y)}, \max_Y \psi_Y)$      ▷ Remove by max (decision vars)
 7:     $\widehat{\delta}_Y \leftarrow \arg\max_Y \psi_Y$      ▷ Optimal policy
 8: end if
 9: $(\Phi, \Psi) \leftarrow (\Phi \backslash \Phi_Y \cup \{\phi'_Y\}, \Psi \backslash \Psi_Y \cup \{\psi'_Y\})$      ▷ Update
10: return $(\Phi, \Psi)$
```

Finally, in line 9 the sets of current potentials in the ID are updated: potentials containing variable Y are replaced by those obtained in lines 4 and 6. If the resulting PP is a unity potential, this potential does not need to be included in the potential set Φ.

Example 18 (evaluation of the oil wildcatter's ID using VE) *Let us consider the ID shown in Example 11 with the potentials* $\Phi = \{\phi(O), \phi(S|O,T)\}$*, and* $\Psi = \{\psi(T), \psi(O,D)\}$*. Then, the computations required for evaluating such ID using the VE algorithm considering the elimination order* $\{O, D, S, T\}$ *are:*

1. $\text{ElimVar}(O, \Phi, \Psi)$*:*

   (a) Select and combine relevant potentials (with variable O*):*

   $\phi(O,S|T) \leftarrow \phi(O) \cdot \phi(S|O,T)$ *(probabilities)*

   $\psi(O,D)$ *(utilities)*

   (b) Remove by sum:

   $\phi(S|T) \leftarrow \sum_O \phi(O,S|T) \quad \psi(S,T,D) \leftarrow \frac{\sum_O \phi(O,S|T) \cdot \psi(O,D)}{\sum_O \phi(O,S|T)}$

(c) *Update the sets of current potentials:*

$\Phi \leftarrow \{\phi(S|T)\} \quad \Psi \leftarrow \{\psi(S,T,D), \psi(T)\}$

2. $\mathrm{ElimVar}(D, \Phi, \Psi)$*:*

(a) *Select and combine relevant potentials (with variable D):*

$\psi(S,T,D)$ *(utilities)*

(b) *Remove by max:*

$\psi(S,T) \leftarrow \max_D \psi(S,T,D)$

$\widehat{\delta}_D(S,T) \leftarrow \arg\max_D \psi(S,T,D)$

(c) *Update the sets of current potentials:*

$\Phi \leftarrow \{\phi(S|T)\} \quad \Psi \leftarrow \{\psi(S,T), \psi(T)\}$

3. $\mathrm{ElimVar}(S, \Phi, \Psi)$*:*

(a) *Select and combine relevant potentials (with variable S):*

$\phi(S|T)$ *(probabilities)*

$\psi(S,T)$ *(utilities)*

(b) *Remove by sum:*

$1_T \leftarrow \sum_S \phi(S|T) \quad \psi_2(T) \leftarrow \frac{\sum_S \phi(S|T)\cdot\psi(T)}{\sum_S \phi(S|T)}$

(c) *Update the sets of current potentials:*

$\Phi \leftarrow \emptyset \quad \Psi \leftarrow \{\psi(T), \psi_2(T)\}$

4. $\mathrm{ElimVar}(T, \Phi, \Psi)$*:*

(a) *Select and combine relevant potentials (with variable T):*

$\psi_3(T) \leftarrow \psi(T) + \psi_2(T)$ *(utilities)*

(b) *Remove by max:*

$MEU \leftarrow \max_D \psi_3(T)$

$\widehat{\delta}_T \leftarrow \arg\max_D \psi_3(T)$

The set of all variables contained in the relevant potentials for the removal of a variable is called *clique candidate* or *group*. That is $C_Y := \{dom(\phi_Y) \cup dom(\psi_Y)\}$ is the clique candidate created when Y is removed. Even though the concept of clique is usually only used with triangulation algorithms, it corresponds with the variables involved during the elimination of a variable using VE algorithm. The *size* of a clique candidate $|C_Y|$ is the number of variables on it. The *weight* of a clique candidate is the product of the number of states of each variable, i.e. $w(C_Y) := \prod_{X_i \in C_Y} |\Omega_{X_i}|$.

During the elimination process, it could happen that two variables, that are not together in any of the potentials, might appear in the new potentials resulting of removing a variable. When that happens, we say that a *fill-in arc* has been added. For example, let us consider we aim to remove a variable Y from $\phi(Y|A)$ and $\psi(Y, B)$, then the resulting UP is $\psi(A, B)$. In this case, a fill-in arc is added between A and B as there is not any potential in $\Phi_Y \cup \Psi_Y$ containing both variables. The weight of an arc (A, B) is defined as $w(A, B) := |\Omega_A| \cdot |\Omega_B|$.

The complexity of VE is linear in the size of the largest potential generated during the evaluation [70], which will be either the result of the combination of $\phi_Y \cdot \psi_Y$ in line 4 of Algorithm 2 or the potential ψ_Y in line 6. That is, the complexity of VE for evaluating and ID with n variables (chance or decision) is $\mathcal{O}(n \cdot N_{max})$ where N_{max} is the largest potential ever created during the evaluation. However, the size of a potential is exponential in the number of variables in its domain. Thus, the computational cost of the VE algorithm depends on the sizes of the intermediate potentials generated. Note that the complexity can be related to the notion of *treewidth* [6, 98] which correspond swith the size of the largest clique generated if the optimal order is followed. Suppose we have an ID with treewidth w, then the complexity of VE, following an optimal elimination order, is $\mathcal{O}(n \cdot exp(w))$.

4.5.1.1 Elimination heuristics

An element of crucial importance for the efficiency of the VE algorithm is finding an optimal elimination order, which can reduce the complexity of operations performed during the ID evaluation. Like for BNs, the problem of finding an optimal order (an optimization problem) is NP-hard [68]. In fact, any method for finding an optimal order in a BN can be adapted for IDs. The single difference is that it must be consistent with the partial temporal order $\mathcal{I}_0 \prec D_1 \prec \mathcal{I}_1 \prec \cdots \prec D_n \prec \mathcal{I}_n$. Thus, the problem consists on finding an order for each $\mathcal{I}_i$. Several approaches have been proposed in order to search for close-to-optimal elimination orderings. Some of the most efficient methods are greedy algorithms that choose at each step the next variable to remove using a deterministic heuristic. Some of these heuristics are:

- **Minimum size**: this heuristic is based on selecting as the next variable to be removed that one which minimises the size of the generated candidate clique [99]. That is, it selects a variable Y with a minimal $|C_Y|$.
- **Minimum weight**: this heuristic is based on selecting as the next variable to be removed that one which minimises the weight of the generated candidate clique [68]. That is, it selects a variable X_i with a miminal $w(C_Y)$.
- **Cano and Moral**: this heuristic is very similar to *minumum weight*, at each case it chooses a variable X_i that minimises $w(C_Y)/|\Omega_Y|$ [27]. That is, it first removes variables of a larger number of states.
- **Minimum fill-in arcs weight**: this heuristic selects a variable to remove that one which minimises the weights of the fill-in arcs added [45].

4.5.2 Arc reversal

Arc reversal (AR) [102] is another evaluation algorithm for IDs. In AR the orientation of an arc among two chance nodes can be reversed by Bayes rule. AR is based on a simple observation: the elimination of a variable, no matter whether chance or decision, having a utility node as unique direct successor involves only two potentials, thus does not affect the overall inferential complexity. Such patterns can be always created by properly changing the orientation of some arcs. In the original proposal, AR copes with IDs with a single utility node. If this is not the case, it is sufficient to apply Definition 26.

Definition 26 (merging utility nodes) *Given an ID with multiple utility nodes and assuming an additive model, add a new utility node $\tilde{U}$, which is a barren child of all the parents of the utility nodes, i.e., $pa(\tilde{U}) := \bigcup_{U \in \mathcal{U}_V} pa(U)$. Define a UP associated to $\tilde{U}$ as $\psi(pa(\tilde{U})) := \sum_{U \in \mathcal{U}_V} \psi(pa(U))$. Finally, remove all the utility nodes different from $\tilde{U}$ and the corresponding UPs.*

In order to evaluate an ID, AR uses three basic transformations: *chance node removal*, *decision node removal* and *arc reversal*. These transformations, which are described here below, maps the original ID into an *equivalent* one (two IDs are equivalent if they have the same expected utility and the same optimal policies for the remaining decisions).

Transformation 1 (chance node removal) *Assume that a chance node Y has the utility node U as unique direct successor. Let $\mathbf{X}_I$ be the direct predecessors of Y, and let $\mathbf{X}_J$ the direct predecessors of U without Y. The elimination of Y is done by conditional expectation: replace the PP $\phi(Y|\mathbf{X}_I)$ and the UP $\psi(Y, \mathbf{X}_J)$, with the UP $\psi(\mathbf{X}_{I \cup J}) := \sum_{y \in \Omega_Y} \psi(y, \mathbf{X}_J) \cdot \phi(y|\mathbf{X}_I)$. We finally remove Y from $\mathcal{G}$ and add arcs to connect the nodes in $\mathbf{X}_I$ with U.*

Transformation 2 (decision node removal) *Assume that a decision node D has the utility node U as unique direct successor. Let $\mathbf{X}_I$ be the direct predecessors of Y. Assume that the direct predecessors of U others than D, and denoted as $\mathbf{X}_J$, are also direct predecessors of D. To eliminate D, replace the UP $\psi(D, \mathbf{X}_J)$ with the UP $\psi(\mathbf{X}_J) := \max_{d \in \Omega_D} \psi(d, \mathbf{X}_J)$. Finally, D should be removed from the graph together with its incoming and leaving arcs.*

Transformation 3 (arc reversal) *Assume that the chance nodes Y and X are directly connected by an arc, but not by other directed paths. Let $\phi(Y|\mathbf{X}_I)$ and $\phi(X|Y, \mathbf{X}_J)$ be the relative PPs, which means that $\mathbf{X}_I$ are the direct predecessors of Y and $\mathbf{X}_J$ those of X others than Y. Change the orientation of the arc and add arcs from $\mathbf{X}_I$ towards X and from $\mathbf{X}_J$ towards Y. The new PP for X is $\phi(X|\mathbf{X}_I, \mathbf{X}_J) := \sum_y \phi(y|\mathbf{X}_I) \cdot \phi(X|y, \mathbf{X}_J)$, while the PP for Y is such that $\phi(y|x, \mathbf{X}_I, \mathbf{X}_J) \propto \phi(y|\mathbf{X}_I) \cdot \phi(x|Y, \mathbf{X}_J)$, with the proportionality constant obtained by normalization.*

AR applies previous transformations until only the utility node remains. The UP attached to this unique utility node is the MEU. Algorithm 3 outlines the whole scheme: at each iteration, the algorithm tries to remove a chance node (lines 2 to 3); if it's not possible a decision might be removed (lines 5 to 7). In case neither a chance node nor a decision could be removed, an arc $X \rightarrow Y$ is reversed and X is removed (lines 10 to 16).

The optimal policies are computed when each decision D is removed by Transformation 2, i.e. $\widehat{\delta}_D(\mathbf{X}_J) := \arg\max_D \psi(D, \mathbf{X}_J)$. After the removal of a decision, any of the chance or decision nodes might become barren. In that case, they can be directly removed.

Algorithm 3 ArcRev - Arc Reversal Scheme

```
 1: while $pa(U) \neq \emptyset$ do
 2:     if exists $X \in \mathcal{U}_C \cap pa(U)$ such that $ch(X) = \{\ U\}$ then
 3:         remove $X$                                        ▷ Transformation 1
 4:
 5:     else if exists $D \in \mathcal{U}_D \cap pa(U)$ such that $pa(U) \subset pa(D) \cup \{D\}$ then
 6:         remove $D$                                        ▷ Transformation 2
 7:         remove barren nodes
 8:
 9:     else
10:         find $X \in \mathcal{U}_C \cap pa(U)$ such that $\mathcal{U}_D \cap ch(X) = \emptyset$
11:         while $\mathcal{U}_C \cap ch(X) \neq \emptyset$ do
12:             find $Y \in \mathcal{U}_C \cap pa(X)$ such that
13:             there is no other directed path from $X$ to $Y$
14:             reverse arc $X \rightarrow Y$                 ▷ Transformation 3
15:         end while
16:         remove $X$
17:     end if
18: end while
```

Compared to VE, the complexity of AR is not reduced because of the additional arcs added when reversing the arcs. Unlike VE, each step of AR can be regarded as a transformation of an ID in an equivalent one with fewer variables. It has been proved empirically that the complexity of VE is never higher than the complexity of AR [1, 9].

Example 19 (Evaluation of the oil wildcatter's ID using AR) *Consider the ID in Example 11 with the graph in Figure 4.2. We first apply Definition 26 to merge the two utility nodes C and P. The resulting equivalent ID with a single utility node is in Figure 4.8.a. Then we reverse the arc from O to S by Definition 3. As shown in Figure 4.8.b, this makes the utility node $\tilde{P}$ the unique direct successor of O. The new PPs attached to nodes O and S are:*

$$\phi(O|S,T) = \frac{\phi(O) \cdot \phi(S|O,T)}{\sum_O \phi(O) \cdot \phi(S|O,T)} \qquad \phi(S|T) = \sum_O \phi(O) \cdot \phi(S|O,T)$$

The chance node O can be now eliminated by means of the procedure in Transformation 1, resulting the graph shown in Figure 4.8.c. Quantitatively this corresponds to replace the potentials $\phi(O|S,T)$ and $\psi(O,D)$ with the following UP:

$$\psi(S,T,D) = \sum_O \phi(O|S,T) \cdot \psi(O,S)$$

In the resulting model, D can be removed by Transformation 2 since the other direct predecessors of $\tilde{P}$ are also predecessors of D. The resulting graph is depicted in Figure 4.8.d while the new UP attached to the utility node and the optimal policy for D are computed as follows:

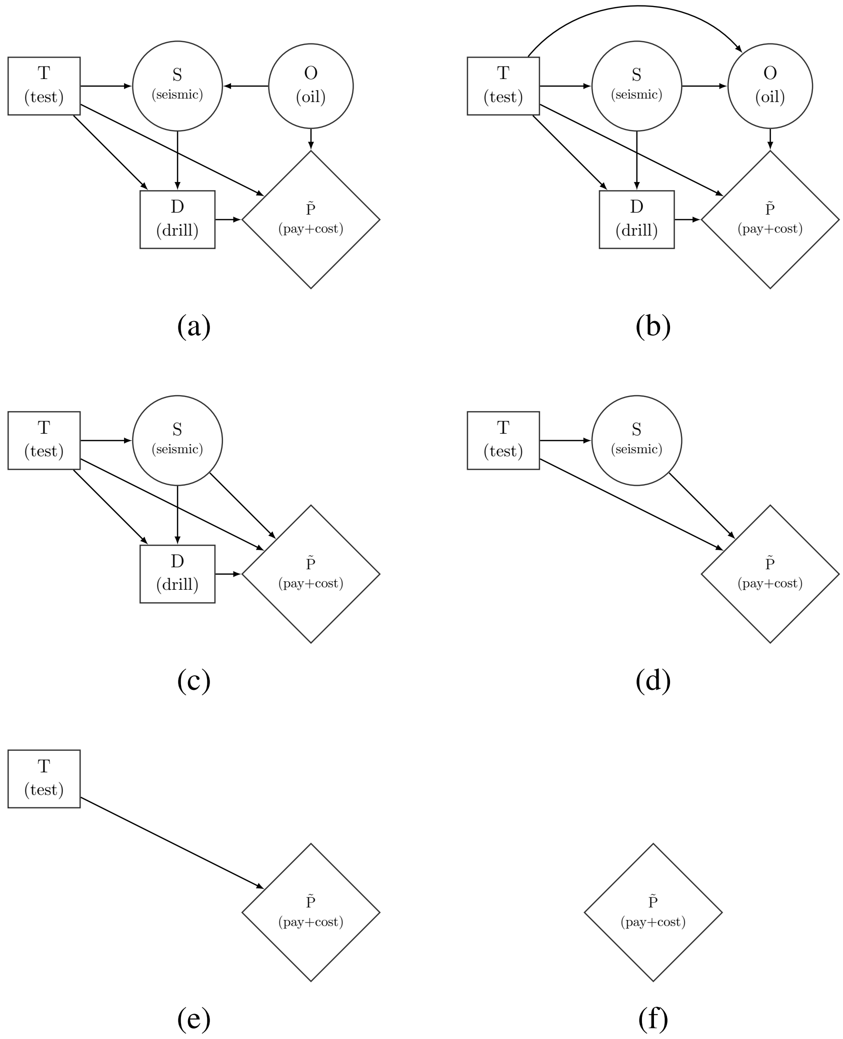
$$\psi(S,T) = \max_D \psi(S,T,D) \qquad \widehat{\delta}_D(S,T) = \arg\max_D \psi(S,T,D)$$

By similarly continuing we eliminate chance node S, resulting the ID shown in Figure 4.8.e and the following UP attached to $\tilde{P}$:

$$\psi(T) = \sum_S \phi(S|T) \cdot \psi(S,T)$$

Finally, decision node T can be removed. The resulting ID contains a unique utility node (see Figure 4.8.f), whose constant UP is the MEU. The optimal policy for T is computed as well:

$$MEU = \max_{T} \psi(T) \qquad \widehat{\delta}_T = \arg\max_{T} \psi(T)$$



(a) (b) (c) (d) (e) (f)

Figure 4.8: Transformations applied to the graph during the evaluation the oil wildcatter's ID using the AR algorithm.

4.5.3 Lazy evaluation

Lazy evaluation (LE) was already used for making inference in BNs [80], so it can be adapted for evaluating IDs [79]. The basic idea of this method is to maintain the decomposition of the potentials and to postpone computations for as long as possible, as well as to exploit barren variables. LE is based on message passing in a *strong junction tree*, which is a representation of an ID built by moralization and by triangulating the moral graph using a strong elimination order [68]. Nodes in the strong junction trees correspond to *cliques* (maximal complete sub-graphs) of the triangulated graph. Each clique is denoted by C_i where i is the index of the clique. The root of the strong junction tree is denoted by C_1. Two neighbour cliques are connected by a separator which contains the intersection of the variables in both cliques. Figure 4.9 shows the strong junction tree for the ID shown in Figure 4.6.

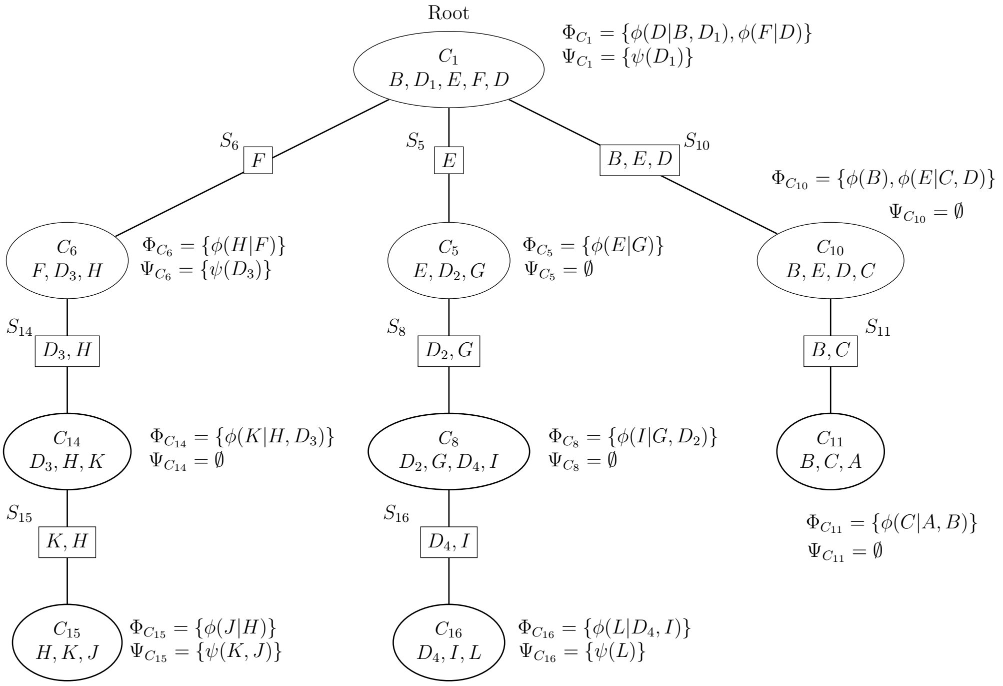

Root
C_1
B, D_1, E, F, D
$\Phi_{C_1} = \{\phi(D|B, D_1), \phi(F|D)\}$
$\Psi_{C_1} = \{\psi(D_1)\}$
S_6
F
S_5
E
B, E, D
S_{10}
$\Phi_{C_{10}} = \{\phi(B), \phi(E|C, D)\}$
$\Psi_{C_{10}} = \emptyset$
C_6
F, D_3, H
$\Phi_{C_6} = \{\phi(H|F)\}$
$\Psi_{C_6} = \{\psi(D_3)\}$
C_5
E, D_2, G
$\Phi_{C_5} = \{\phi(E|G)\}$
$\Psi_{C_5} = \emptyset$
C_{10}
B, E, D, C
S_{14}
D_3, H
S_8
D_2, G
B, C
S_{11}
C_{14}
D_3, H, K
$\Phi_{C_{14}} = \{\phi(K|H, D_3)\}$
$\Psi_{C_{14}} = \emptyset$
C_8
D_2, G, D_4, I
$\Phi_{C_8} = \{\phi(I|G, D_2)\}$
$\Psi_{C_8} = \emptyset$
C_{11}
B, C, A
S_{15}
K, H
S_{16}
D_4, I
$\Phi_{C_{11}} = \{\phi(C|A, B)\}$
$\Psi_{C_{11}} = \emptyset$
C_{15}
H, K, J
$\Phi_{C_{15}} = \{\phi(J|H)\}$
$\Psi_{C_{15}} = \{\psi(K, J)\}$
C_{16}
D_4, I, L
$\Phi_{C_{16}} = \{\phi(L|D_4, I)\}$
$\Psi_{C_{16}} = \{\psi(L)\}$

Figure 4.9: Strong junction tree for the ID shown in Figure 4.6 with the sets of potentials associated to each clique.

LE is a message-passing algorithm whose general scheme is shown in Algorithm 4. Initially, each potential in the ID is associated to the clique closest to the root containing all its variables. These potentials are not combined, so during propagation each clique and separator keeps two sets of potentials (one for probabilities and another for utilities). Sets of potentials stored in a clique C_j are denoted Φ_{C_j} and Ψ_{C_j}. Similarly, sets of potentials (or messages) stored in a separator S_j are denoted $\Phi^*_{S_j}$ and $\Psi^*_{S_j}$. Message propagation starts by invoking the *CollectMessage* in the root (Algorithm 5). This algorithm is used for traversing the tree until the leaves.

Algorithm 4 LazyEvaluation

```
input : $\Phi, \Psi$ (sets of potentials in the ID), $\{\mathcal{I}_0, D_1, \mathcal{I}_1, \dots, D_n, \mathcal{I}_n\}$ (partitions of nodes in the ID).
1: Build a strong junction tree with root $C_1$ from the ID
2: Associate each potential in $\Phi \cup \Psi$ to the clique closest to the root containing
   all its variables
3: Invoke CollectMessage in $C_1$                                    ▷ Algorithm 5
```

Algorithm 5 CollectMessage

```
   /* Let $C_j$ be a clique where Collect Message is invoked,
   then:*/
1: $C_j$ invokes Collect Message in all its children
2: The message to the clique parent of $C_j$ is built and sent by absorption (Algorithm 6)
```

When a clique has received all the messages from its children it can send the message to its parent (*Absorption*) as detailed in Algorithm refalgo:absorb. Consider a clique C_j and its parent separator S_j. Absorption in C_j amounts to eliminating the variables of $C_j \backslash S_j$ from the list of probability and utility potentials associated with C_j and with each separator $S' \in ch(C_j)$ and then associating the obtained potentials with S_j. Initially, the algorithm collects the potentials in the clique an child seprators (line 1). Then it determines which are the variables $\mathbf{X}$ to be removed, i.e. those present in the clique but not in the parent separator (line 2). Afterwards, a removal order is chosen and the removes $\mathbf{X}$ from the relevant potentials. Finally, the resulting sets of potentials are associated to the parent separator.

Algorithm 6 Absorption

```
/* Let $C_j$ be a clique, $S_j$ be the parent separator
and $S' \in ch(C_j)$ be each of the child separators. If
Absorption is invoked on $C_j$, then:*/
1: $\mathcal{R}_{S_j} \leftarrow \Phi_{C_j} \cup \Psi_{C_j} \cup \bigcup_{S' \in ch(C_j)} (\Phi^*_{S'} \cup \Psi^*_{S'})$    ▷ Relevant potentials
2: $\mathbf{X} \leftarrow \{X | X \in C_j, X \notin S_j\}$;    ▷ Variables to be removed
3: Choose a strong order to remove the variables in $\mathbf{X}$
4: Marginalize out all variables in $\mathbf{X}$ from $\mathcal{R}_{S_j}$. Let $\Phi^*_{S_j}$ and $\Psi^*_{S_j}$ be the set of probability and utility potentials obtained
5: Associate $\Phi^*_{S_j}$ and $\Psi^*_{S_j}$ to the parent separator $S_j$
```

Note that the original proposal [79] uses VE for removing the variables. Thus, we will refer to this method as VE-Lazy Evaluation (VE-LE). Figure 4.10 shows the flow of messages in strong junction tree for the ID shown in Figure 4.6.

The propagation finishes when the root clique has received all the messages. The optimal policy $\widehat{\delta}_{D_i}$ is recorded when D_i is eliminated from the closest clique to the root and containing the decision. In case of decisions that are attached to the root node, the optimal policy is calculated by marginalizing out all the variables in the root clique that do not belong to the relevant past of the decision. Finally to compute the MEU, all the variables in the root node are eliminated.

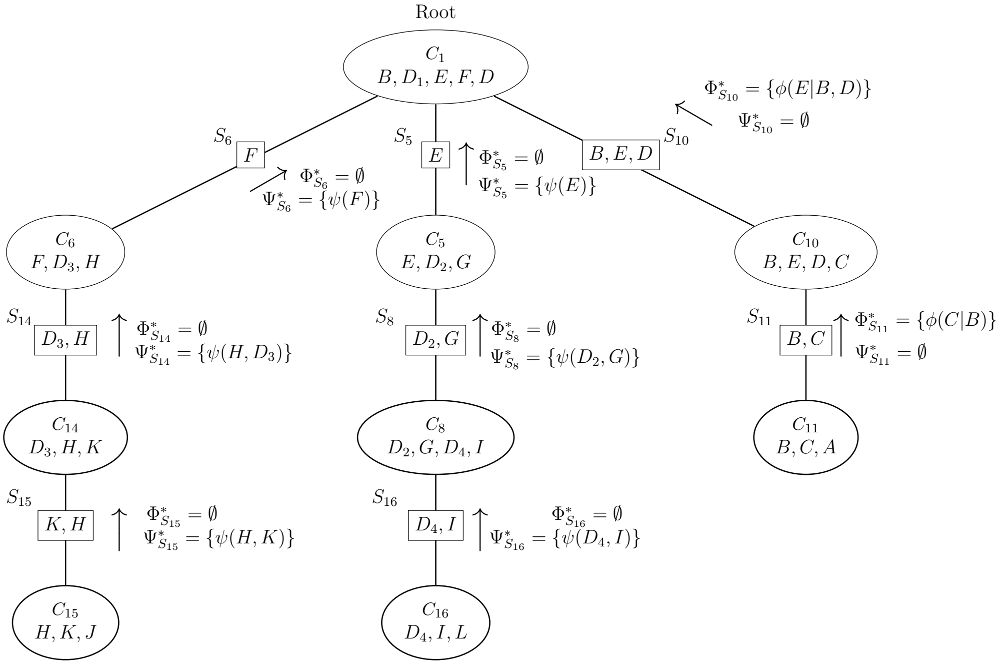

Root
C_1
B, D_1, E, F, D
S_6
F
$\Phi^*_{S_6} = \emptyset$
$\Psi^*_{S_6} = \{\psi(F)\}$
S_5
E
$\Phi^*_{S_5} = \emptyset$
$\Psi^*_{S_5} = \{\psi(E)\}$
B, E, D
S_{10}
$\Phi^*_{S_{10}} = \{\phi(E|B, D)\}$
$\Psi^*_{S_{10}} = \emptyset$
C_6
F, D_3, H
C_5
E, D_2, G
C_{10}
B, E, D, C
S_{14}
D_3, H
$\Phi^*_{S_{14}} = \emptyset$
$\Psi^*_{S_{14}} = \{\psi(H, D_3)\}$
S_8
D_2, G
$\Phi^*_{S_8} = \emptyset$
$\Psi^*_{S_8} = \{\psi(D_2, G)\}$
S_{11}
B, C
$\Phi^*_{S_{11}} = \{\phi(C|B)\}$
$\Psi^*_{S_{11}} = \emptyset$
C_{14}
D_3, H, K
C_8
D_2, G, D_4, I
C_{11}
B, C, A
S_{15}
K, H
$\Phi^*_{S_{15}} = \emptyset$
$\Psi^*_{S_{15}} = \{\psi(H, K)\}$
S_{16}
D_4, I
$\Phi^*_{S_{16}} = \emptyset$
$\Psi^*_{S_{16}} = \{\psi(D_4, I)\}$
C_{15}
H, K, J
C_{16}
D_4, I, L

Figure 4.10: Flow of messages in a strong junction tree for the ID shown in Figure 4.6

Part II

Representation

Chapter 5

Binary Trees

5.1 Introduction

Traditionally, potentials in PGMs with discrete variables (and so in IDs) have been represented using tables. In fact, this is the data structure used in previous chapters to introduce the concepts of potential and CPD. A table representing a PP $\phi(\mathbf{X}_I|\mathbf{X}_J)$ is denoted $\mathcal{T}^{\phi}(\mathbf{X}_I, \mathbf{X}_J)$. Similarly, a table representing a UP $\psi(\mathbf{X}_I)$ is denoted $\mathcal{T}^{\psi}(\mathbf{X}_I)$. Example 20 shows two potentials represented as tables.

Example 20 (A PP and a UP represented using tables) *Let X, Y, Z and D four discrete variables whose domains are $\Omega_X = \{x_1, x_2, x_3, x_4\}$, $\Omega_Y = \{y_1, y_2, y_3\}$, $\Omega_Z = \{z_1, z_2\}$ and $\Omega_D = \{d_1, d_2, d_3\}$. Let $\phi(X|Y, Z)$ and $\psi(Y, Z, D)$ be a PP and a UP respectively with the following representation as tables:*

$$\mathcal{T}^{\phi}(X,Y,Z) = \begin{array}{cc} \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array} & \\ \begin{bmatrix} 0.25 & 0.25 & 0.1 & 0.4 \\ 0.25 & 0.25 & 0.1 & 0.4 \\ 0.25 & 0.25 & 0.1 & 0.4 \\ 0.1 & 0.5 & 0.2 & 0.2 \\ 0.1 & 0.5 & 0.2 & 0.2 \\ 0.15 & 0.35 & 0.3 & 0.2 \end{bmatrix} & \begin{array}{cc} y_1 & \\ y_2 & z_1 \\ y_3 & \\ y_1 & \\ y_2 & z_2 \\ y_3 & \end{array} \end{array}$$

$$\begin{array}{r|cccccc|l}
 & & z_1 & & & z_2 & & \\
 & d_1 & d_2 & d_3 & d_1 & d_2 & d_3 & \\
\mathcal{T}^{\psi}(Y,Z,D) = & 30 & 30 & 30 & 30 & 30 & 30 & y_1 \\
 & 45 & 45 & 45 & -5 & -5 & 25 & y_2 \\
 & 0 & 0 & 0 & 0 & 0 & 5 & y_3
\end{array}$$

One problem of tables is that this representation is exhaustive, which means that a table entry is required for all the possible configurations of the variables in a given potential. The number of these entries is the *size* of the table, and can be defined more formally as follows.

Definition 27 (Size of a table representing a potential) *Let ψ be a potential (no matter whether PP or UP) defined over the set of variables $\mathbf{X}_I$. Then, the size of a table $\mathcal{T}^{\psi}$ representing such potential is:*

$$size(\mathcal{T}^{\psi}) = \prod_{X_i \in \mathbf{X}_I} |\Omega_{X_i}|$$

Note that the size of a table increases exponentially with the number of variables. During the evaluation, intermediate potentials can be extremely large as they usually contain many variables. As a consequence, the evaluation of IDs modelling complex decision problems may become infeasible due to its computational cost: the set of information states may exceed the storage capacity of a computer or the optimal policy could need a large computation time to be obtained.

To address this problem, IDs can be evaluated with alternative methods such as *LIMIDs* [72], or simulation techniques [31, 22]. Other solutions propose using alternative representations for the potentials (instead of tables) trying to offer efficient data structures for storing and managing quantitative information (probabilities and utilities). In Section 5.2, we review some of these alternative representations such as *recursive probability trees* (RPTs) or *numerical trees* (NTs). These representations are tree-based where identical values can be grouped into a single one offering a compact storage. This is possible due to the presence of *context-specific independencies (CSIs)* [7]. Moreover, when trees are too large they can be pruned and converted into smaller trees leading to approximate encodings. As a consequence, less memory space is required for storing the potentials using this tree-based structures.

In this dissertation, we propose representing potentials in IDs using *binary trees* (BTs), i.e. a tree-based structure whose internal nodes always have two children. The advantage of BTs resides in their capability of representing not only CSIs, but also other forms of independencies which are more fine-grained compared to those encoded using NTs. In Section 3.3.4 we described some of these forms of independence such as *partial conditional independencies (PCIs)* [92, 75] and *contextual weak independencies (CWIs)* [110, 10]. This enhanced capability makes the representation of potentials even more compact. As PCIs and CWIs are quite frequent in large IDs representing real world decision problems, their evaluation should be more efficient. In addition, approximate solutions obtained with BTs should be more accurate than those obtained with NTs.

In the literature about PGMs, there are other approaches for taking advantage of these types of independecies. Geiger and Heckerman [52] proposed the use of *Bayesian multinets*. This approach consists on defining multiple standard BNs, one for each context. Boutilier et al. [7] proposed using a single BNs but including auxiliary nodes to capture CSIs. Some types of PGMs can encode natively CSIs, such as *probabilistic decision graphs* (PDGs) proposed by M. Jaeger [59]. It should be remarked, however, that these approches are not applyed to IDs.

The chapter is organized as follows. Section 5.2 reviews some of the previous potential representations that take advantage of CSIs. In Section 5.3 the key concepts for representing potentials as BTs are explained. The procedures for learning and approximating BTs (from tables) are detailed in Section 5.4.

5.2 Previous approaches for potential representation

5.2.1 Numerical trees

Numerical trees (NTs), also called *probability trees*, have been used to represent potentials in BNs [28, 101] and IDs [54]. This tree-like representation takes advantage of context-specific independencies (see Section 3.3.4.1) so that many identical values can be grouped into a single one offering a compact storage. Moreover, when NTs are too large they can be pruned and converted into smaller trees leading to approximate encodings. A NT can be defined as follows.

Definition 28 (numerical tree) *A NT defined over the set of variables $\mathbf{X}_I$ is a directed tree, where each internal node is labelled with a variable (random or decision), and each leaf node is labelled with a number (a probability or a utility value). We use L_t to denote the label of node t. Each internal node has an outgoing arc for each state of the variable associated with that node. Outgoing arcs from node X_i are labelled with a state ($x_i \in \Omega_{X_i}$) of X_i.*

NTs can be used for representing any function defined over a set of discrete variables. However, we will focus on how they can be used for representing the potentials involved in an ID: a NT representing a PP $\phi(\mathbf{X}_I|\mathbf{X}_J)$ will be called *numerical probability tree* and denoted $\mathcal{NT}^{\phi}(\mathbf{X}_I, \mathbf{X}_J)$. Similarly, a NT representing a UP $\psi(\mathbf{X}_I)$ will be called *numerical utility tree* and denoted $\mathcal{NT}^{\psi}(\mathbf{X}_I)$.

A numerical tree on the set of variables $\mathbf{X}_I = \{X_i | i \in I\}$ represents a potential (no matter whether PP or UP) $\psi : \Omega_{\mathbf{X}_I} \rightarrow \mathbb{R}$ if for each $\mathbf{x}_I \in \Omega_{\mathbf{X}_I}$ the value $\psi(\mathbf{x}_I)$ is the number stored in the leaf node that is reached by starting from the root node and selecting the child corresponding to state x_i for each internal node labelled X_i. For example, the potentials previously shown with tables in Example 20 are depicted as NTs in Figure 5.1.

The size of a given $\mathcal{NT}$, denoted $size(\mathcal{NT})$, corresponds with the number of nodes (internal nodes and leaves). The main advantage of NTs is that the size of the potentials can be reduced since some identical values can be grouped into a

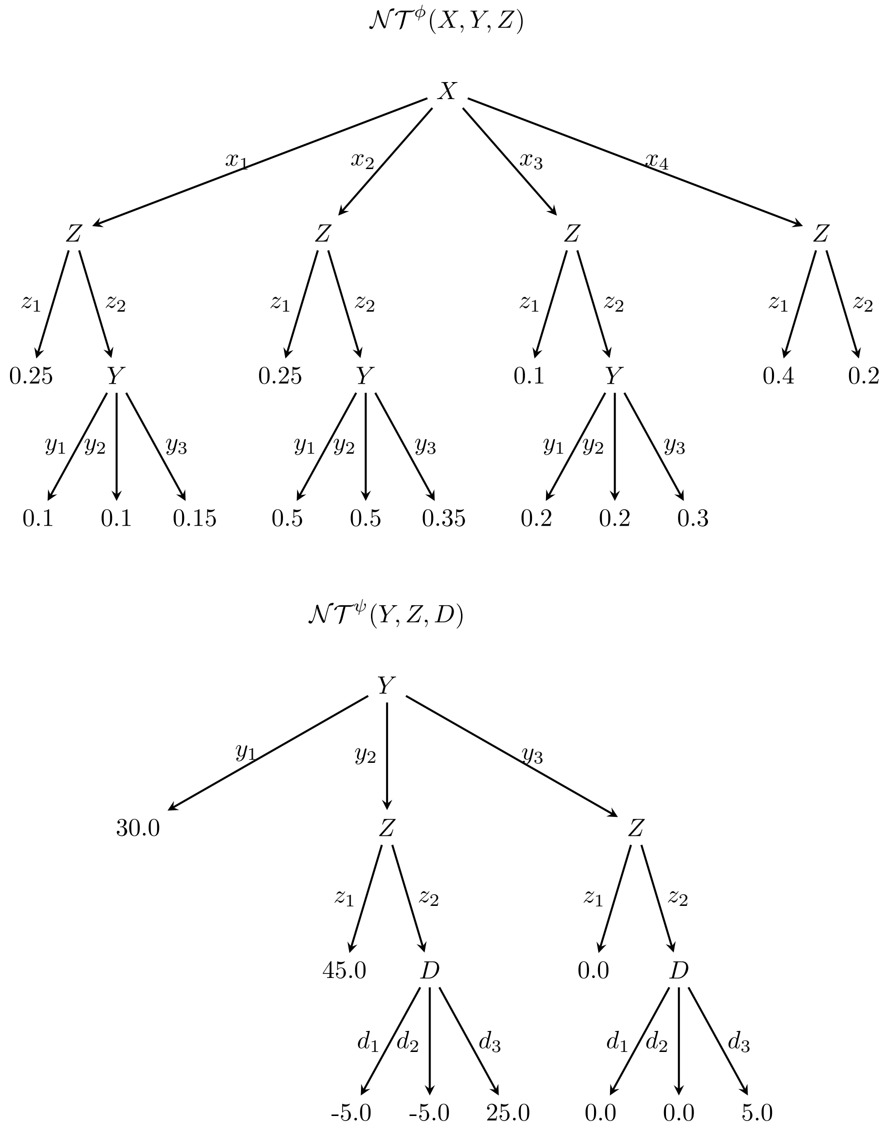


Figure 5.1: Two exact NTs representing $\phi(X|Y, Z)$ and $\psi(Y, X, D)$ from Example 20.

single one. For example, for NTs shown in Figure 5.1, we obtain the following reductions (w.r.t. the table representation): $\mathcal{T}^{\phi}(X, Y, Z)$ contains 24 table entries while $\mathcal{NT}^{\phi}(X, Y, Z)$ has 22 nodes. For the UP, the size is reduced from 18 to 14. This reduction is possible due to the presence of CSIs. For example, for $\phi(X|Y, Z)$, it holds that $I(X \perp Y | Z = z_1)$. In other words, when $Z = z_1$, the probability will not depend on the value of Y. Therefore, all the values consistent with $Z = z_1$ and with the same value of X can be represented with a single value.

When we use the term of independence, we actually refer to a probabilistic independence (e.g. the probability of X is independent of Y). In practice this means that a given PP takes the same value for a subset of configurations. A similar situation can be found in the UPs, but in this case we will talk about functional independencies. For example, in $\psi(Y, Z, D)$ we can find (among others) the following CSIs: when $Y = y_1$ the potential will always take the value 30.0; when $Y = y_2$ and $Z = z_1$ the potential is always equal to 45.0.

Another advantage is that NTs can be pruned [28, 101, 54] in order to reduce even more its storage size. Thus, approximate versions of the potentials will be obtained. Figure 5.2 presents pruned NTs approximating the potentials $\phi(X|Y, Z)$ and $\psi(Y, Z, D)$. The pruning process consists in replacing some *terminal trees*[1] by the average of their leaves. Even though this obtains an smaller representation, it introduces error during computing. But it must be considered that an exact solution will be infeasible for some complex IDs.

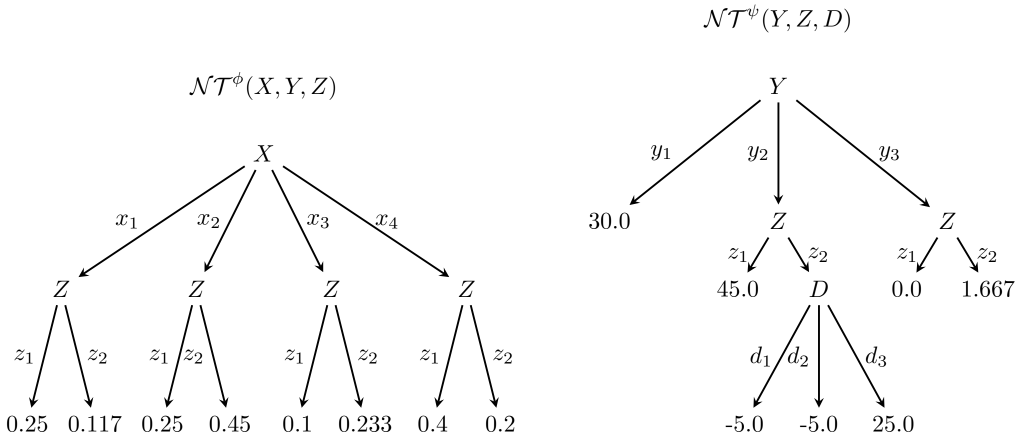


Figure 5.2: Two pruned NTs approximating $\phi(X|Y, Z)$ and $\psi(Y, Z, D)$ from Example 20.

[1] We say that a part of a NT is a terminal tree if it contains only one node labeled with a variable, and all the children are leaves.

5.2.2 Recursive probability trees

Recursive probability trees (RPTs) [23, 24, 25] are a generalization of NTs. They are also tree-like structures that allow to capture CSIs, but now they can maintain a potential in a factorized way. The recursivity comes by the fact that each factor can be again represented by a recursive subtree, and so on. So the capacity to represent context-specific independencies and decomposition of potentials at the same time, usually produce more compact representations than NTs. We can define RPTs as follows.

Definition 29 (recursive probability tree) *A RPT defined over the set of variables $\mathbf{X}_I$ is a directed tree representing a potential $\phi(\mathbf{X}_I)$. A node in a RPT can be:*

- *A Split node represents a variable $X_i \in \mathbf{X}_I$. It has an outgoing arc for each $x_i \in \Omega_{X_i}$; each state labels one arc.*
- *A List node is used to list the factors in which a potential is decomposed. It has as many outgoing arcs as factors in the decomposition. Each child of a list node is again a recursive probability sub-tree that represents a potential (a factor) for a subset of variables $\mathbf{X}_J$, $\mathbf{X}_J \subseteq \mathbf{X}_I$.*
- *A Potential node stores a potential. Internally can be represented using any data-structure (e.g. tables, NTs, etc.)*
- *A Value node stores a non-negative real number.*

Figure 5.3 shows an example of a NT and a RPT representing a PP $\phi(A, B|C)$. This RPT contains a list node depicted as a split ellipse. The rest of the internal nodes are split nodes while all the leaves are value nodes. This RPT does not contain any potential node. In this PP, we can find the following proportionality: $2 \cdot \phi(a_1, B|c_1) = \phi(a_1, B|c_2)$. Therefore, the sub-tree $\mathcal{NT}^{\phi}(a_1, B, C)$ can be expressed as a multiplicative factorization using a list node.

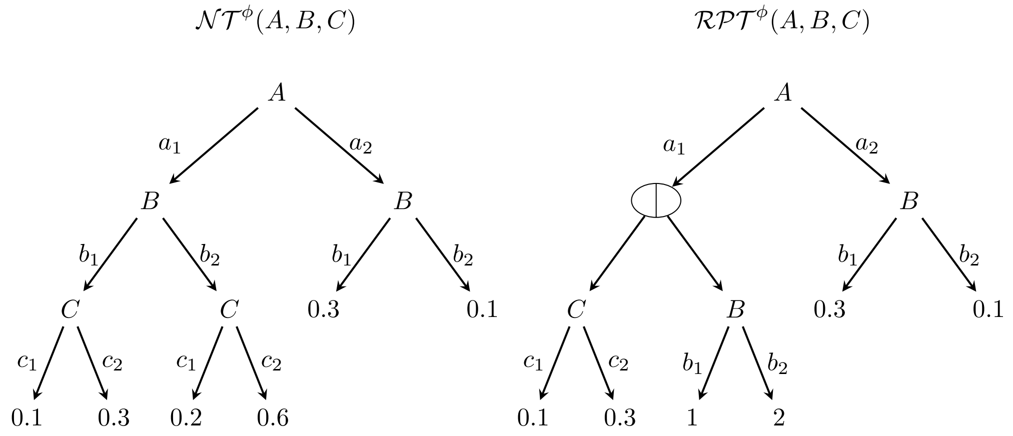


Figure 5.3: Example of a NT and a RPT representing a PP $\phi(A, B|C)$.

RPTs can be considered as a general representation for potentials involved in PGMs, moreover an RPT is able to represent a full model like a BN. It should be noticed that, until now, the only existing works about RPTs propose their use for making inference in BNs. Yet, RPTs could be easily adapted for evaluating IDs.

5.3 Binary trees

5.3.1 Definitions and notation

A *binary tree* (BT) [26, 17, 19] is another tree-like structure representing a potential for a set of variables. Like NTs, this data structure can also encode CSIs and can be pruned when approximate solutions are required. In fact, a BT can be seen as a particularization of a NT, but in this case each internal node has always two outgoing arcs (or branches). More formally, a BT can be defined as follows.

Definition 30 (binary tree) *A BT defined over the set of variables* $\mathbf{X}_I$ *is a directed tree, where each internal node is labelled with a variable (random or decision), and each leaf node is labelled with a number (a probability or a utility value). We use* L_t *to denote the label of node* t*. Each internal node has always two children. We denote by* $L_{lb(t)}$ *and* $L_{rb(t)}$ *the labels (two proper disjoint subsets* *of Ω_{X_i}) of the left and right branches of node t. Then, we denote by t_l and t_r the two children of t (t_r for the right child and t_l for the left one).*

BTs can be used for representing any potential over a set of discrete variables involved in an IDs: a BT representing a PP $\phi(\mathbf{X}_I|\mathbf{X}_J)$ will be called *binary probability tree* and denoted $\mathcal{BT}^{\phi}(\mathbf{X}_I|\mathbf{X}_J)$. For example, the PP previously shown as a table in Example 20 and as NTs in Figure 5.1 is depicted as BT in Figure 5.4.

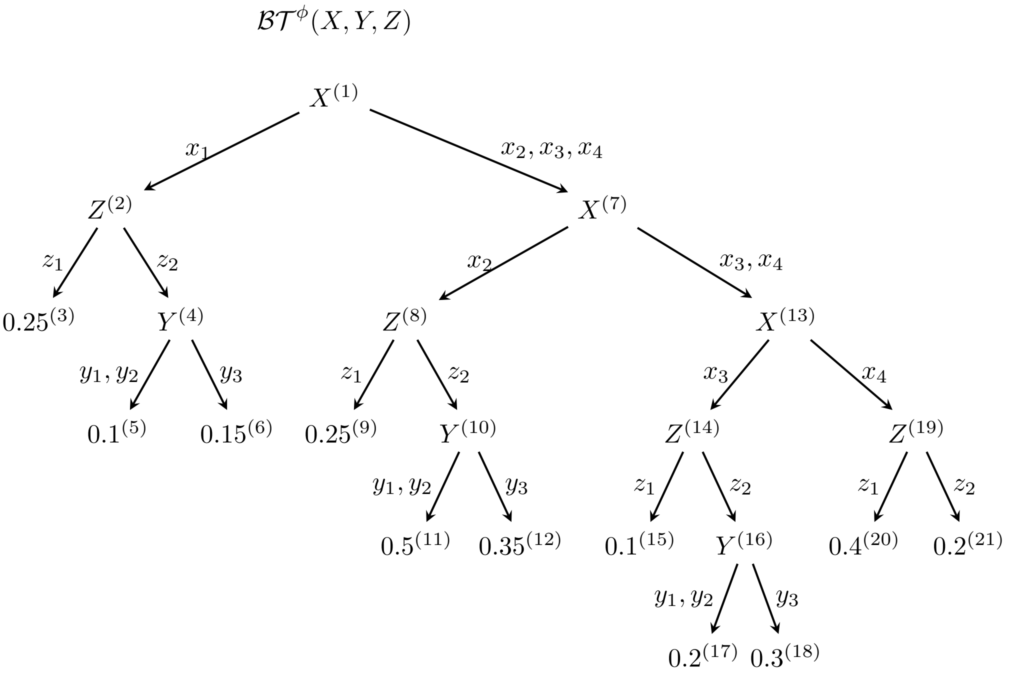


Figure 5.4: Exact BTs representing the potential $\phi(X|Y,Z)$ from Example 20. We use a superscript number at each node of $\mathcal{BT}^{\phi}(X,Y,Z)$, in order to easily identify them.

In the same way, a BT representing a UP $\psi(\mathbf{X}_I)$ will be called *binary utility tree* and denoted $\mathcal{BT}^{\psi}(\mathbf{X}_I)$. The UP previously shown as a table in Example 20 and as NTs in Figure 5.1 is depicted as BT in Figure 5.5.

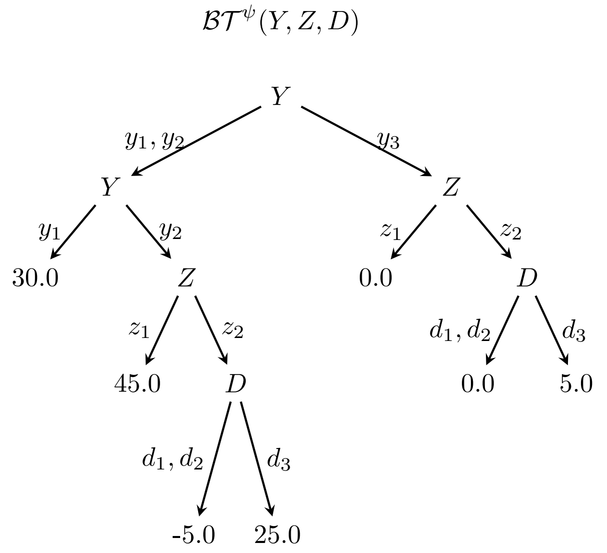


Figure 5.5: Exact BTs representing the UP $\psi(Y, Z, D)$ from Example 20.

In a BT, we can differ two types of nodes. First, we have the *internal nodes*, which are those labelled with a variable and having two children. Secondly, the leaves are those nodes labelled with a number and without any children. For example, in $\mathcal{BT}^{\phi}(X, Y, Z)$ shown in Figure 5.4, the node (2) is an internal one while the node (3) is a leaf. Note that the root will be considered as an internal node if there is more than a node in the BT.

The number of both internal and leaf nodes is the *size* of a given $\mathcal{BT}$ and denoted $size(\mathcal{BT})$. The number of its leaf nodes is denoted $leaves(\mathcal{BT})$. Theorem 2 states the relationship between the size of a tree and the number of leaves.

Theorem 2 *A binary tree $\mathcal{BT}$ with $n = size(\mathcal{BT})$ contains $\frac{n+1}{2}$ leaf nodes.*

Proof 2 *We will use induction on n to prove the previous theorem. When $n = 1$, there is $\frac{1+1}{2} = 1$ leaf which is the single node in the $\mathcal{BT}$.*

If we assume that a given $\mathcal{BT}$ has $n \geq 3$ nodes[2] *and $l = \frac{n+1}{2}$ leaves, the resulting*

[2] Any BT always contains an odd number of nodes.

tree of removing two leaves with a common parent[3] *in $\mathcal{BT}$ must have $n-2$ and $l-2+1$ leaves (the common node turns into a leave). Therefore the new tree with $n-2$ nodes must have $\frac{n+1}{2}-2+1=\frac{(n-2)+1}{2}$ leaves.* ■

As an example of previous theorem, in Figure 5.4, $\mathcal{BT}^{\phi}(X,Y,Z)$ has 21 nodes, being $\frac{21+1}{2}=11$ of them leaf nodes. On the other hand, the size of $\mathcal{BT}^{\psi}(Y,Z,D)$ is 13 and it has $\frac{13+1}{2}=7$ leaves.

As variables are not usually binary, a variable in a $\mathcal{BT}(\mathbf{X}_I)$ might appear more than once labelling the nodes in the path from the root to a given node t. The set of variables in such path is called the *ancestors of t* and denoted by $\mathbf{X}_I^t$. For example, the set of ancestors of node (3) in $\mathcal{BT}^{\phi}(X,Y,Z)$ is $\{X,Z\}$. Moreover, the union of the states labelling the outgoing branches of a given node is not always equal to the domain of the variable labelling such node. This is the case of node (13) in $\mathcal{BT}^{\phi}(X,Y,Z)$ in Figure 5.4, where only the states x_3 and x_4 appear in the outgoing branches. All the states of a given variable X_i that can label the outgoing branches of a node t conform its *set of available states* which is denoted by $\Omega_{X_i}^t$. The way for computing such set and an example are given in Definition 31 and in Example 21 respectively.

Definition 31 (set of available states calculation) *Let t be a node in a BT defined over the set of variables $\mathbf{X}_I$ and let X_i be any variable in $\mathbf{X}_I$. If $X_i\in\mathbf{X}_I^t$, then $\Omega_{X_i}^t$ is equal to the set of states labelling the outgoing branch of X_i in its last occurrence in the path from the root to t. Otherwise, $\Omega_{X_i}^t$ is equal to Ω_{X_i}.*

Example 21 *Let as consider the $\mathcal{BT}^{\phi}(X,Y,Z)$ show Figure 5.4. It is trivial that for the root node, as it has not ancestors, the set of available states of any variable is equal to their domains, i.e. if $t=(1)$ then it holds that $\Omega_X^t=\Omega_X$, $\Omega_Y^t=\Omega_Y$, $\Omega_Z^t=\Omega_Z$. By contrast, for the node (7) the set of available states of X is $\{x_2,x_3,x_4\}$, which is the label of the outgoing branch of X in its last occurrence, i.e. node (1).*

[3] Internal nodes in a BT have exactly 2 children.

A binary tree $\mathcal{BT}$ on the set of variables $\mathbf{X}_I = \{X_i | i \in I\}$ represents a potential (no matter whether PP or UP) $\psi : \Omega_{\mathbf{X}_I} \rightarrow \mathbb{R}$ if for each $\mathbf{x}_I \in \Omega_{\mathbf{X}_I}$ the value $\psi(\mathbf{x}_I)$ is the number obtained with the recursive Algorithm 7. The procedure starts from the root node t of the given $\mathcal{BT}$, traversing recursively the full structure to the leaves. Depending on the kind of node, the algorithm applies a different action. If t is a leaf node, the corresponding value is returned (lines 2 and 3); otherwise t is an internal node, then the procedure is recursively called to the child with the given configuration (lines 4 to 11).

Algorithm 7 getValue

input : t (node of a $\mathcal{BT}$ defined on $\mathbf{X}_I$), $\mathbf{x}_I$ (a configuration)

output : $\mathcal{BT}(\mathbf{x}_I)$

```
1:  Let $val$ be a new numerical variable
2:  if $t$ is a leaf node then
3:      $val \leftarrow L_t$
4:  else
5:      Let $X_i$ be the variable labelling $t$
6:      Let $x_i$ the state of $X_i$ consistent with $\mathbf{x}_I$
7:      if $x_i \in L_{lb(t)}$ then
8:          $val \leftarrow$ getValue($t_l$, $\mathbf{x}_I$)
9:      else if $x_i \in L_{lr(t)}$ then
10:         $val \leftarrow$ getValue($t_r$, $\mathbf{x}_I$)
11:     end if
12: end if
13: return $val$
```

Example 22 *Let us consider the binary tree $\mathcal{BT}^{\phi}(X, Y, Z)$ shown in Figure 5.4. We want to obtain the value associated to the configuration $\{b_4, a_1\}$, then we invoke Algorithm 7, i.e.* $\text{getValue}((1), \{x_2, y_1, z_1\})$. *The recursive calls are explained bellow.*

- $\text{getValue}((1), \{x_2, y_1, z_1\})$*: node* (1) *is an internal node labelled with the variable X so the algorithm will continue traversing the BT. The label of the right outgoing arc contains the state x_2, then the algorithm is now recursively invoke on the right child (lines 9 to 10).*
- $\text{getValue}((7), \{x_2, y_1, z_1\})$*: again, the algorithm has reached a node which is not a leaf, so the algorithm is recursively called on the left child which is labelled with x_2 (lines 7 to 8).*
- $\text{getValue}((7), \{x_2, y_1, z_1\})$*: node* (8) *is an internal node labelled with the variable Z. As the left outgoing arc contains the state Z_1, the algorithm is called on the left child (lines 7 to 8).*
- $\text{getValue}((8), \{x_2, y_1, z_1\})$ *the algorithm has reached a leaf node and the value* 0.5 *is assigned to the numerical variable val (lines 2 to 3) and returned (line 13).*

Finally, each of the recursive calls in the stack also assign the returned value to the numerical variable val.

5.3.2 Extended configuration

As explained in Section 4.3, given a set of variables $\mathbf{X}_I$, the elements of $\Omega_{\mathbf{X}_I}$ are called configurations and denoted $\mathbf{x}_I$. A configuration can also be seen as a mapping assigning to each $X_i \in \mathbf{X}_I$ a state $x_i \in \Omega_{X_i}$. In a table representing a potential, each entry corresponds to only one configuration. However, this is not the case of BTs, where each leaf may correspond to a set of configuration. For that reason, it is necessary to generalize the concept of configuration and hence we can define an *extended configuration* as follows.

Definition 32 (extended configuration) *An extended configuration for a set of variables $\mathbf{X}_I$, denoted by $\mathbf{A}_{\mathbf{X}_I}$ or simply $\mathbf{A}$, is defined as the multi-set $\{A_i \subseteq \Omega_{X_i} | X_i \in \mathbf{X}_I\}$.*

An extended configuration can also be seen as a mapping assigning to each $X_i \in \mathbf{X}_I$ a subset $A_i \subseteq \Omega_{X_i}$. Note that a (usual) configuration can also be seen as an extended configuration that assigns subsets with exactly one element. The intuition behind this new concept, is that an extended configuration for a set of variables defines a set of configurations for that set of variables. That is, the set of configurations defined by $\mathbf{A}_{\mathbf{X}_I}$ is denoted by $S_{\mathbf{A}_{\mathbf{X_I}}}$ and it is obtained with the Cartesian product of the subsets of sates in $\mathbf{A}_{\mathbf{X}_I}$.

Example 23 *Let us consider the set of variables $\mathbf{X}_I = \{A, B\}$ whose domains are $\Omega_A = \{a_1, a_2, a_3\}$ and $\Omega_B = \{b_1, b_2, b_3, b_4\}$ respectively. Then, $\mathbf{x}_I = \{a_3, b_1\}$ is a configuration and $\mathbf{A}_{\mathbf{X}_I} = \{\{a_3\}, \{b_1, b_2\}\}$ is an extended configuration which defines the set of configurations $S_{A_{\mathbf{X}_I}} = \{\{a_3, b_1\}, \{a_3, b_2\}\}$.*

The *associated extended configuration* for t, denoted by $\mathbf{A}^t$, is the multi-set $\{\Omega^t_{X_i} | X_i \in \mathbf{X}_I\}$. For example, let us consider the binary tree $\mathcal{BT}^{\phi}(X, Y, Z)$ shown in Figure 5.4, the associated extended configuration for the node (13) is $\{\{x_3, x_4\}, \{y_1, y_2, y_3\}, \{z_1, z_2\}\}$. We denote by $\psi^{R(\mathbf{A}^t)}$ the potential consistent with $\mathbf{A}^t$; it corresponds to the sub-tree where t is the root node.

5.3.3 Independencies encoded with BTs

Like NTs, the advantage of using BTs instead of tables for representing potentials is that the storage requirements are reduced. This tree-like structure is not an exhaustive representation since identical values can be grouped into a single one, i.e. a leaf node might corresponds to more than one configuration. The presence of similar values in a PP is usually possible due to the presence of independencies between the variables in the ID. The advantage of representing potentials as trees is that they allow encoding independencies that are more general[4], i.e. more fine-

[4] In Section 3.3.4 we described the following forms of independence: CSIs, PCIs and CWIs.

grained, than those structurally encoded by IDs, i.e. *conditional independencies (CIs)* (see Sections 3.3.3 and 4.4). Unlike CIs, these fine-grained independencies hold if we only consider some states in the variables domains.

NTs, and by extension BTs, can encode *context-specific independencies (CSIs)* (see Section 3.3.4.1). In short, CSIs are a kind of independence that only hold for certain contexts, i.e. given a specific assignment of values to some variables. As an example, let us consider the PP $\phi(X|Y,Z)$ represented as a table as follows.

$$\mathcal{T}^{\phi}(X,Y,Z) = \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & & \\ 0.25 & 0.25 & 0.1 & 0.4 & y_1 & \\ 0.25 & 0.25 & 0.1 & 0.4 & y_2 & z_1 \\ 0.25 & 0.25 & 0.1 & 0.4 & y_3 & \\ 0.1 & 0.5 & 0.2 & 0.2 & y_1 & \\ 0.1 & 0.5 & 0.2 & 0.2 & y_2 & z_2 \\ 0.15 & 0.35 & 0.3 & 0.2 & y_3 & \end{array}$$

In previous PP, X and Y are contextually independent given $Z = z_1$, i.e. $I(X \perp Y | Z = z_1)$. This independence implies that, for each $x \in \Omega_X$, if $Z = z_1$ the value of the probability is the same regardless of the state of Y. When representing this PP as a tree (NT or BT) the leaf nodes related to this independence will be collapsed into a single one. Figure 5.6 depicts previous PP as a NT and as a BT. All the leaf nodes that represent more than one configuration due to this CSI are highlighted with a dashed box. This is the case of the left-most leaf node in $\mathcal{NT}^{\phi}(X,Y,Z)$ and in $\mathcal{BT}^{\phi}(X,Y,Z)$, which correspond with the configurations defined by the extended configuration $\{\{x_1\},\{y_1,y_2,y_3\},\{z_1\}\}$.

Another kind of fine-grained independence is *partial conditional independence (PCI)*, which is basically a CSI that holds if certain subsets of the variable domains are considered. If we consider again the previous PP, we can observe that X and Y are partially conditional independent in the domain $\{y_1, y_2\}$ and given the context $Z = z_2$, i.e. $I(X \perp Y | \{y_1, y_2\}, z_2)$. NTs cannot take advantage of PCIs as, by definition, any internal node in a NT contains an outgoing branch for each of the states in the domain of the variable labelling such node. This is not the case for BTs, where outgoing branches might be labelled with a subset of the variable

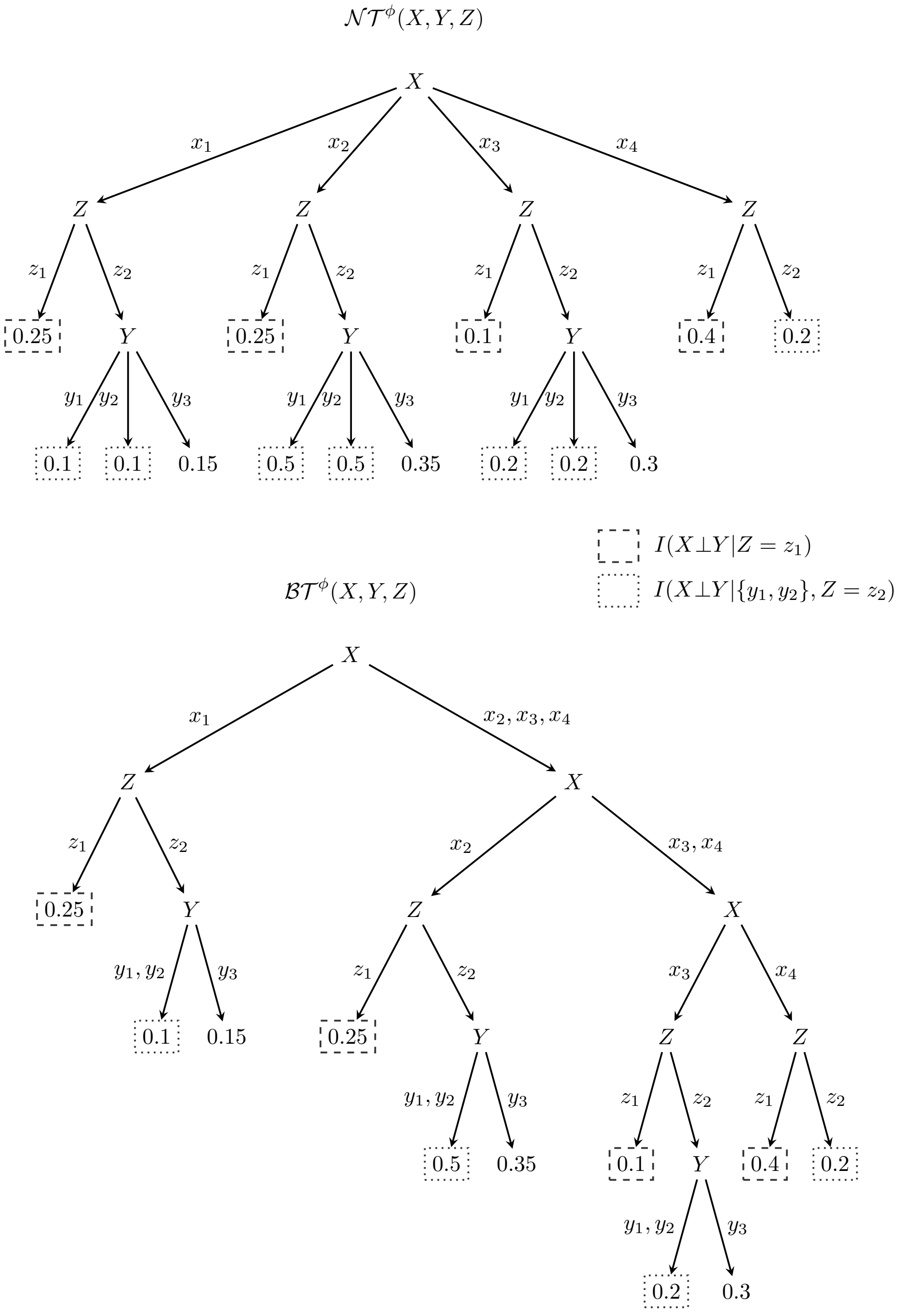


Figure 5.6: Leaf nodes related to the CSIs and PCIs present in $\phi(X|Y,Z)$ when such PP is represented as a NT and as BT.

domain. In Figure 5.6, all the leaf nodes related to this PCI are highlighted with a dotted pattern. For example, let us consider the configurations $\{x_2, y_2, z_2\}$ and $\{x_2, y_3, z_2\}$. In $\mathcal{NT}^{\phi}(X, Y, Z)$ two equal leaf nodes are required. By contrast, in $\mathcal{BT}^{\phi}(X, Y, Z)$ these two configurations correspond to only one leaf node.

The structural capacity of encoding CSIs and PCIs makes the representation with BTs more compact. In the example, a table representing $\phi(X, Y, Z)$ requires 24 entries. By contrast, $\mathcal{NT}^{\phi}(X, Y, Z)$ requires 22 nodes (14 of them are leaves). Similarly, $\mathcal{BT}^{\phi}(X, Y, Z)$ requires 20 nodes (11 of them are leaves).

Another form of independence generalizing CSI is *contextual weak independence (CWI)* (see Section 3.3.4.3). Unlike CSIs, in this case the sub-domains of the variables in which the independence holds is not explicitly defined by a particular assignment. For example, in $\phi(D|B, A)$, depicted as a table below, D and B are weakly independent given the context $A = a_1$, i.e. it holds that $WI(D \perp B|\ A = a_1)$[5]

$$\mathcal{T}^{\phi}(D,B,A) = \begin{array}{rrrrcc} d_1 & d_2 & d_3 & d_4 & & \\ 0.3 & 0.7 & 0 & 0 & b_1 & \\ 0.3 & 0.7 & 0 & 0 & b_2 & a_1 \\ 0 & 0 & 0.1 & 0.9 & b_3 & \\ 0 & 0 & 0.1 & 0.9 & b_4 & \\ 0.6 & 0.4 & 0 & 0 & b_1 & \\ 0.8 & 0.2 & 0 & 0 & b_2 & a_2 \\ 0 & 0 & 0.2 & 0.8 & b_3 & \\ 0 & 0 & 0.3 & 0.7 & b_4 & \end{array}$$

Like PCIs, these independencies can be exploited by BTs but not by NTs. Figure 5.7 shows a NT and a BT representing $\phi(D|B, A)$ from Example 7. All the leaf nodes related to this CWI are highlighted with a dash-dot pattern.

[5] The independence holds in the domains defined by the equivalence classes $\pi_1 = \{(d_1, b_1, a_1), (d_2, b_1, a_1), (d_1, b_2, a_1), (d_2, b_2, a_1)\}$ and $\pi_2 = \{(d_3, b_3, a_1), (d_4, b_3, a_1), (d_3, b_4, a_1), (d_4, b_4, a_1)\}$ from the equivalence relation $\theta_1(DA = a_1) \circ \theta(BA = a_1)$.

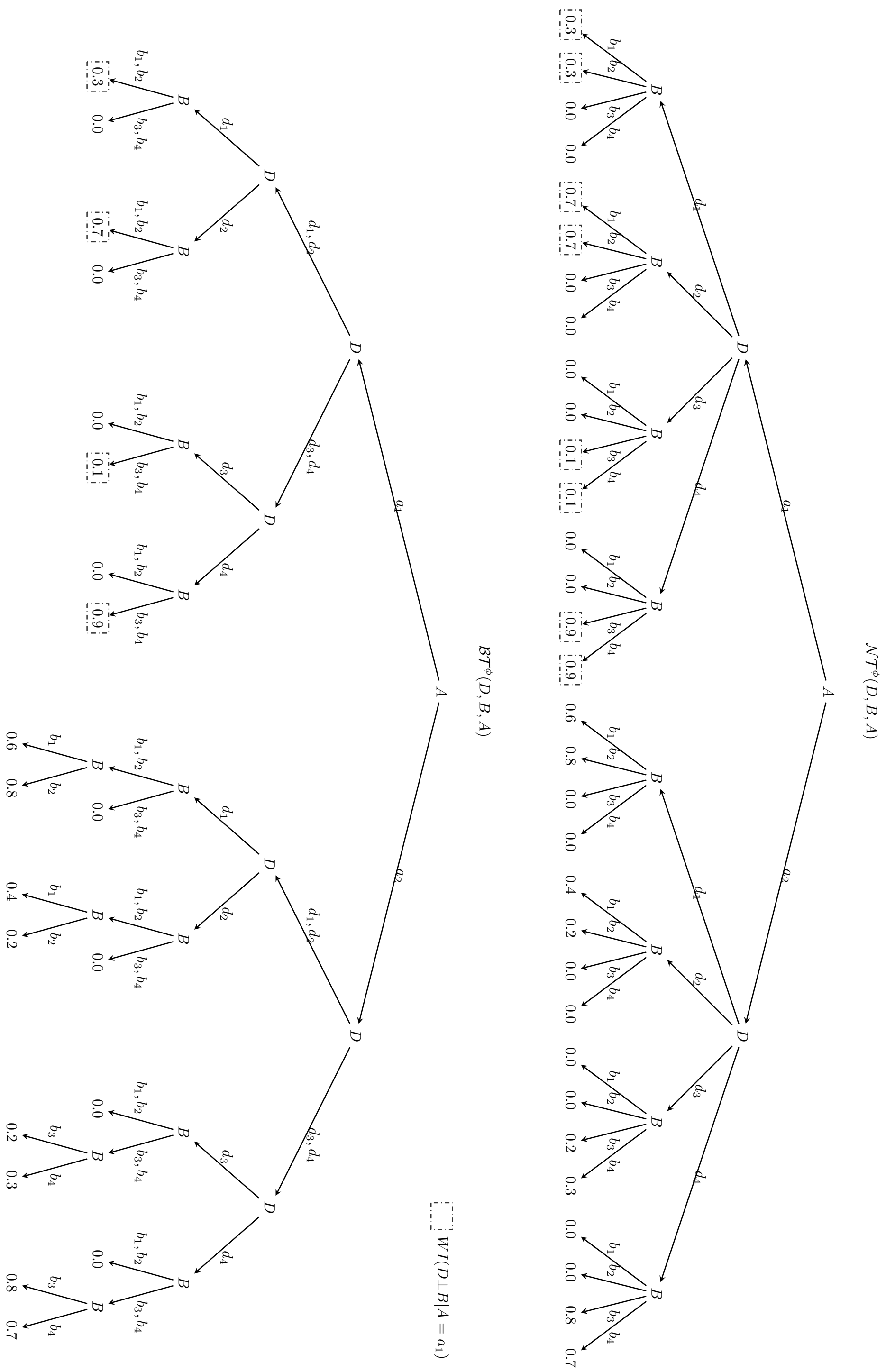


Figure 5.7: Leaf nodes related to the CWI present in $\phi(D|B, A)$ when such PP is represented as a NT and as BT.

In addition, if we change the order of appearance of the variables in $\mathcal{BT}^{\phi}(D, B, A)$ we obtain a more compact representation. The result is shown in Figure 5.8.

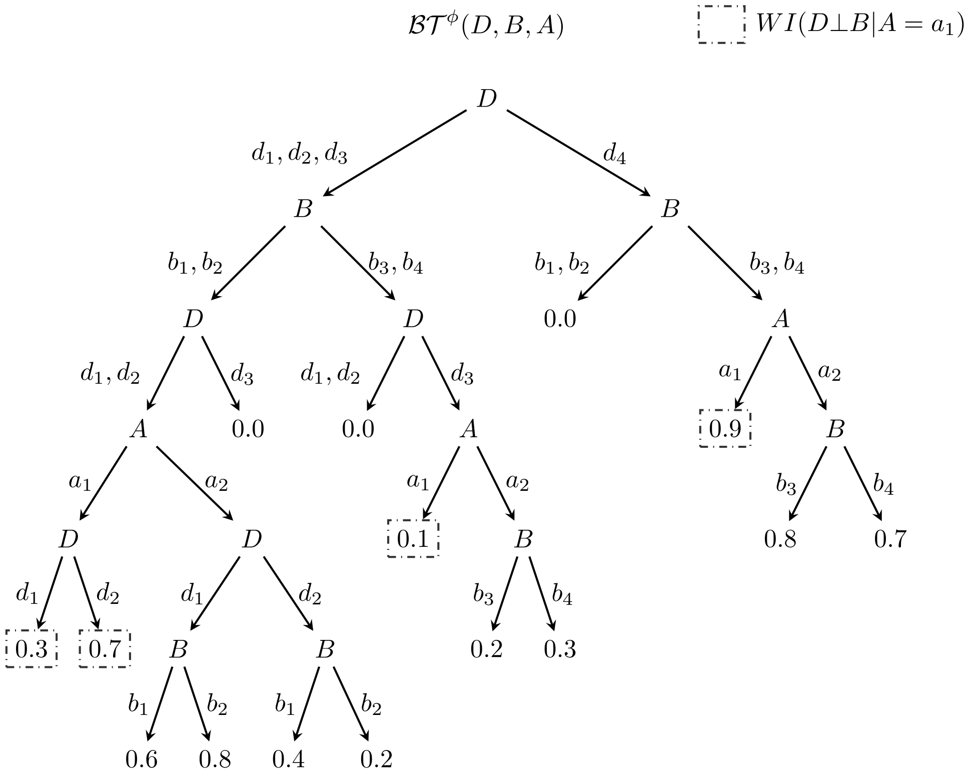


Figure 5.8: A more compact representation of the potential shown in 5.7 as a BT and obtained by reordering its nodes.

It can be observed that the representation of $\phi(D|B, A)$ as a BT reduces the size of the potential as well: $\mathcal{T}^{\phi}(D, B, A)$ requires 32 table entries, while $\mathcal{BT}^{\phi}(D, B, A)$ contains 29 nodes (15 of them are leaves).

When we use the term of independence, we actually refer to a probabilistic independence (e.g. the probability of X is independent of Y). In practice this means that a given PP takes the same value for a subset of configurations. A similar situation can be found in the UPs, but in this case we will talk about functional independencies. As happens when representing PPs, a BT is a more compact representation for a UP than a NT: many of the identical values that cannot be grouped into a single value in a NT, can actually be grouped when using a BT.

As an example, Figure 5.9 shows a comparison of a NT and a BT representing the potential $\psi(Y, Z, D)$. Note that now the size is reduced to 14 and 13 nodes (9 and 7 of them are leaves).

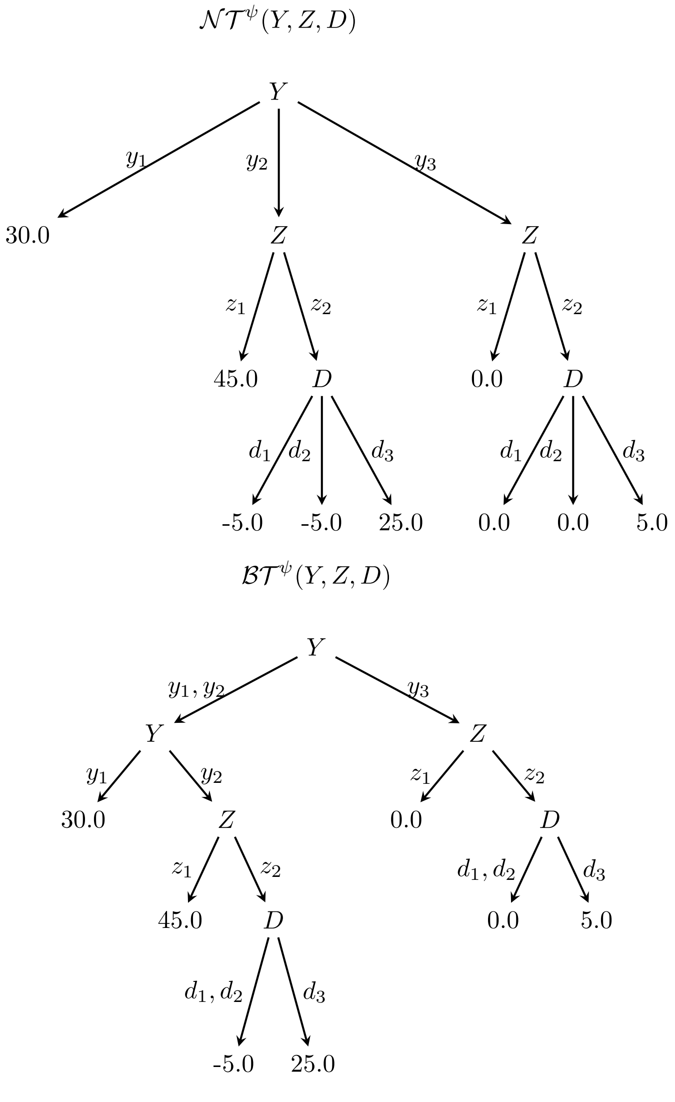


Figure 5.9: A NT and a BT representing the potential $\psi(Y, Z, D)$.

Another advantage of BTs (also present in NTs) is that they can be pruned in order to reduce even more their sizes and the required execution time in propagation algorithms. In the following section, we propose an algorithm for building (i.e. learning) and pruning BTs.

5.4 Learning exact and approximate BTs

5.4.1 Building binary trees

The method for building a BT (from a table) representing a PP was described in a previous work [26] while the equivalent one for UPs was described in [17, 19]. Such methods are inspired by the algorithms for learning classification trees from a set of examples [95]. It must be noticed that, in general, many BTs can be built to represent the same potential. However, we are interested in finding the smaller one. As an example, let us consider the following UP:

$$\mathcal{T}^{\psi}(A,B) = \begin{array}{cccc|c} b_1 & b_2 & b_3 & b_4 & \\ 30 & 30 & 30 & 30 & a_1 \\ 15 & 15 & 20 & 20 & a_2 \\ 25 & 25 & 25 & 25 & a_3 \end{array}$$

Figure 5.10: Example of a UP represented as a table.

This potential can be represented with many BTs of different sizes. Some of them are depicted in Figure 5.11. Note that these BTs contain 7, 9 and 11 nodes respectively.

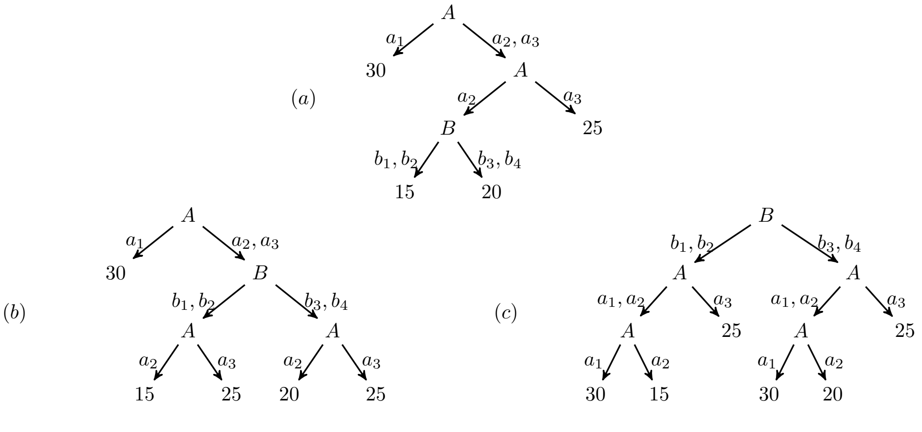


Figure 5.11: Three BTs of different sizes representing the potential $\psi(A,B)$

The task of building a BT from a table can be seen as an optimization problem interested in choosing the labels for internal nodes (variables) and branches (states). This requires a heuristic procedure for ordering the variables according to their information. The most informative ones will be located at the highest nodes of the tree. The motivation for such ordering is to obtain neighbor leaf nodes as similar as possible. This condition will minimize the error produced when pruning the trees collapsing several leaves into a single one.

The proposed algorithm builds a BT using a top-down approach, choosing at each iteration a variable and a partition of its states. The general scheme for building a BT representing a potential ψ defined over the set of variables $\mathbf{X}_I$ is detailled in Algorithm 8. Note that this procedure is almost the same for building trees representing PPs or UPs. The kind of potential only affects to the *spliting criteria* used for selecting the next node and states for expanding the tree (step 2.b).

Algorithm 8 Build an exact BT

```
1. Build an initial $\mathcal{BT}$ with a single node labelled with the average of the values in the potential: $L_t = \sum_{\mathbf{x}_I \in \Omega_{\mathbf{X_I}}} \psi(\mathbf{x}_I)/|\Omega_{\mathbf{x}_I}|$
2. While $\mathcal{BT}$ contains any leaf node $t$ that can be expanded:
    (a) Select a leaf node $t$.
    (b) According to any criteria (see Section 5.4.1.1), select a variable $X_i$ and a partition of its available states at $t$ into two subsets, $\Omega_{X_i}^{t_l}$ and $\Omega_{X_i}^{t_r}$.
    (c) Expand the leaf node $t$ (with $t$ rooting the terminal tree and being $t_l$ and $t_r$ its children).
    (d) Label $t$ with $X_i$ and the two outgoing arcs with $\Omega_{X_i}^{t_l}$ and $\Omega_{X_i}^{t_r}$. The two leaf nodes $t_l$ and $t_r$ will be labelled with the average of values consistent with the states labelling the path from the root to $t_l$ and $t_r$ respectively.
3. Return $\mathcal{BT}$
```

The process begins with an initial $\mathcal{BT}_0$ which has only one node labelled with the average of the values in the potential (step 1). Then, a greedy algorithm (step

2) is applied until an exact tree is obtained (there is not any leaf node that can be expanded). At each iteration, a new $\mathcal{BT}_{j+1}$ is generated from the previous one, $\mathcal{BT}_j$. This new tree is the result of expanding one of the leaf nodes t in $\mathcal{BT}_j$ with a terminal tree (with t rooting the terminal tree and being t_l and t_r its children). The label of the node t will be replaced with one of the *candidate variables*. A variable X_i is candidate if the associated extended configuration $\mathbf{A}^t$ contains more than one state for X_i. The set of available states $\Omega^t_{X_i}$ of the chosen candidate variable X_i will be partitioned into two subsets, $\Omega^{t_l}_{X_i}$ and $\Omega^{t_r}_{X_i}$. Each subset labels one of the two outgoing arcs (left and right) of t. The two leaf nodes t_l and t_r in the new terminal tree will be labelled with the average of values consistent with the states labelling the path from the root to t_l and t_r respectively.

5.4.1.1 Splitting criteria

The candidate variable and the partition of its states must be chosen using any criteria or heuristic (step 2.b in Algorithm 8). Any partition of $\Omega^t_{X_i}$ would be possible, but checking all of them would be a very time-consuming task. In order to reduce the complexity, we assume that the set of available states for X_i at node t is ordered and we only check partitions into subsets with consecutive states. For example, given a variable X with $\Omega^t_X = \{x_1, x_2, x_3\}$, we will only check the partitions $\{\{x_1\}, \{x_2, x_3\}\}$ and $\{\{x_1, x_2\}, \{x_3\}\}$. In our approach we propose choosing the variable and partition that maximizes the *information gain* that can be defined as follows.

Definition 33 (information gain) *Let ψ be the potential (no matter whether PP or UP) to be represented as a tree, $\mathcal{BT}_j$ and $\mathcal{BT}_{j+1}(t, X_i, \Omega^{t_r}_{X_i}, \Omega^{t_l}_{X_i})$ the resulting tree of expanding the leaf node t with the candidate variable X_i and a partition of its available states into sets $\Omega^{t_l}_{X_i}$ and $\Omega^{t_r}_{X_i}$. Let $D(\psi, \mathcal{BT}_j)$ be the distance between a potential and a tree. The information gain can be defined as:*

$$I(t, X_i, \Omega^{t_l}_{X_i}, \Omega^{t_r}_{X_i}) = D(\psi, \mathcal{BT}_j) - D(\psi, \mathcal{BT}_{j+1}(t, X_i, \Omega^{t_l}_{X_i}, \Omega^{t_l}_{X_i})) \qquad (5.1)$$

That is, the information gain is the difference between the distance from the current tree to the real potential before and after expanding the leaf node labelled with X_i. Thus, we need to use a distance or a similarity measure between an exact potential and a tree. When building a BT representing a PP, Kullback Leibler divergence will be used (Equation (5.2)). In the case of utilities we propose to use the Euclidean distance (Equation (5.3)).

$$D_{KL}(\phi, \mathcal{BT}) = \sum_{\mathbf{x}_I \in \Omega_{\mathbf{X}_I}} \psi(\mathbf{x}_I) \cdot log \frac{\psi(\mathbf{x}_I)}{\mathcal{BT}(\mathbf{x}_I)} \tag{5.2}$$

$$D_{EU}(\psi, \mathcal{BT}) = \sqrt{\sum_{\mathbf{x}_I \in \Omega_{\mathbf{X}_I}} \left(\psi(\mathbf{x}_I) - \mathcal{BT}(\mathbf{x}_I)\right)^2} \tag{5.3}$$

where $\mathcal{BT}(\mathbf{x}_I)$ is the value that the tree assigns to the configuration $\mathbf{x}_I$. Now we will denote the information gain by I_{KL} or I_{EU} depending on distance used.

Example 24 *In order to illustrate the process for building a BT, let us consider the potential shown in Figure 5.10. The intermediate trees obtained during the building process of this tree are shown in Figure 5.12.*

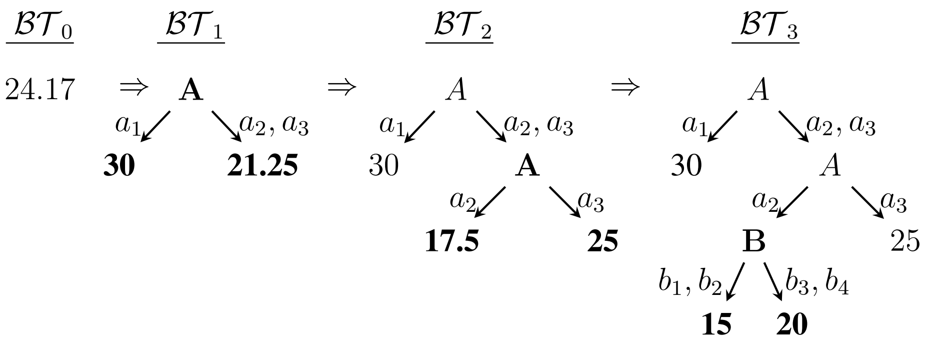


Figure 5.12: Process for building a BT from the potential in Figure 5.10

The initial tree $\mathcal{BT}_0$ *contains a single node labelled with the value* 24.17*, which is approximately the average of all the values in the potential. Then, the information gain for each candidate variable and partition is calculated as follows:*

$$I_{EU}(t, A, \{a_1\}, \{a_2, a_3\}) \simeq 18.484 - 11.726 = 6.758$$

$$I_{EU}(t, A, \{a_1, a_2\}, \{a_3\}) \simeq 18.484 - 18.371 = 0.113$$

$$I_{EU}(t, B, \{b_1\}, \{b_2, b_3, b_4\}) \simeq 18.484 - 18.409 = 0.075$$

$$I_{EU}(t, B, \{b_1, b_2\}, \{b_3, b_4\}) \simeq 18.484 - 18.257 = 0.227$$

$$I_{EU}(t, B, \{b_1, b_2, b_3, \}, \{b_4\}) \simeq 18.484 - 18.409 = 0.075$$

The highest value for the information gain is obtained if the variable A and the partition $\{\{a_1\}, \{a_2, a_3\}\}$ are chosen to expand the node. After that, the algorithm repeats the same process until the exact BT is obtained. The node labelled with 30 *at $\mathcal{BT}_1$ will not be expanded any more: all configurations in the utility potential consistent with $A = a_1$ are equal to 30.*

5.4.1.2 Efficient computation of the information gain

Some of the computations performed to calculate the information gain when a node is expanded are also performed in posterior iterations when the children are expanded. Proposition 6 shows an alternative expression for computing the information gain that takes advantage of these repeated computations.

Proposition 6 *Suppose that we aim to represent a potential as a BT. Let $\mathcal{BT}_j$ be an intermediate tree in the building process and $\mathcal{BT}_j(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r})$ the tree resulting of expanding the leaf node t with the candidate variable X_i and a partition of its available states into sets $\Omega_{X_i}^{t_l}$ and $\Omega_{X_i}^{t_r}$. Let $sum(\bullet)$ be the addition of all the values in a given potential and let $sumSqr(\bullet)$ be the addition of all its square values. If we aim to build a tree representing a PP, say ϕ, then the information gain (Equation (5.1)) using the Kullback Leibler divergence can be calculated in the following way.*

$$\begin{aligned}
I_{KL}(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) = & sum(\phi^{R(\boldsymbol{A}^t)}) \cdot \log(\left|\Omega_{X_i}^t\right| / sum(\phi^{R(\boldsymbol{A}^t)})) \\
& + sum(\phi^{R(\boldsymbol{A}^{t_l})}) \cdot \log(sum(\phi^{R(\boldsymbol{A}^{t_l})}) / \left|\Omega_{X_i}^{t_l}\right|) \\
& + sum(\phi^{R(\boldsymbol{A}^{t_r})}) \cdot \log(sum(\phi^{R(\boldsymbol{A}^{t_r})}) / \left|\Omega_{X_i}^{t_r}\right|)
\end{aligned} \tag{5.4}$$

Otherwise, if we aim to build a tree representing a UP, say ψ, then the information gain using the Euclidean distances will be computed as follows.

$$\begin{aligned}
I_{EU}(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) = & \sqrt{sumSqr(\psi^{R(\boldsymbol{A}^t)}) - \tfrac{(sum(\psi^{R(\boldsymbol{A}^t)}))^2}{size(\psi^{R(\boldsymbol{A}^t)})}} \\
- & \sqrt{sumSqr(\psi^{R(\boldsymbol{A}^{t_l})}) - \tfrac{(sum(\psi^{R(\boldsymbol{A}^{t_l})}))^2}{size(\psi^{R(\boldsymbol{A}^{t_l})})} + sumSqr(\psi^{R(\boldsymbol{A}^{t_r})}) - \tfrac{(sum(\psi^{R(\boldsymbol{A}^{t_r})}))^2}{size(\psi^{R(\boldsymbol{A}^{t_r})})}}
\end{aligned} \tag{5.5}$$

It should be noticed that computing the information gain with previous expressions is more efficient than using Equation (5.1), since the first term can be obtained from the information gain computed when the father node was expanded. In Appendix A, it is proved that the expressions (5.4) and (5.5) for computing the information gain are equivalent to Equation (5.1).

5.4.2 Pruning Binary Trees

Sometimes the size of a BT can be too large making infeasible its management during the propagation. In that case, it can be pruned in order to get an approximate one. Pruning a BT consists of replacing a terminal tree[6] by the average value

[6] We say that a part of a BT is a terminal tree if it contains only one node labeled with a variable, and its two children are leaves.

of its leaves. Figure 5.13 shows an example of pruning, where the node consistent with the configuration $A = \{a_2\}$ in the left tree has been replaced by the average value of its children in the right tree.

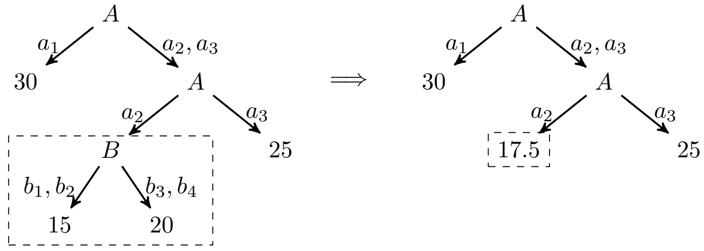


Figure 5.13: Example of pruning a terminal tree in a BT

We apply the following heuristic procedure to decide when to prune a terminal tree. This procedure is guided by a threshold ε.

Definition 34 (pruning condition of a terminal tree) *Let $\mathcal{BT}$ be a binary tree encoding a potential, t the root of a terminal tree of $\mathcal{BT}$ labelled with X_i, t_l and t_r its children, $\Omega_{X_i}^{t_l}$ and $\Omega_{X_i}^{t_r}$ the sets of states for the left and right child respectively, and ε a given threshold such that $\varepsilon \geq 0$. Then we will consider the following conditions to decide whether the terminal tree can be pruned or not:*

- *If $\mathcal{BT}$ represents a UP ψ, the terminal tree can be pruned if:*

$$I_{EU}(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) \leq \varepsilon \cdot (max(\psi) - min(\psi)) \tag{5.6}$$

  where $max(\psi)$ and $min(\psi)$ are the maximum and minimum values in ψ.

- *Alternativelly, if $\mathcal{BT}$ represents a PP, the terminal tree can be pruned if:*

$$I_{KL}(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) \leq \varepsilon \tag{5.7}$$

In previous definition, it should be noted that I_{EU} and I_{KL} can be locally computed using Equations (5.5) or (5.4) respectively. The goal of pruning involves detecting leaves that can be replaced by one value without a great increment in the distance between $\mathcal{BT}$ and the exact potential. Here, I_{EU} and I_{KL} are considered as the *information loss* produced in a BT when a terminal tree is removed. The pruning process would finish when there are not any other terminal tree in $\mathcal{BT}$ verifying the condition in Equations (5.6) or (5.7). In case of trees representing UPs, which are not normalized, the highest information loss allowed when pruning depends on a threshold $\varepsilon \geq 0$ but also on the maximum and minimum values in ψ. Low values of ε will produce large trees with low errors while low values of ε will lead to small trees and big errors. If $\varepsilon = 0$, there is not approximation: a terminal tree will be pruned only if both children have the same value.

5.5 Conclusions

In this chapter we have formalized the use of BTs for representing the potentials involved in IDs. This kind of tree allows representing context-specific independencies that are more fine-grained compared to those encoded using NTs or tables. In particular, some of these forms of independence are partial conditional independencies and contextual weak independencies. As a result, this tree-like structure allows a compact representation of the potentials: many equal values can be represented with the same leaf node. Additionally, BTs can be pruned leading to approximate versions of the potentials. In particular, we have provided detailed methods for building and pruning a BTs representing PPs and UPs.

In Chapter 8, we will explain how IDs whose potentials are represented as BTs can be evaluated. Another advantage of BTs is that they can be used for representing the qualitative information in a decision problem due to asymmetries. This capability is detailed in Chapter 6.

Chapter 6

Asymmetries Representation with Binary Trees

6.1 Introduction

In some decision problems, particular acts or events can lead to very different possibilities and, as a consequence, not all decisions and variables are considered in all circumstances. Such kind of problems are so-called asymmetric [108], and an important drawback of IDs is related to their inability for efficiently representing them. To be modelled as an ID, asymmetric decision problems must be symmetrized by adding artificial states to the domains of some variables. Even more, potentials might contain some impossible configurations (due to the asymmetries of the problem) that are not relevant for computing the optimal policies. However, in conventional evaluation algorithms, operations on potentials are performed with all their values. This implies that a considerable amount of unnecessary memory space and computation may be involved during the evaluation of IDs representing asymmetric decision problems.

Several approaches have been made to solve this drawback. Call and Miller [21], Fung and Shachter [104], Smith et al. [108], Qi et al. [94], Covaliu and Oliver [36], Shenoy [107], Nielsen and Jensen [84], Demirer and Shenoy [43], Díez and Luque [44] have proposed different frameworks for solving asymmetric decision problems. Some of these approaches imply important changes in the ID framework and require new specific evaluation algorithms. Our approach, however, tries to keep the framework without changes, as well as using the standard algorithms as much as possible. In particular, we propose the use of BTs in order to improve the efficiency of the evaluation of IDs representing asymmetric decision problems. Our proposal consists of using BTs for representing potentials and asymmetries. As the same data structure is used for both, potentials and asymmetries, they can be easily applied in order to reduce the number of scenarios to consider.

In Section 6.2, we first explain the problematic related to the evaluation of IDs representing problems with asymmetries. Then, in Section 6.3.1 we introduce *constraints rules*, which are logical expressions for defining asymmetries in a more intuitive way. The basic concepts about the representation of asymmetries as BTs, namely *binary constraints trees*, are described in Section 6.3.2.

6.2 Motivation

6.2.1 Asymmetric decision problems

In asymmetric decision problems [108], the possible outcomes and decision options for some variables may depend on the past. For example, let us consider a decision problem where you have the option of doing a test. After that decision, any possible result for this test is meaningless if you have decided not doing it. In general, in asymmetric decision problems, we will find conditioned scenarios such as "if X takes the value x_1, then Y cannot take y_2".

There are several possible sources of asymmetries in decision problems. Though there are different classifications in the literature, we will follow the one proposed by Jensen and Nielsen [64]:

- *Functional asymmetries*: the domain of a variable (chance or decision) may vary depending on the value taken by another variable.

- *Structural asymmetries*: the very occurrence of an observation or a decision depends on the past.

- *Order asymmetries*: several orderings of the decisions are possible.

In our approach we will only consider those asymmetries where the domain of a variable is partially or totally restricted, (i.e., functional and structural asymmetries).

For a better comprehension of asymmetric decision problems we will introduce the so-called reactor problem, which was initially described by Covaliu and Oliver [36] and modified by Bielza and Shenoy [3, 4]. Example 25 gives the details of the modified version of this problem.

Example 25 (the reactor problem) *An electric company must decide about the type of reactor to build* (D_2)*. There are three alternatives: a reactor of conventional design* (c)*, a reactor of advanced design* (a)*, or neither* (n)*. The conventional one is less profitable but safer: such kind of reactor* (C) *has a probability* 0.02 *of failing* (cf) *while a probability of* 0.98 *of being success* (cs)*. By contrast, an advanced reactor* (A) *has probability* 0.096 *of a major accident* (am)*, probability* 0.244 *of a limited accident* (al) *and probability* 0.66 *of being success* (as)*. The profits for the conventional reactor are* $\$8B$ *in case of being success while* $-\$4B$ *if there is a failure. For the advanced one, profits are* $\$12B$ *in case of being success,* $-\$6B$ *if there is a limited accident, and* $-\$10B$ *if there is a major accident. Before taking the decision, the company can make a test* (D_1) *for analysing the components of the advanced reactor that costs* $\$1B$*. Thus, it should be decided if the test is made* (t) *or not* (nt)*. The results of this test* (T) *can be excellent* (e)*, good* (g)*, bad* (b)*. If the test is not made, there are not available results* (nr)*. In case of a bad result, the company directly discards the option of building an advanced reactor.*

In the reactor problem, there are several examples of functional asymmetries in this problem:

A1. If the test is not made ($D_1 = nt$), there are not available results ($T = nr$).

A2. If the test is made ($D_1 = t$), the possible outcomes of the test are excellent, good or bad ($T \in \{e, g, b\}$).

A3. In case of obtaining bad results ($T = b$), the company will discard the option of building an advanced reactor (i.e., D_2 cannot take the value a and hence $D_2 \in \{c, n\}$).

On the other hand, the structural asymmetries present in this problem are:

A4. If the decision is not to build an advanced reactor ($D_2 \in \{c, n\}$) or the result for the test is bad ($T = b$), the probabilities of failure in such kind of reactor will not affect to the expected profit $(A \in \{\})$.

A5. If the decision is to build a reactor different to the conventional one ($D_2 \in \{a, n\}$), the probabilities of failure in such kind of reactor will not affect to the expected profit $(C \in \{\})$.

The reactor problem does not contain order asymmetries since the order of decisions is fixed at the time the model is specified.

Unlike IDs, decision trees, which were explained in Section 4.2, are flexible and expressive enough to take advantage of asymmetries. For example, Figure 6.1 depicts the previous problem as a decision tree that has been simplified thanks to the asymmetries. In case of functional asymmetries, some outgoing branches can be removed. For example, due to asymmetry A2, the node T consistent with $D_1 = t$ does not contain any outgoing branch labelled with nt. In case of structural asymmetries, some variables do not appear in some scenarios[1]. For example, due to asymmetry A5 the variable C does not appear in every scenario consistent with $D_2 = a$ or $D_2 = n$.

[1] In a decision tree, a path from the root to a leaf is a scenario.

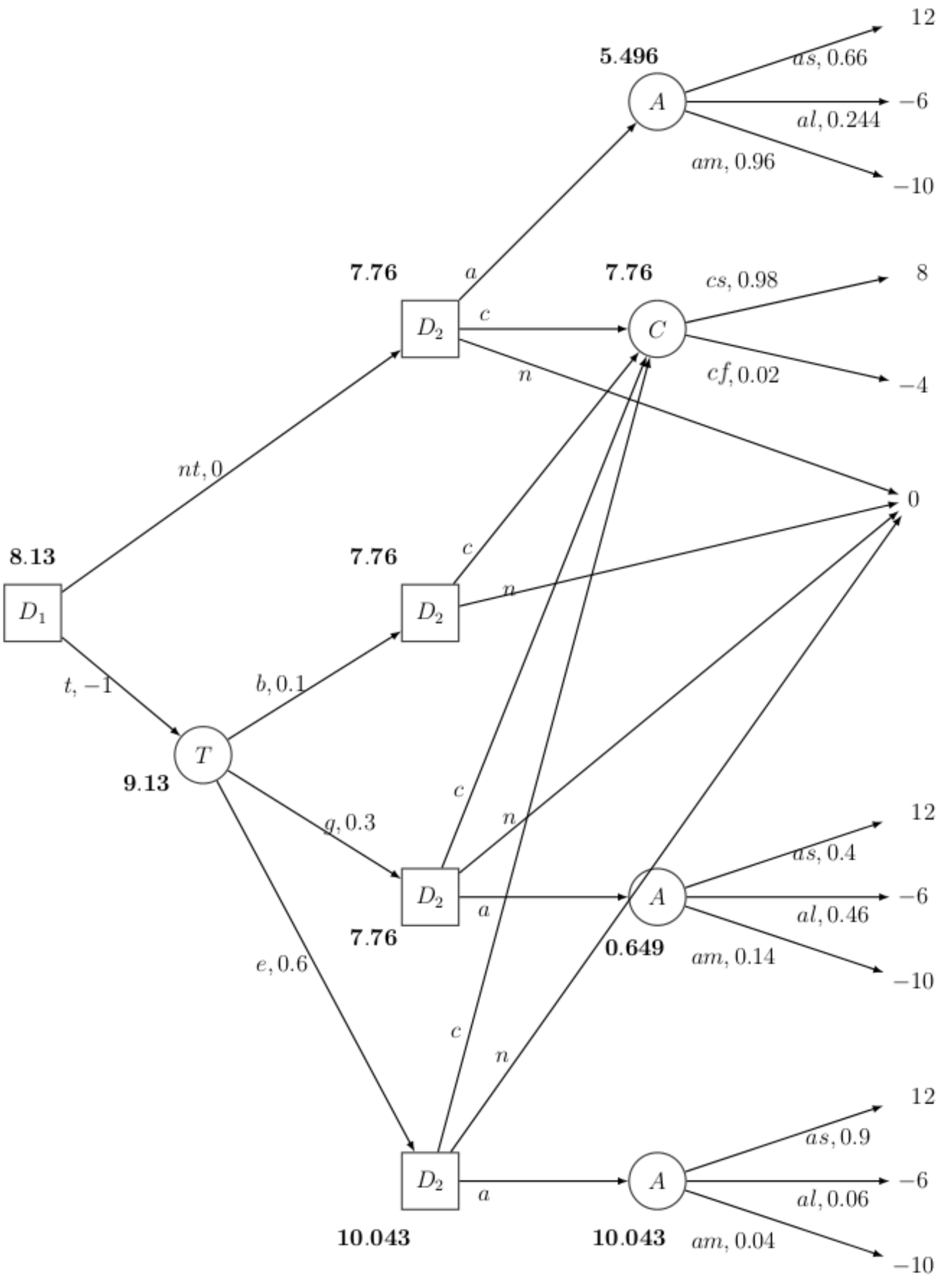


Figure 6.1: Decision tree representing the reactor problem described in Example 25. Details for building and evaluating this decision tree are given in [3].

Based on the tree representation of a decision problem, Definition 35 states a simple way for determining if a decision problem is asymmetric.

Definition 35 *A decision problem is said to be asymmetric if there exists at least one tree representation satisfying one of the following conditions:*

(i) The number of scenarios is not equal to the cardinality of the Cartesian product of the domains of all chance and decision variables.

(ii) The sequence of variables is not the same in all the scenarios.

In the previous decision tree there are scenarios with different sequences of variables, e.g. $\{D_1 = nt, D_2 = a, A = as\}$ and $\{D_1 = t, T = e, D_2 = a, A = am\}$. Therefore, condition *(ii)* of Definition 35 is satisfied and hence this problem is asymmetric. Condition *(i)* is neither satisfied as this tree shows 21 scenarios but the cardinality of the Cartesian product of the domains is 108.

Note that the (functional or structural) asymmetries imply that some scenarios in the problem are not allowed. These are called impossible scenarios and are not relevant for the evaluation of the problem. For example, in the reactor problem, the scenario defined by $\{D_1 = t, T = nr, D_2 = a, A = am\}$ is impossible due to the asymmetry A1. Moreover, any scenario containing the configuration $\{D_1 = t, T = nr\}$ will lead to an impossible scenario. In an asymmetric decision problem, we can identify those configurations of variables leading to an impossible scenario, which will be called impossible configurations. Given the asymmetries previously detailed, the configurations leading to impossible scenarios in the reactor problem are shown in Table 6.1.

In the decision tree representation, the simplification is done by removing any scenario containing impossible configurations. In our approach for reducing the complexity of the evaluation of asymmetric decision problems, we will also need to identify these impossible configurations in order to exclude them from the computations.

Asymmetry	Impossible configurations
A1	$\{D_1 = t, T = nr\}$
A2	$\{D_1 = nt, T = e\}, \{D_1 = nt, T = g\}, \{D_1 = nt, T = b\}$
A3	$\{T = b, D_2 = a\}$
A4	$\{T = b, A = as\}, \{T = b, A = al\}, \{T = b, A = am\}$, $\{D_2 = c, A = as\}$, $\{D_2 = c, A = al\}$, $\{D_2 = c, A = am\}$, $\{D_2 = n, A = as\}$, $\{D_2 = n, A = al\}$, $\{D_2 = n, A = am\}$,
A5	$\{D_2 = a, C = cs\}$, $\{D_2 = a, C = cf\}$, $\{D_2 = n, C = cs\}$, $\{D_2 = n, C = cf\}$,

Table 6.1: Configurations leading to impossible scenarios in the reactor problem.

6.2.2 IDs and asymmetries

The drawback of using IDs to model asymmetric decision problem is well known: to be modelled as an ID, such problems must be symmetrized by adding artificial states. For example, let us consider the ID modelling the reactor problem as shown in Example 26. Note that we must include the state nr to represent the situation in which the test is not performed (and hence, there are not results). This artificial state does not appear in the decision tree modelling the reactor problem (see Figure 6.1). However, the state nr does appear in the corresponding ID representation for the same problem. As a consequence, the addition of artificial states increases the size of the potentials. Moreover, potentials in IDs representing asymmetric decision problems might contain impossible configurations. For instance, $\psi(C, D_2)$ is defined for $\{D_2 = a, C = cs\}$, $\{D_2 = a, C = cf\}$, $\{D_2 = n, C = cs\}$ and $\{D_2 = n, C = cf\}$ which are impossible configurations due to the asymmetry A5 detailed in the previous section.

Example 26 (the reactor ID [3]) *Figure 6.2 shows an ID for the reactor problem. The sets of nodes are $\mathcal{U}_C = \{T, A, C\}$, $\mathcal{U}_D = \{D_1, D_2\}$ and $\mathcal{U}_V = \{U_1, U_2, U_3\}$. The sets of PPs and UPs are $\Phi = \{\phi(T|D_1, A), \phi(A), \phi(C)\}$ and $\Psi = \{\psi(D_1), \psi(A, D_2), \psi(C, D_2)\}$ respectively. Their numerical values are depicted below. Table entries corresponding with impossible configurations are shown in grey.*

$$\phi(T|D_1,A) = \begin{array}{ccccccc} & t & & & nt & & \\ as & al & am & as & al & am & \\ 0.818 & 0.147 & 0.25 & 0.0 & 0.0 & 0.0 & e \\ 0.182 & 0.565 & 0.437 & 0.0 & 0.0 & 0.0 & g \\ 0.0 & 0.288 & 0.313 & 0.0 & 0.0 & 0.0 & b \\ 0.0 & 0.0 & 0.0 & 1.0 & 1.0 & 1.0 & nr \end{array}$$

$$\phi(A) = \begin{array}{cc} 0.660 & as \\ 0.244 & al \\ 0.096 & am \end{array}, \ \phi(C) = \begin{array}{cc} 0.98 & cs \\ 0.02 & cf \end{array}, \ \psi(D_1) = \begin{array}{cc} -10 & t \\ 0.0 & nt \end{array},$$

$$\psi(A,D_2) = \begin{array}{cccc} as & al & am & \\ 12.0 & -6.0 & -12.0 & a \\ 0.0 & 0.0 & 0.0 & c \\ 0.0 & 0.0 & 0.0 & n \end{array}, \ \psi(C,D_2) = \begin{array}{ccc} cs & cf & \\ 0.0 & 0.0 & a \\ 8.0 & -4.0 & c \\ 0.0 & 0.0 & n \end{array},$$

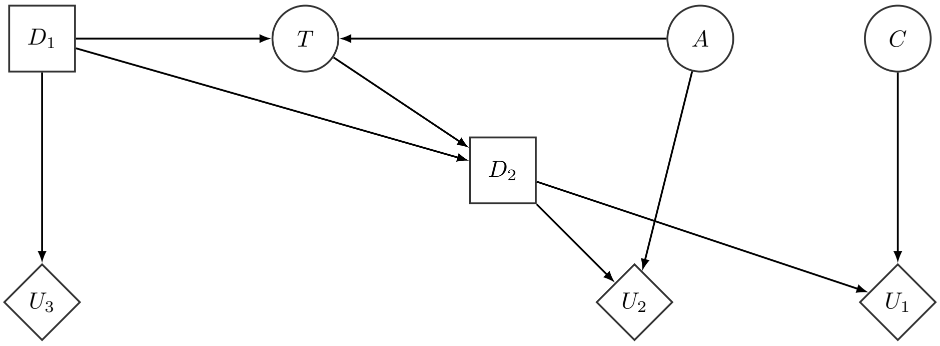


Figure 6.2: ID representing the reactor problem described in Example 25.

In conventional evaluation algorithms, operations on potentials are performed with all their values, even with those corresponding to impossible configurations. This implies that a considerable amount of unnecessary memory space and operations may be involved during the evaluation. We will say that unnecessary (arithmetic) operations are those done for computing values in a potential associated to impossible configurations. To illustrate this drawback, Example 27 details the operations performed with the VE algorithm[2] during the first iteration.

Example 27 (the reactor ID evaluated with VE) *Let us consider the removal order A, C, D_1, T, D_2. In the first iteration, Algorithm 2 is invoked for removing variable A:*

- *Step 1: the sets of relevant PPs and UPs for removing A, are* $\Phi_A = \{\phi(T|D_1, A), \phi(A))\}$ *and* $\Psi = \{\psi(A, D_2)\}$.
- *Step 2: combine the PPs obtaining* $\phi_A(T, A|D_1)$*. As there is only one UP containing A, the combination of utilities is not done, i.e.* $\psi_A(A, D_2) = \psi(A, D_2)$.

$$\phi_A(T, A|D_1) = \phi(T|D_1, A) \cdot \phi(A) =$$

$$\begin{array}{cccccc|c}
 & t & & & nt & & \\
as & al & am & as & al & am & \\
\hline
0.540 & 0.036 & 0.024 & 0.0 & 0.0 & 0.0 & e \\
0.120 & 0.138 & 0.042 & 0.0 & 0.0 & 0.0 & g \\
0.0 & 0.070 & 0.030 & 0.0 & 0.0 & 0.0 & b \\
0.0 & 0.0 & 0.0 & 0.660 & 0.244 & 0.096 & nr
\end{array} \tag{6.1}$$

The resulting potential $\phi_A(T, A|D_1)$ *contains 15 values associated to impossible configurations due to asymmetries A1, A2 and A3. Therefore, the computation of this PP implies 15 unnecessary multiplications, out of 24.*

- *Step 4: sum-marginalize variable A out of* $\phi_A(T, D_1|A)$*:*

$$\phi'_A(T|D_1) = \sum_A \phi_A(T, A|D_1) = \begin{array}{cc|c}
t & nt & \\
\hline
0.6 & 0.0 & e \\
0.3 & 0.0 & g \\
0.1 & 0.0 & b \\
0.0 & 1.0 & nr
\end{array} \tag{6.2}$$

[2] More details about the VE algorithm are given in Section 4.5.1.

Previous potential contains 4 values associated to impossible configurations due to asymmetries A1 and A2. Thus, this sum-marginalization implies 8 unnecessary additions out of 16.

Combine the PP and UP:

$$\psi(D_2,T,D_1,A) = \phi_A(T,A|D_1)\cdot\psi_A(A,D_2) =$$

$$\begin{array}{cccccccc}
 & t & & & nt & & & \\
as & al & am & as & al & am & & \\
6.479 & -0.215 & -0.288 & 0.0 & 0.0 & 0.0 & e & \\
1.441 & -0.827 & -0.503 & 0.0 & 0.0 & 0.0 & g & a \\
0.0 & -0.422 & -0.361 & 0.0 & 0.0 & 0.0 & b & \\
0.0 & 0.0 & 0.0 & 7.92 & -1.464 & -1.152 & nr & \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & e & \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & g & c \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & b & \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & nr & \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & e & \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & g & n \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & b & \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & nr &
\end{array} \tag{6.3}$$

The potential $\psi(D_2,T,D_1,A)$ contains 63 values associated to impossible configurations due to asymmetries A1, A2, A3 and A4. Thus, 63 unnecessary multiplications out of 72 are performed.

Sum-marginalize A out of $\psi(D_2,T,A,D_1)$:

$$\psi(D_2, D_1, T) = \sum_A \psi(D_2, T, D_1, A) =$$

$$\begin{array}{cc}
\begin{array}{cccccccc}
 & t & & & & nt & & \\
e & g & b & nr & e & g & b & nr
\end{array} & \\
\begin{bmatrix}
5.975 & 0.111 & -0.782 & 0.0 & 0.0 & 0.0 & 0.0 & 5.304 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0
\end{bmatrix} & \begin{array}{c} a \\ c \\ n \end{array}
\end{array} \tag{6.4}$$

The resulting UP of the sum-marginalization has 13 values associated to impossible configurations due to asymmetries A1, A2 and A3. The computation of each value requires 2 additions. Thus, this operation implies 26 unnecessary additions out of 48.

Normalize the resulting UP:

$$\psi'_A(D_2, D_1, T) = \frac{\psi(D_2, D_1, T)}{\phi'_A(T|D_1)} =$$

$$\begin{array}{cc}
\begin{array}{cccccccc}
 & t & & & & nt & & \\
e & g & b & nr & e & g & b & nr
\end{array} & \\
\begin{bmatrix}
9.963 & 0.370 & -7.800 & 0.0 & 0.0 & 0.0 & 0.0 & 5.304 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0
\end{bmatrix} & \begin{array}{c} a \\ c \\ n \end{array}
\end{array} \tag{6.5}$$

This UP has 13 values associated to impossible configurations due to asymmetries A1, A2 and A3. As a consequence, this division of potentials implies 13 unnecessary divisions out of 24.

- *Step 9: update the sets of potentials:* $\Phi = \{\phi'_A(T|D_1), \phi(C)\}$, *and* $\Psi = \{\psi(D_1), \psi'_A(D_2, D_1, T), \psi(C, D_2)\}$.

With previous example, we have illustrated that, if asymmetries are not taken into account, many unnecessary operations are involved during the evaluations:

only for the removal of variable A, a total of 184 arithmetic operations are required and 126 of them are unnecessary. This number could be higher if we also consider as unnecessary those operations whose operands are values associated to impossible configurations. Moreover, potentials in asymmetric decision problems usually contain a high number of values equal to 0.0. Thus, many multiplications could also be avoided if we detect that one of the operands is equal to 0.0.

6.3 Asymmetries and binary trees

6.3.1 Constraint rules

A constraint rule is intuitive representation of an asymmetry present in a decision problem. More formally, a constraint rule κ is a logical expression in the form of $antecedent \Rightarrow consequent$. An *atomic sentence* is a pair (variable, set of values): $X \in \{x_i, \ldots x_j\}$. If this set of values contains only one element, e.g., $X \in \{x_i\}$, we can use the notation $X = x_i$. Atomic sentences can be connected with logical operators to form *logical sentences*. Valid logical operators are $\wedge(and)$, $\vee(or)$ and $\neg(not)$. For constraint rules, both antecedents and consequents are expressed using logical sentences.

Example 28 (constraint rules for the reactor problem) *The asymmetries detailed in Section 6.2.1 for the reactor problem can be defined by the following set of constraint rules*[3]*:*

$$\begin{aligned}
\kappa_1 &: \quad D_1 = t \Rightarrow T \in \{e, g, b\} \\
\kappa_2 &: \quad D_1 = nt \Rightarrow T \in \{nr\} \\
\kappa_3 &: \quad T = b \Rightarrow D_2 \in \{c, n\} \\
\kappa_4 &: \quad (T = b) \vee (D_2 \in \{c, n\}) \Rightarrow A \in \{\} \\
\kappa_5 &: \quad D_2 \in \{a, n\} \Rightarrow C \in \{\}
\end{aligned}$$

[3] In this example, the asymmetry $A1$ corresponds with the constraint rule κ_1, $A2$ with κ_2, and so on.

Impossible configurations can be identified using the constraint rules, which usually correspond with the values equal to 0 in the potential. Considering the constraint rule κ_1 and the probability potential $\phi(T|D_1, A)$, we can state that:

$$\phi(nr|t, as) = \phi(nr|t, al) = \phi(nr|t, am) = 0$$

The configuration $\{D_1 = t, T = nrt\}$ corresponds to an impossible scenario (see Table 6.1) that must not be considered for computations. An atomic sentence could have an empty set of values for the consequent, e.g the constraint rule κ_5 means that $\{D_2 = a, C = cs\}$, $\{D_2 = a, C = cf\}$, $\{D_2 = n, C = cs\}$ and $\{D_2 = n, C = cf\}$ lead to impossible scenarios.

The use of constraint rules have several advantages. First of all, they make the elicitation process easier (reducing the number of scenarios and therefore the number of parameters to assess); secondly, they help to make both qualitative and quantitative knowledge consistent; and thirdly, they clearly state invalid scenarios, making the contingent nature of the decision problem clear. In IDs representing asymmetric decision problems, we will often have more than one constraint rule. Thus, a set of such logical expressions will be denoted as $\mathcal{K}$.

6.3.2 Binary constraints trees

Even though the use of constraint rules is an intuitive way for describing asymmetries in a decision problem, we need to combine the information encoded in these rules with the quantitative information of an ID (i.e., potentials). Assuming that potentials are represented using BTs, constraint rules should be also transformed into BT before the evaluation. In doing so, the combination is almost straightforward as the same data structure is used. A BT that represents a constraint rule will be called *binary constraint tree* (BCT).

In a previous work, the use of NTs for representing constraint rules [54] was proposed. However, BTs can capture more fine-grained independencies than those captured using NTs. Potentials and constraints need less memory space to be represented as a BT than as a NT. As a consequence, more efficient evaluation algorithms will be obtained.

Definition 36 (binary constraint tree) *A binary constraint tree $\mathcal{BT}^{\kappa}(\mathbf{X}_I)$ for a constraint rule κ with variables $\mathbf{X}_I$ is a BT whose leaf nodes contain either a 0 or 1. Given a configuration $\mathbf{x}_I \in \mathbf{X}_I$, if $\mathcal{BT}^{\kappa}(\mathbf{x}_I)$ is equal to 0 means that $\mathbf{x}_I$ is an impossible configuration.*

Example 29 (BCTs for the reactor problem) *Given the set of constraint rules stated in Example 28, the BCTs representing such sets of constraints are shown in Figure 6.3.*

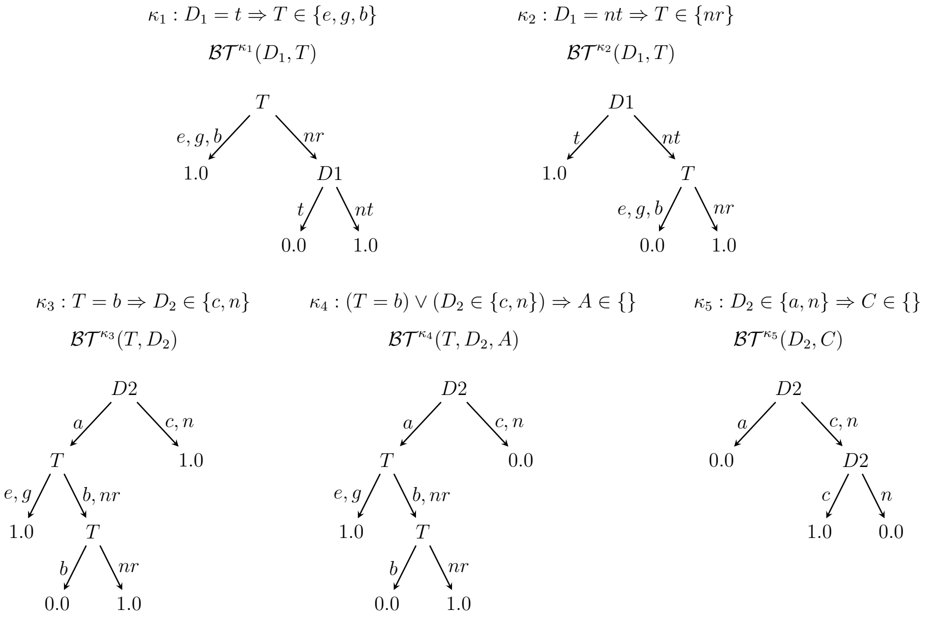

$\kappa_1 : D_1 = t \Rightarrow T \in \{e, g, b\}$
$\mathcal{BT}^{\kappa_1}(D_1, T)$
T
e, g, b
nr
1.0
$D1$
t
nt
0.0
1.0
$\kappa_2 : D_1 = nt \Rightarrow T \in \{nr\}$
$\mathcal{BT}^{\kappa_2}(D_1, T)$
$D1$
t
nt
1.0
T
e, g, b
nr
0.0
1.0
$\kappa_3 : T = b \Rightarrow D_2 \in \{c, n\}$
$\mathcal{BT}^{\kappa_3}(T, D_2)$
$D2$
a
c, n
T
1.0
e, g
b, nr
1.0
T
b
nr
0.0
1.0
$\kappa_4 : (T = b) \vee (D_2 \in \{c, n\}) \Rightarrow A \in \{\}$
$\mathcal{BT}^{\kappa_4}(T, D_2, A)$
$D2$
a
c, n
T
0.0
e, g
b, nr
1.0
T
b
nr
0.0
1.0
$\kappa_5 : D_2 \in \{a, n\} \Rightarrow C \in \{\}$
$\mathcal{BT}^{\kappa_5}(D_2, C)$
$D2$
a
c, n
0.0
$D2$
c
n
1.0
0.0

Figure 6.3: BCTs representing the sets of constraints rules stated in Example 28 for the reactor problem.

Given a $\mathcal{BT}^{\kappa}(\mathbf{X}_I)$ representing a constraint rule κ, impossible scenarios defined by κ are all the configurations consistent with each leaf node t labelled with 0, i.e. $\Omega^t_{\mathbf{X}_I}$. For example, the impossible configurations defined by $\mathcal{BT}^{\kappa_2}(D_1, T)$

are $\{D_1 = nt, T = e\}, \{D_1 = nt, T = g\}, \{D_1 = nt, T = b\}$. A value equal to 1 means that, taking into account only this constraint tree, the configuration is possible (it could be impossible according to another constraint). For instance, the configuration $\{D_1 = nt, T = e\}$ is possible according to $\mathcal{BT}^{\kappa_1}(D_1, T)$ but impossible according to $\mathcal{BT}^{\kappa_2}(D_1, T)$.

Given a constraint rule κ over the set of variables $\mathbf{X}_I$, the process for building $\mathcal{BT}^{\kappa}$ is quite simple: First, create a table $\mathcal{T}(\mathbf{X}_I)$. Then, the logical expression κ is evaluated for each $\mathbf{x}_I \in \mathbf{X}_I$. If the result is true, then set $\mathcal{T}(\mathbf{x}_I)$ to 1, otherwise set $\mathcal{T}(\mathbf{x}_I)$ to 0. Finally, build $\mathcal{BT}^{\kappa}$ from $\mathcal{T}(\mathbf{X}_I)$ using the same algorithm that was proposed for building a BT representing a potential (see Section 5.4). Notice that in a BCT branches with the same value can be grouped (as happens with a BT from a potential) and as a consequence, a variable in the domain of the constraint may not appear in any node of the tree. This situation happens, for example, in $\mathcal{BT}^{\kappa_5}(D_2, C)$ where variable C is not present in the tree. Despite of that, the variable C should be taken into account for checking the applicability of the rule (concepts about the applicability of the rules are explained in Chapter 10).

6.4 Conclusions

In this chapter, we have introduced the so-called asymmetric decision problems, where a particular action or observation can lead to very different possibilities and, as a consequence, not all decisions and variables are considered in all circumstances. In particular, this kind of problems are characterized by the presence of configurations of variables that are not allowed (i.e., they are impossible) due to the asymmetries. We have explained the drawback of IDs related to their inability for efficiently representing this kind of problems: potentials might contain some of these impossible configurations that are not relevant for computing the optimal policies. In standard evaluation algorithms, operations on potentials are performed with all their values, including those related to impossible configurations. As a consequence, a considerable amount of unnecessary memory space and computation may be involved.

For addressing this problem, we have introduced constraints rules, which are logical expressions used for intuitively defining asymmetries. We have proposed transforming these rules into BTs, which are called binary constraint trees (BCTs). This makes possible their combination with the potentials and hence avoids much of the unnecessary computation. In this chapter we have explained the basic concepts about BCTs. The details for their application during the evaluation will be later given in Chapter 10.

Chapter 7

Interval-valued Potentials

7.1 Introduction

Exactly as BNs, IDs usually demand a sharp estimation of their parameters (i.e., the set of associated potentials). Both probabilities and utilities might be quantified by expert knowledge or statistical learning. Yet, the structures for the representation of potentials detailed in previous chapters (i.e. tables or trees) assign a single (i.e., sharp or precise) value to each configuration in the potential. Thus, these data structures can be unable to express an imprecise expert judgement (e.g., which is the number modelling the probability for an option more probable than its negation? And the probability of $X = x_1$ is high). This issue appears also when coping with scarce or missing data (e.g., probabilities conditional on rare events).

In order to deal with this problem, in this dissertation we propose replacing with intervals the sharp values of potentials involved in an ID . Such generalization, namely *interval-valued potentials*, is formalized in this chapter. By contrast, the algorithms for evaluating IDs with interval-valued potentials are later explained in Chapter 11.

7.2 Related work

In the last two decades, various extensions of BNs intended to support generalized probabilistic statements have been proposed. These models have been developed in the fields of possibility theory [2], evidence theory [111], and imprecise probability [35, 93]. The latter models, called *credal set* (CS) [37], are closed convex sets of probability distributions: a CS represents an infinite number of probability distributions. CSs are a very suitable and general tool for the representation of the quantitative information in PGMs. First, they can be used for modelling imprecision, being therefore suited to model in a natural way ignorance, vagueness and contrast. For example, they can model statements such as “the probability of $X = x_1$ is high” or “$X = x_1$ is more probable than $X = x_2$”. Secondly, they are very general: the majority of the models used nowadays for the representation of uncertainty can be considered as special cases of CSs.

A CS for a random variable X is denoted by $K(X)$ and it holds that $\sum_{x \in \Omega_X} P(x) = 1$ for each $P(X) \in K(X)$. Geometrically, $K(X)$ is a polytope in the hyperplane of all possible probability distributions for the variable X. Such CS can be described by enumerating only the set of its extreme probability distributions, which is finite and corresponds to the vertices of the polytope. These vertices are called *extreme points* and denoted $ext[K(X)]$. An example of CS described by its extreme points is given below.

Example 30 *From the oil wildcatter's decision problem detailed in Example 10 (page 49), let us consider the random variable O with the domain* $\Omega_O = \{e, w, s\}$. *In the elicitation process we obtain the following constraints:*

- $P(s)$ *is low, let us say* 0.2 *or lower.*
- $P(e)$ *is greater or equal than* $P(w)$.

Then, a CS considering such information is depicted in Figure 7.1. Each point in the graphic is a tuple $(P(e), P(w), P(s))$*, i.e., a probability distribution* $P(O)$. *Two overlapped planes are shown. The grey one represents any possible point* *satisfying* $P(e) + P(w) + P(s) = 1$. *On the other hand, the blue one represents those points also satisfying previous constraints.*

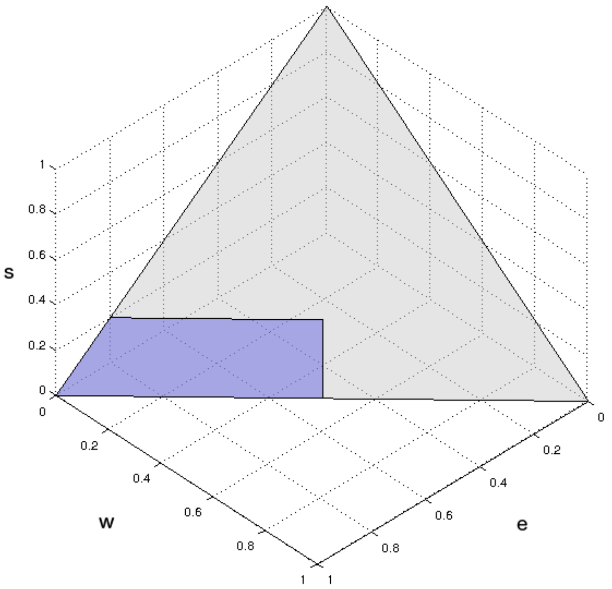


Figure 7.1: CS represented with the extreme points $ext[K(O)]=$ $\{[1, .0, .0]^T$, $[.5, .5, .0]^T$, $[.4, .4, .2]^T$, $[.8, .0, .2]^T\}$.

A BN with CSs attached to each node is called a *credal network* [37]. We can consider the following sensitivity-analysis interpretation: a credal network defines a collection of Bayesian networks, all over the same variables and with the same graph, whose parameters are consistent with constraints modelling a limited ability in the assessment of sharp estimates. Note that CSs are a generalization of the interval potentials proposed here, however they have an important drawback: the complexity of the inference increases exponentially in the number of extreme points.

Something similar has been also done with decision trees [58, 60, 67], while the situation is different for IDs. The early attempts of Fertig and Breese [50, 51] first, and Zaffalon [47] after, to extend these models to non-sharp quantification are among the few works in this direction.[1,2]

The Fertig and Breese work can be considered as the closest approach to ours. They also propose the use of an interval (i.e. a lower and upper bound) for each configuration in a potential. In case of PPs, only the lower bounds are specified, while the upper ones are computed from the lower ones. Here, we will use the term *one-sided* potential to refer to such generalization. For example, let us consider a PP over X, then for each state x in Ω_X we have a lower bound denoted $b(x)$. Each upper bound $u(x)$ is computed as follows.

$$u(x) = 1 - \sum_{x' \in \Omega_X \setminus \{x\}} b(x') \tag{7.1}$$

7.3 Interval-valued potentials

Herein we formalize the notion of interval-valued potential that will be used in this dissertation. It is basicaly a generalization of the notions of UP and PP explained in Section 4.3. For the case of utilities we base on the interval utilities proposed by Fertig and Brese [50, 51], and we will call them *interval-valued utility potentials* (IUPs). By contrast, for the probabilities, we use the notion of probability interval proposed by Campos et al. [41], and we will use the term *interval-valued probability potential* (IPP). Thus notion of UP can be extended to intervals as follows.

Definition 37 (interval utilities) *An interval-valued utility potential (IUP) over the set of variables $\mathbf{X}_I$ is a pair of UPs over $\mathbf{X}_I$. We use the compact notation $\underline{\overline{\psi}}(\mathbf{X}_I)$ for an IUP over $\mathbf{X}_I$, $\underline{\psi}$ and $\overline{\psi}$ are the two UPs involved in the specification and are called, respectively, the lower and upper bounds of the IUP. The extension*

[1] The work of Zhou et al. [114] combining sharp probabilities with interval-valued utilities is just a trivial special case of the general framework we present here.

[2] Sensitivity analysis does not require the specification of more general classes of models, being only focused on the results of the inferences. Thus, it should be regarded as a different topic, which, as a matter of fact, received more attention (e.g., [85]).

$\overline{\underline{\psi}}^*(\mathbf{X}_I)$ *of this IUP is the set of UPs consistent with the bounds, i.e.,*

$$\overline{\underline{\psi}}^*(\mathbf{X}_I) := \left\{\psi : \Omega_{\mathbf{X}_I} \to \mathbb{R} \;\middle|\; \underline{\psi}(\mathbf{x}_I) \leq \psi(\mathbf{x}_I) \leq \overline{\psi}(\mathbf{x}_I), \forall \mathbf{x}_I \in \Omega_{\mathbf{X}_I}\right\} . \qquad (7.2)$$

Note that we consider the following sensitivity-analysis interpretation: an IUP is a set of (precise) UPs consistent with the interval constraints. Such set is called *extension*, and for an IUP $\overline{\underline{\psi}}$ is non-empty if and only if $\underline{\psi}(\mathbf{x}_I) \leq \overline{\psi}(\mathbf{x}_I)$ $\forall \mathbf{x}_I \in \Omega_{\mathbf{X}_I}$. Similarly, PPs can be extended as follows.

Definition 38 (interval probabilities) *An interval-valued probability potential (IPP) over* $\mathbf{X}_I$ *given* $\mathbf{X}_J$ *is a pair of (in general not normalized) PPs over* $\mathbf{X}_I$ *given* $\mathbf{X}_J$*. We denote such an IPP as* $\overline{\underline{\phi}}(\mathbf{X}_I|\mathbf{X}_J)$*, where* $\underline{\phi}(\mathbf{X}_I|\mathbf{X}_J)$ *and* $\overline{\phi}(\mathbf{X}_I|\mathbf{X}_J)$ *are the two (unnormalized) bounds. The extension* $\overline{\underline{\phi}}^*(\mathbf{X}_I|\mathbf{X}_J)$ *of this IPP is the set of PPs consistent with the bounds, i.e.,*

$$\overline{\underline{\phi}}^*(\mathbf{X}_I|\mathbf{X}_J) := \left\{\phi : \Omega_{\mathbf{X}_I}\times \Omega_{\mathbf{X}_J}\to \mathbb{R}_0^+ \;\middle|\; \begin{array}{l} \sum_{\mathbf{x}_I} \phi(\mathbf{x}_I|\mathbf{x}_J) = 1 \\ \underline{\phi}(\mathbf{x}_I|\mathbf{x}_J) \leq \phi(\mathbf{x}_I|\mathbf{x}_J) \leq \overline{\phi}(\mathbf{x}_I|\mathbf{x}_J) \\ \forall(\mathbf{x}_I, \mathbf{x}_J) \in \Omega_{\mathbf{X}_I} \times \Omega_{\mathbf{X}_J} \end{array}\right\} . \qquad (7.3)$$

The difference between an IPP and a precise one is that, instead of a single value, associated to each configuration $(\mathbf{x}_I, \mathbf{x}_J)$ there is an interval $[\underline{\phi}(\mathbf{x}_I|\mathbf{x}_J), \overline{\phi}(\mathbf{x}_I|\mathbf{x}_J)]$. Note that an IPP represents a bounded and infinite set of precise PPs (its extension). If the extension is non-empty, the IPP is said to be *proper*. More formally, this concept can be defined as follows.

Definition 39 (proper interval probability potential) *Let* $\overline{\underline{\phi}}(\mathbf{X}_I|\mathbf{X}_J)$ *be an IPP, then we say that this IPP is proper iff satisfies that:*

$$\sum_{\mathbf{x}_I \in \Omega_{\mathbf{X}_I}} \underline{\phi}(\mathbf{x}_I|\mathbf{x}_J) \leq 1 \leq \sum_{\mathbf{x}_I \in \Omega_{\mathbf{X}_I}} \overline{\phi}(\mathbf{x}_I|\mathbf{x}_J) , \quad \forall \mathbf{x}_J \in \Omega_{\mathbf{X}_J} \qquad (7.4)$$

We will require an additional condition for PPs, called *reachability* [41], which can be defined as follows.

Definition 40 (reachable interval probability potential) *Let* $\overline{\underline{\phi}}(\mathbf{X}_I|\mathbf{X}_J)$ *be an IPP, then we say that this IPP is reachable iff satisfies the following two conditions:*

$$\overline{\phi}(\mathbf{x}_I|\mathbf{x}_J) \quad + \sum_{\mathbf{x}'_I \in \Omega_{\mathbf{X}_I} \setminus \{\mathbf{x}_I\}} \underline{\phi}(\mathbf{x}'_I|\mathbf{x}_J) \leq 1 \tag{7.5}$$

$$\underline{\phi}(\mathbf{x}_I|\mathbf{x}_J) \quad + \sum_{\mathbf{x}'_I \in \Omega_{\mathbf{X}_I} \setminus \{\mathbf{x}_I\}} \overline{\phi}(\mathbf{x}'_I|\mathbf{x}_J) \geq 1 \tag{7.6}$$

for each $(\mathbf{x}_I, \mathbf{x}_J) \in \Omega_{\mathbf{X}_I} \times \Omega_{\mathbf{X}_J}$

The meaning is that for each $p \in [\underline{\phi}(\mathbf{x}_I|\mathbf{x}_J), \overline{\phi}(\mathbf{x}_I|\mathbf{x}_J)]$, there is at least a PP $\phi \in \overline{\underline{\phi}}^*$ such that $\phi(\mathbf{x}_I|\mathbf{x}_J) = p$. Note also that an IPP with non-empty extension can be always reduced to a reachable one by shrinking its bounds and this has no effect on its extension. Given an IPP, we always check whether it is reachable and, if not, we apply the transformation detailed in Proposition 7. In all the algorithms with IPPs proposed in this dissertation, the reachability transformation will be applied after any modification of the IPPs.

Proposition 7 (reachability transformation) *Let* $\overline{\underline{\phi}}(\mathbf{X}_I|\mathbf{X}_J)$ *be a proper IPP. Then* $\overline{\underline{\phi}}'(\mathbf{X}_I|\mathbf{X}_J)$ *is the reachable IPP associated whose upper and lower bounds are:*

$$\underline{\phi}'(\mathbf{x}_I|\mathbf{x}_J) = \max\{\underline{\phi}(\mathbf{x}_I|\mathbf{x}_J),\ 1 \ - \sum_{\mathbf{x}'_I \in \Omega_{\mathbf{X}_I} \setminus \{\mathbf{x}_I\}} \overline{\phi}(\mathbf{x}'_I|\mathbf{x}_J)\} \tag{7.7}$$

$$\overline{\phi}'(\mathbf{x}_I|\mathbf{x}_J) = \min\{\overline{\phi}(\mathbf{x}_I|\mathbf{x}_J),\ 1 \ - \sum_{\mathbf{x}'_I \in \Omega_{\mathbf{X}_I} \setminus \{\mathbf{x}_I\}} \underline{\phi}(\mathbf{x}'_I|\mathbf{x}_J)\} \tag{7.8}$$

for each $(\mathbf{x}_I, \mathbf{x}_J) \in \Omega_{\mathbf{X}_I} \times \Omega_{\mathbf{X}_J}$

Both the extensions of an IUP and an IPP are convex sets of, respectively, UPs and PPs. Convex sets of UPs and PPs which are not extensions of IUPs and IPPs can be also considered, but this topic would be beyond the scope of this dissertation. In Example 31 we illustrate the concepts of interval-valued potentials and reachability.

Example 31 *Consider the precise potentials attached to the oil wildcatter ID specified in Example 11 in page 55. Then, we can specify the following interval-valued potentials over the same variables as follows*[3].

$$\overline{\underline{\phi}}(O) = \begin{array}{c} \begin{bmatrix} [.475, .525] \\ [.285, .335] \\ [.190, .240] \end{bmatrix} \end{array} \begin{array}{c} e \\ w \\ s \end{array} \qquad \overline{\underline{\psi}}(T) = \begin{bmatrix} [-10, -5] \\ [-5, 5] \end{bmatrix} \begin{array}{c} t \\ nt \end{array}$$

$$\overline{\underline{\psi}}(O, D) = \begin{array}{c} \begin{array}{cc} d & nd \end{array} \\ \begin{bmatrix} [-75, -65] & [-5, 5] \\ [45, 55] & [-5, 5] \\ [195, 205] & [-5, 5] \end{bmatrix} \end{array} \begin{array}{c} \\ e \\ w \\ s \end{array}$$

$$\overline{\underline{\phi}}(S|O, T) = \begin{array}{c} \begin{array}{cccccc} & t & & & nt & \\ e & w & s & e & w & s \end{array} \\ \begin{bmatrix} [.095, .145] & [.285, .335] & [.475, .525] & [.317, .367] & [.317, .367] & [.317, .367] \\ [.288, .335] & [.380, .430] & [.380, .430] & [.317, .367] & [.317, .367] & [.317, .367] \\ [.570, .620] & [.285, .335] & [.095, .145] & [.317, .367] & [.317, .367] & [.317, .367] \end{bmatrix} \end{array} \begin{array}{c} \\ \\ c \\ o \\ d \end{array},$$

It is a trivial exercise to check that these potentials have non-empty extensions, the UPs and PPs in Example 11 are included in these extensions. In order to check that the IPPs are reachable, we have to check conditions given in Definition 40. As an example, $\overline{\underline{\phi}}(O)$ is reachable as the following conditions are satisfied.

$$\underline{\phi}(e) + \overline{\phi}(w) + \overline{\phi}(s) = 0.475 + 0.335 + 0.24 = 1.05 \geq 1$$

$$\overline{\phi}(e) + \underline{\phi}(w) + \underline{\phi}(s) = 0.525 + 0.285 + 0.19 = 1 \leq 1$$

$$\overline{\phi}(e) + \underline{\phi}(w) + \overline{\phi}(s) = 0.525 + 0.285 + 0.24 = 1.05 \geq 1$$

[3] When showing IPPs in matrix form, the integer digits will be omitted in probability values lower that 1 (e.g., 0.475 will be displayed as .475).

$$\underline{\phi}(e) + \overline{\phi}(w) + \underline{\phi}(s) = 0.475 + 0.335 + 0.19 = 1 \leq 1$$

$$\overline{\phi}(e) + \overline{\phi}(w) + \underline{\phi}(s) = 0.525 + 0.335 + 0.19 = 1.05 \geq 1$$

$$\underline{\phi}(e) + \underline{\phi}(w) + \overline{\phi}(s) = 0.475 + 0.285 + 0.24 = 1 \leq 1$$

The difference w.r.t. other similar approaches is that upper bounds are explicitly defined. In the one-sided potentials (see Section 7.2) upper bounds are computed from the lower ones. As a consequence, the intervals defined by one-sided potentials might be unnecessarily large compared to those defined by interval-valued potential (assuming that reachability transformation has been applied). An example of such situation is shown below.

Example 32 *Let X be a variable with the domain $\Omega_X = \{x_1, x_2, x_3\}$. Let us consider the following reachable IPP:*

$$\underline{\overline{\phi}}(X) = \begin{array}{cc} \begin{bmatrix} [.2, .3] \\ [.5, .6] \\ [.1, .2] \end{bmatrix} & \begin{array}{c} x_1 \\ x_2 \\ x_3 \end{array} \end{array}$$

Now, consider the equivalent one-sided potential with the same lower bounds, i.e., $b(x_1) = 0.2$, $b(x_2) = 0.5$ and $b(x_3) = 0.1$. Using Equation 7.1, the upper bounds can be computed as follows.

$$u(x_1) = 1 - (b(x_2) + b(x_3)) = 1 - (0.5 + 0.1) = 0.4$$

$$u(x_2) = 1 - (b(x_1) + b(x_3)) = 1 - (0.2 + 0.1) = 0.7$$

$$u(x_3) = 1 - (b(x_1) + b(x_2)) = 1 - (0.2 + 0.5) = 0.3$$

Note that the size of the intervals defined by the one-sided potentials are larger that those defined by the IPP, i.e., it holds that $\overline{\phi}(x_i) - \underline{\phi}(x_i) < u(x_i) - b(x_i)$ for each $x_i \in \Omega_X$.

7.4 Conclusions

In this chapter we have presented the notion of interval-valued potential. This will allow to generalize the standard ID formalism to intervals, allowing an imprecise estimation of their potentials. The definitions of the usual operations over interval potentials and the generalization of the evaluation algorithms will be later described in Chapter 11.

Part III

Evaluation

Chapter 8

Evaluation with Binary Trees

8.1 Introduction

In Chapter 5, we described the key issues about BTs for representing the potentials involved in an ID (i.e. PPs and UPs). In particular, methods for building and pruning BTs were given. As a reminder, the advantage of this kind of tree is the possibility of representing context-specific independencies that are more fine-grained compared to those encoded using other representations. This enhanced capability can be used to improve the efficiency of the inference algorithms used for evaluating IDs.

In this chapter, we explain how to evaluate IDs whose potentials are represented as BTs. For that, operations considered in Section 4.3.2.1 for tables need to be extended to BTs. Then, we explain how to adapt the main ID evaluation algorithms to work with BTs. In the experimentation section, the behaviour of these algorithms using BTs is analysed . For that purpose, BTs, NTs and tables are compared in different aspects (computation time, storage and error level) .

8.2 Operations with binary trees

Evaluating an ID requires performing several operations with its potentials. Herein we describe how these operations can be performed directly on BTs. In particular, evaluation algorithms require the *restriction* operation, multiple types of element-wise operations (*multiplication, division, addition* and *maximum*) and two types of marginalizations (*sum-marginalization* and *max-marginalization*).

8.2.1 Restriction

First, we will explain the *restriction* operation which is required by other operations on BTs (e.g., by combinations). The *restriction* of a given $\mathcal{BT}$ to a set of states S_X, denoted by $\mathcal{BT}^{R(X,S_X)}$, consists of returning the part of the potential that is compatible with S_X. In other words, we select those configurations in which X takes one of the values in S_X. The proposed procedure for this operation is detailed in Algorithm 9, which describes the restriction of a $\mathcal{BT}$ to a set of states S_X $(S_X \subseteq \Omega_X)$ of a variable X. The inputs of the procedure are: t (root node of $\mathcal{BT}$); X (variable to restrict); S_X (set of states of X to restrict). The procedure starts from the root node t, traversing recursively the full structure to the leaves. At each node, the algorithm selects those branches consistent with S_X and discards the rest of them. In particular, the following cases can be considered depending on the variable and states labelling the current node t and the outgoing branches respectively:

- *Case A* (line 5): the current node is labelled with the target variable X and the left branch does not contain any of the states in S_X. Then, the current node is replaced by the resulting BT of restricting the right child to the set of states $L_{rb(t)} \cap S_X$. In doing so, the current node and the left branch are removed from the resulting BT.

- *Case B* (line 7): the current node is labelled with the target variable X and the left branch contains some of the states in S_X but the right branch does not. Then, the current node is replaced by the resulting BT of restricting the left child to the set of states $L_{lb(t)} \cap S_X$. In doing so, the procedure removes the current node and the right branch from the resulting BT.

- *Case C* (lines 9 to 12): the current node is labelled with the target variable X and the labels of both outgoing branches contain states in S_X. Then the states not present in S_X are removed from the new outgoing branches. The left child is replaced by the resulting BT of restricting the left child to the set of states $L_{lb(t)} \cap S_X$. Analogously, the right child is replaced by the resulting BT of restricting the left child to the set of states $L_{rb(t)} \cap S_X$.

- *Case D* (lines 15 and 16): the current node is not labelled with the target variable X. Then the algorithm is invoked on both children without changing current node.

Algorithm 9 Restriction

input : t (root node of $\mathcal{BT}$); X (variable to restrict); S_X (set of states of X to restrict)

output : The root of $\mathcal{BT}^{R(S_X)}$

```
 1: if t is not a leaf node then
 2:     if $L_t = X$ then
 3:         Set $S^l_X \leftarrow L_{lb(t)} \cap S_X$ and $S^r_X \leftarrow L_{rb(t)} \cap S_X$
 4:         if $S^l_X = \emptyset$ then                                ▷ Case A
 5:             return Restriction($t_r$,$X$,$S^r_X$)
 6:         else if $S^r_X = \emptyset$ then                           ▷ Case B
 7:             return Restriction($t_l$,$X$,$S^l_X$)
 8:         else                                                       ▷ Case C
 9:             $L_{lb(t)} \leftarrow S^l_X$                ▷ Sets the labels of both branches
10:             $L_{rb(t)} \leftarrow S^r_X$
11:             $t_l \leftarrow Restriction(t_l,X,S^l_X)$          ▷ the new left child of $t$
12:             $t_r \leftarrow Restriction(t_r,X,S^r_X)$         ▷ the new right child of $t$
13:         end if
14:     else                                                           ▷ Case D
15:         $t_l \leftarrow Restriction(t_l,X,S_X)$                      ▷ Sets children
16:         $t_r \leftarrow Restriction(t_r,X,S_X)$
17:     end if
18: end if
19: return $t$
```

As an example, Figure 8.1 shows the restriction of a BT to set of states $\{a_2, a_3\}$ of the variable A.

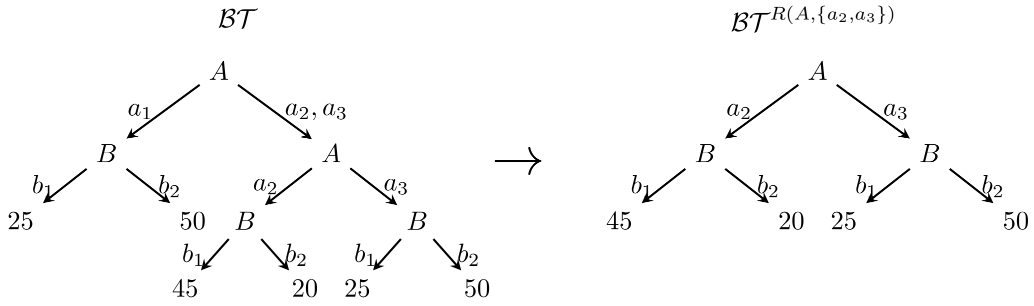


Figure 8.1: Restriction of a BT representing a UP to the set of states $\{a_2, a_3\}$ of the variable A.

8.2.2 Element-wise operations

Herein we will explain how to perform directly on BTs all the element-wise operations required for the ID evaluation: the two types of combinations (multiplication and addition), the division and the auxiliary maximum operations. Each of them can be seen as particularization of a generic combination operation, which is defined as follows.

Definition 41 (generic combination of BTs) *Let $\mathcal{BT}_1$ and $\mathcal{BT}_2$ be two binary trees defined on the sets of variables $\mathbf{X}_I$ and $\mathbf{X}_J$ respectively. Their generic combination $\mathcal{BT}_1 \otimes \mathcal{BT}_2$ is another binary tree over $\mathbf{X}_I \cup \mathbf{X}_J$ such that:*

$$(\mathcal{BT}_1 \otimes \mathcal{BT}_2)(\mathbf{x}) = f(\mathcal{BT}_1(\mathbf{x}^{\downarrow \mathbf{X}_I}), \mathcal{BT}_2(\mathbf{x}^{\downarrow \mathbf{X}_J})) \tag{8.1}$$

for each configuration $\mathbf{x} \in \Omega_{\mathbf{X}_I \cup \mathbf{X}_J}$, where f is a function from $\mathbb{R}^2$ on $\mathbb{R}$.

According to previous definition, each leaf node of the resulting BT is labelled with the result of the function f that depends on the particular operation we are carrying out. Table 8.1 shows these functions for each particular combination of two binary trees $\mathcal{BT}_1$ and $\mathcal{BT}_2$ whose root nodes are t_1 and t_2 respectively.

Operation	notation	f
$multiplication(t_1, t_2)$	$\mathcal{BT}_1 \cdot \mathcal{BT}_2$	$\mathcal{BT}_1(\mathbf{x}^{\downarrow \mathbf{X}_I}) \cdot \mathcal{BT}_2(\mathbf{x}^{\downarrow \mathbf{X}_J})$
$addition(t_1, t_2)$	$\mathcal{BT}_1 + \mathcal{BT}_2$	$\mathcal{BT}_1(\mathbf{x}^{\downarrow \mathbf{X}_I}) + \mathcal{BT}_2(\mathbf{x}^{\downarrow \mathbf{X}_J})$
$division(t_1, t_2)$	$\mathcal{BT}_1 / \mathcal{BT}_2$	$\mathcal{BT}_1(\mathbf{x}^{\downarrow \mathbf{X}_I}) / \mathcal{BT}_2(\mathbf{x}^{\downarrow \mathbf{X}_J})$
$maximum(t_1, t_2)$	$maximum(\mathcal{BT}_1, \mathcal{BT}_2)$	$\max(\mathcal{BT}_1(\mathbf{x}^{\downarrow \mathbf{X}_I}), \mathcal{BT}_2(\mathbf{x}^{\downarrow \mathbf{X}_J}))$

Table 8.1: Particularizations of the generic combination operation $\otimes$ and their corresponding functions f. For the division, convention $0/0 = 0$ is adopted.

The details of the recursive procedure implementing the *generic combination* operation $\otimes$ of two binary trees $\mathcal{BT}_1$ and $\mathcal{BT}_2$ are given in Algorithm 10. The inputs for the algorithm are t_1 and t_2, the root nodes of both trees. At each recursive call of the algorithm, the following cases are considered:

- *Case A* (line 4): when t_1 and t_2 are leaf nodes, a new node t_n labelled with the result of applying the corresponding function f is built.

- *Case B* (lines 6 to 10): If t_1 is a leaf node but t_2 is not, a new node t_n is built labelled with the variable in t_2. The left child of t_n is the result of the generic combination operation between t_1 and the left child of t_2. Similarly, the right child of t_n is the combination between t_1 with the right child of t_2. In doing so, the tree $\mathcal{BT}_2$ is traversed until both nodes are leaf nodes.

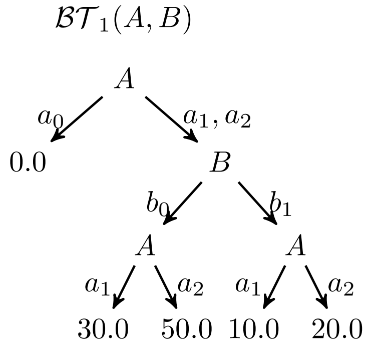
- *Case C* (lines 13 to 18): If t_1 is not a leaf node a new node t_n with the same label t_1 and the same labels $L_{lb(t_1)}$ and $L_{rb(t_1)}$ for its branches is built. The left child of t_n is the result of the generic combination operation between the left child of t_1 and the restriction of t_2 to the states of $L_{lb(t_1)}$. Similarly, the right child of the new tree is the result of the generic combination operation between the right child of t_1 and the restriction of t_2 to the states of $L_{rb(t_1)}$. In doing so, $\mathcal{BT}_1$ is traversed until a leaf node is reached.

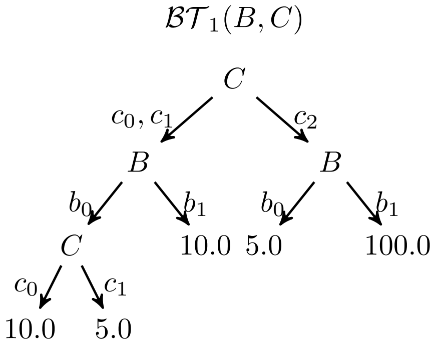
Algorithm 10 Generic Combination algorithm

```
input :  $t_1$ and $t_2$ (root nodes of $\mathcal{BT}_1$ and $\mathcal{BT}_2$)
output :  the root of $\mathcal{BT} = \mathcal{BT}_1 \otimes \mathcal{BT}_2$
Build a new node $t_n$
if $t_1$ is a leaf node then
    if $t_2$ is a leaf node then                                  ▷ Case A
        $L_{t_n} \leftarrow f(L_{t_1}, L_{t_2})$                  ▷ Sets the label of the leaf
    else                                                          ▷ Case B
        $L_{t_n} \leftarrow L_{t_2}$                              ▷ Sets the label of $t_n$
        $L_{lb(t_n)} \leftarrow L_{lb(t_2)}$                      ▷ Sets labels for both branches
        $L_{rb(t_n)} \leftarrow L_{rb(t_2)}$
        $t_{nl} \leftarrow$ combination($t_1, t_{2_l}$)           ▷ Sets children
        $t_{nr} \leftarrow$ combination($t_1, t_{2_r}$)
    end if
else                                                              ▷ Case C
    Let $X_i$ be the variable labelling $t_1$
    $L_{t_n} \leftarrow L_{t_1}$                                  ▷ Sets the label of $t_n$
    $L_{lb(t_n)} \leftarrow L_{lb(t_1)}$                          ▷ Sets the labels of both branches
    $L_{rb(t_n)} \leftarrow L_{rb(t1)}$
    $t_{nl} \leftarrow$ combination($t_{1_l}, \mathcal{BT}_2^{R(X_i, L_{lb(t_1)})}$)    ▷ Sets children
    $t_{nr} \leftarrow$combination($t_{1_r}, \mathcal{BT}_2^{R(X_i, L_{rb(t_1)})}$)
end if
return $t_n$
```

Note that, with the proposed algorithm, the structure of the resulting BT is the same regardless of the particular type of combination (only the values in the leaves are different). This situation can be seen in Example 33, where an example of each type of combination is shown. Additionally, the recursive process for multiplying two BTs is illustrated in Example 34.

Example 33 (generic combination of BTs) *Let us consider the binary trees $\mathcal{BT}_1$ and $\mathcal{BT}_2$ depicted below that represent two UPs.*





Then, the result of applying to $\mathcal{BT}_1$ and $\mathcal{BT}_2$ the operations of multiplication, addition, division and maximum is shown in Figures 8.2, 8.3, 8.4 and 8.5 respectively.

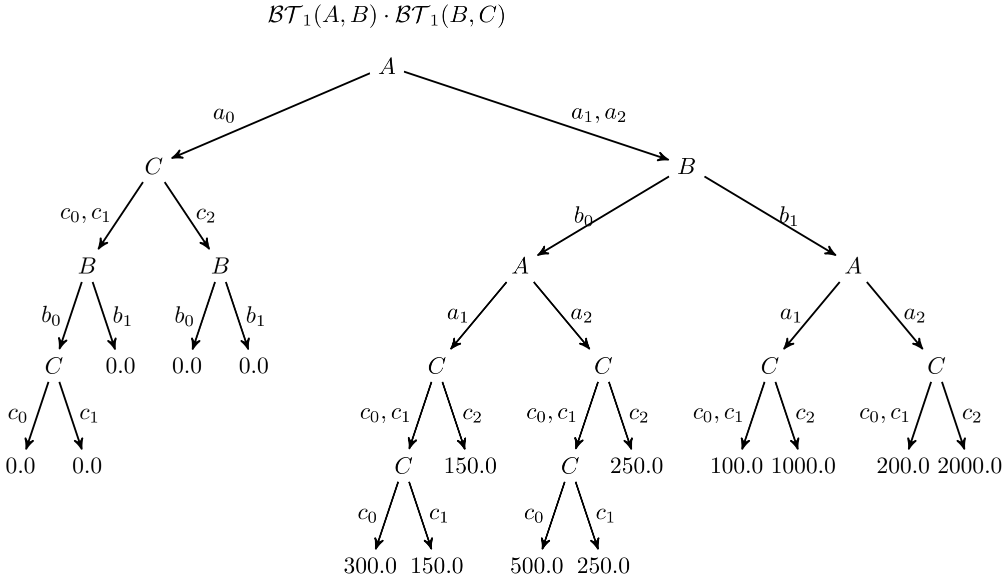


Figure 8.2: Multiplication of two BTs.

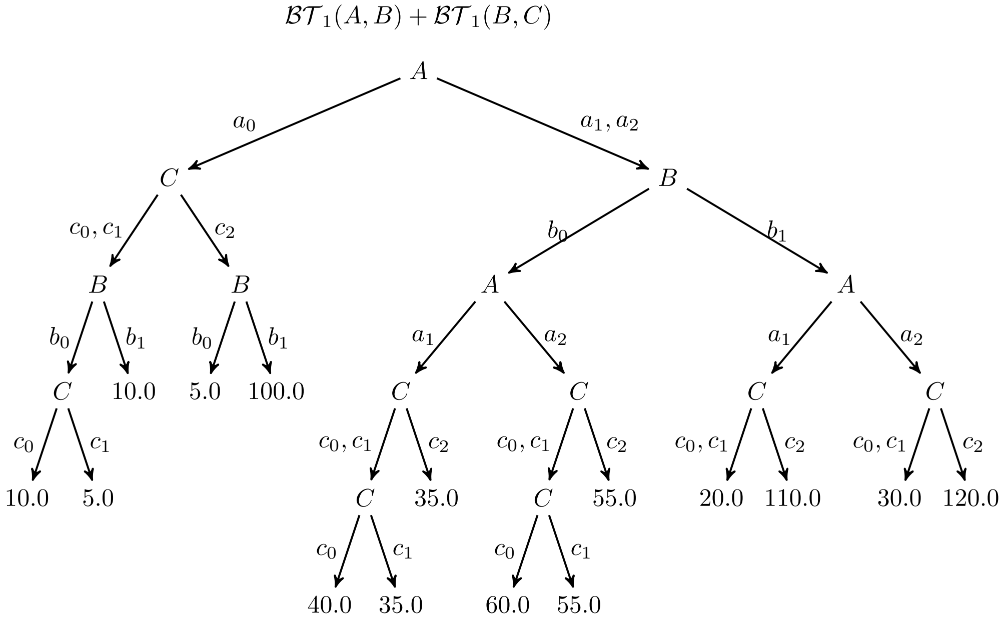


Figure 8.3: Addition of two BTs.

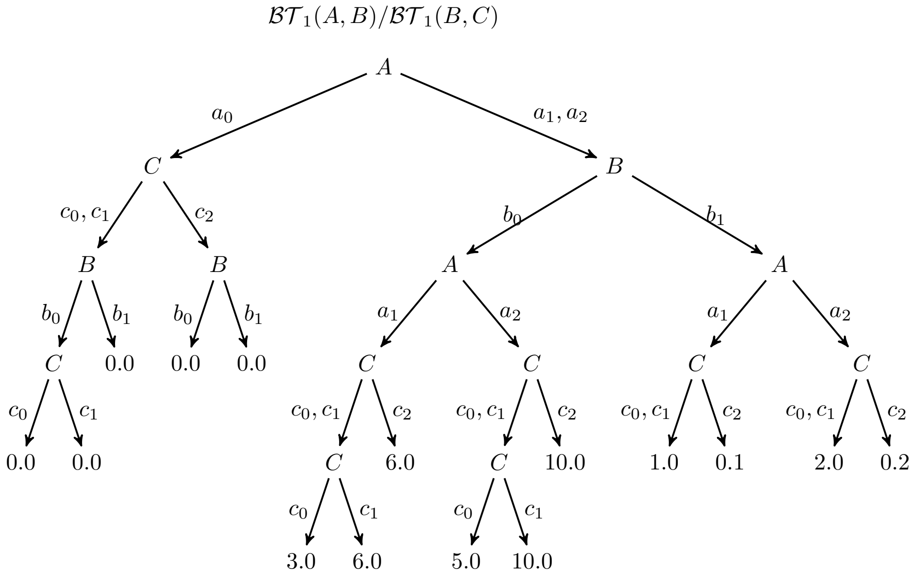


Figure 8.4: Division of two BTs.

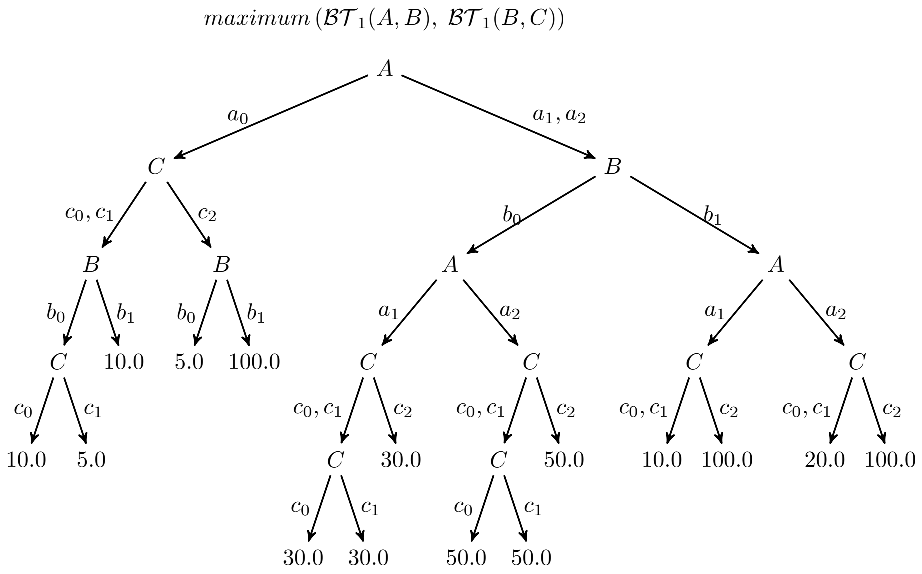
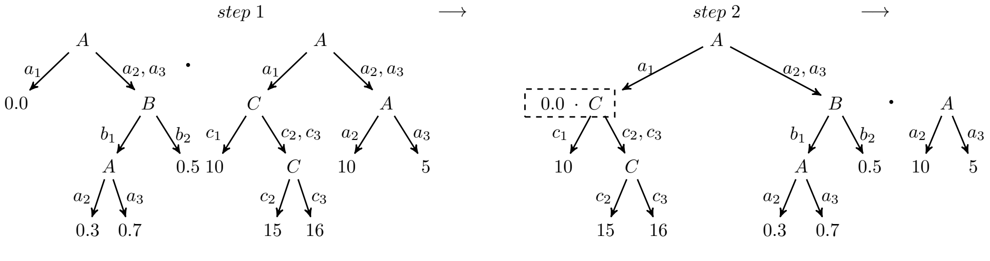
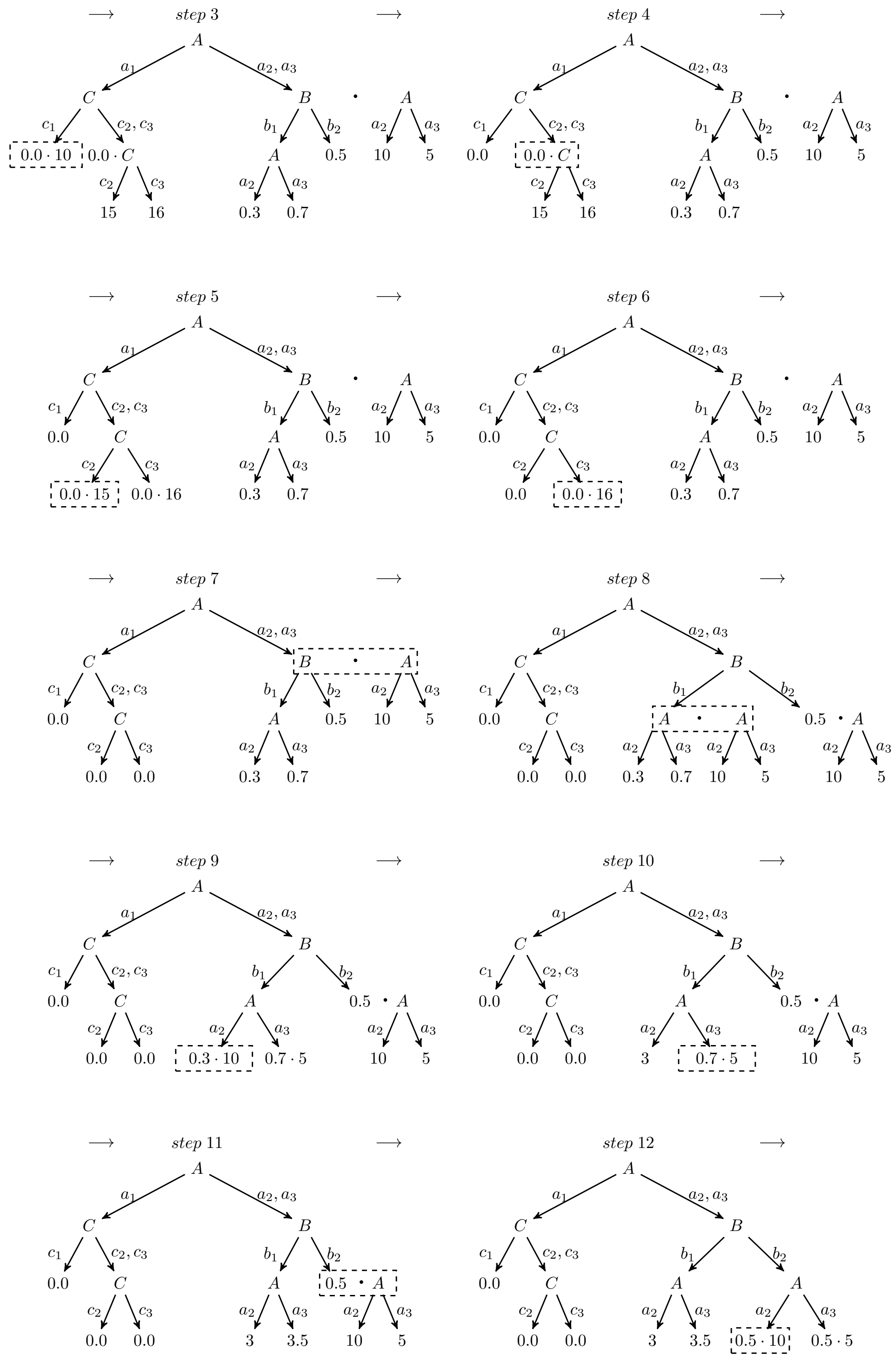
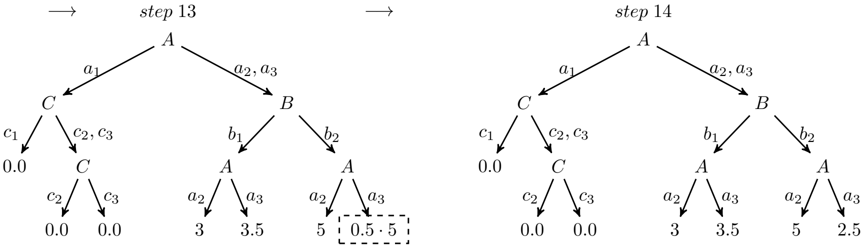


Figure 8.5: Maximum of two BTs.

Example 34 (multiplication of BTs) *Let us consider two binary trees $\mathcal{BT}_1$ and $\mathcal{BT}_2$ representing the potentials $\phi(A|B)$ and $\psi(A,C)$ respectively. If we apply Algorithm 10, the process for performing the combination $\mathcal{BT}_1 \cdot \mathcal{BT}_2$ is detailed below. Those nodes being processed by the algorithm at each step are highlighted with a dashed pattern.*




step 3
A
a1
a2, a3
C
B
•
A
c1
c2, c3
b1
b2
a2
a3
0.0 · 10
0.0 · C
A
0.5
10
5
c2
c3
a2
a3
15
16
0.3
0.7
step 4
0.0
0.0 · C
step 5
0.0 · 15
0.0 · 16
step 6
0.0 · 16
step 7
B • A
0.0
0.0
step 8
A • A
0.5 • A
0.3
0.7
10
5
step 9
0.3 · 10
0.7 · 5
step 10
3
0.7 · 5
step 11
3
3.5
0.5 • A
step 12
0.5 · 10
0.5 · 5



8.2.3 Marginalizations

Now we will explain how to perform on BTs the two types of marginalization used in the ID evaluation: sum-marginalization and max-marginalization. These two operations can be seen as particularization of a generic marginalization operator which is defined as follows.

Definition 42 (generic marginalization of a BT) *Let $\mathcal{BT}$ be a binary tree representing defined on the set of variables $\mathbf{X}_I$ and a variable $Y \in \mathbf{X}_I$ where $\Omega_Y = \{y_1, y_2, \ldots y_n\}$. The generic marginalization of Y out of $\mathcal{BT}$, denoted $\mathsf{M}_Y \mathcal{BT}$, is another binary tree defined over $\mathbf{X}_I \backslash \{Y\}$ such that:*

$$(\mathsf{M}_Y \mathcal{BT})(\mathbf{x}) = g(\mathcal{BT}(\mathbf{x}, y_1), \mathcal{BT}(\mathbf{x}, y_2), \ldots, \mathcal{BT}(\mathbf{x}, y_n)) \qquad (8.2)$$

for each configuration $\mathbf{x} \in \Omega_{\mathbf{X}_I \backslash \{Y\}}$, where g is a function from $\mathbb{R}^n$ on $\mathbb{R}$.

Table 8.2 shows the functions g for carrying out each particular marginalization of a variable Y out of a binary tree $\mathcal{BT}$ whose root node is t.

Operation	notation	g
$sum\text{-}marginalization(t, Y, \lvert\Omega_Y\rvert)$	$\sum_Y \mathcal{BT}$	$\mathcal{BT}(\mathbf{x}, y_1) + \mathcal{BT}(\mathbf{x}, y_2) + \cdots + \mathcal{BT}(\mathbf{x}, y_n)$
$max\text{-}marginalization(t, Y)$	$\max_Y \mathcal{BT}$	$\max(\mathcal{BT}(\mathbf{x}, y_1), \mathcal{BT}(\mathbf{x}, y_2), \ldots, \mathcal{BT}(\mathbf{x}, y_n))$

Table 8.2: Particularizations of the generic marginalization operation M and their corresponding functions g.

Algorithm 11 describes the generic marginalization $\sqcap\!\!\!\!\vee$ of a variable Y out of a binary tree $\mathcal{BT}$. This algorithm must be called using $|\Omega_Y|$ as the input parameter k. In recursive calls to the algorithm, k will be set to the number of available states of Y at the current node of the tree. This algorithm is recursively executed until a node labelled with the variable to be removed is found or a leaf node is reached. In particular, the proposed procedure considers the following cases:

- *Case A* (lines 2 to 7): when a leaf node is reached, the new leaf node is returned whose value depends on the specific type of marginalization: for the max-marginalization this node takes the value L_t and for the sum-marginalization takes the value $L_t \cdot k$.

- *Case B* (lines 9 to 16): a node labelled with the variable to be removed is reached, the algorithm marginalizes the left and right children trees and combines both them. The type of this combination depends on the type of marginalization: addition is used when sum-marginalizing while maximum is used when max-marginalizing.

- *Case C* (lines 18 to 23) : If the variable L_t labelling the current node t differs from the variable to be removed Y (lines 18 to 23), a new node labelled with L_t is built. Their children are then marginalized.

Note that, using the algorithm proposed, the structure of the resulting BT is the same regardless of the particular type of marginalization (only the values in the leaves are different). This situation can be seen in Example 35, where an example of each type of marginalization is shown.

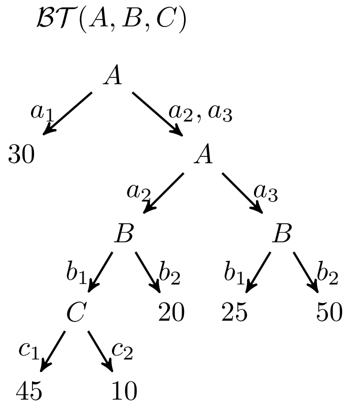
Algorithm 11 Generic Marginalization algorithm

input : t (root node of $\mathcal{BT}$); Y (variable to remove); k (a factor for multiplying the labels of leaf nodes)

output : the root of $\mathsf{M}_Y\mathcal{BT}$

```
Build a new node $t_n$
if $t$ is a leaf node then                                          ▷ Case A
    if sum-marginalization then
        $L_{t_n} \leftarrow L_t \cdot k$
    else
        $L_{t_n} \leftarrow L_t$                                    ▷ max-marginalization
    end if
else
    if $L_t = Y$ then                                               ▷ Case B
        $t_1 \leftarrow$ marginalization($t_l$, $Y$, $|L_{lb(t_n)}|$)    ▷ Make recursive calls
        $t_2 \leftarrow$ marginalization($t_r$, $Y$, $|L_{lr(t_n)}|$)
        if sum-marginalization then
            $t_n \leftarrow addition(t_1, t_2)$
        else
            $t_n \leftarrow maximum(t_1, t_2)$
        end if
    else                                                            ▷ Case C
        $L_{t_n} \leftarrow L_t$                                    ▷ Sets the label of the new node
        $(L_{lb(t_n)}, L_{lr(tn)}) \leftarrow (L_{lb(t)}, L_{lr(t)})$    ▷ Sets the label of branches
        $t_{nl} \leftarrow$ marginalization($t_l, Y, k$)           ▷ Make recursive calls on children
        $t_{nr} \leftarrow$ marginalization($t_r, Y, k$)
    end if
end if
return $t_n$
```

Example 35 (generic marginalization of BTs) *Let us consider the binary tree $\mathcal{BT}$ depicted below that represents a UP.*



Then, the result of applying to $\mathcal{BT}$ the operations of sum-marginalization and max-marginalization is shown in Figure 8.6.

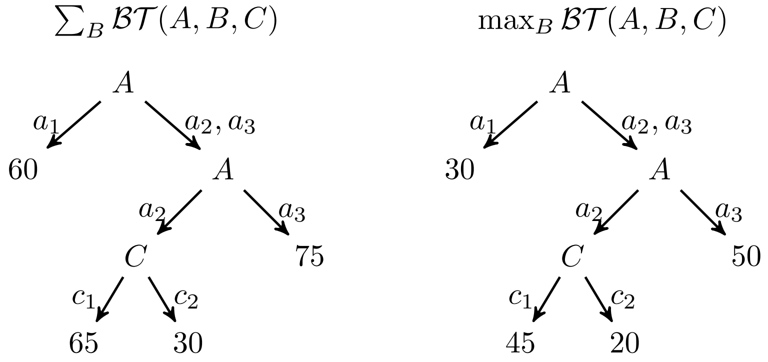


Figure 8.6: Sum-marginalization (left) and max-marginalization (right) of variable A in the same BT.

8.2.4 Complexity analysis

The complexity of the operations on potentials depends on the size of the structure used to represent them. That is, the number of values in a table or the number of nodes in the completely expanded tree (internal nodes and leaves).

Let $\phi(X_1, X_2, \ldots X_m)$ be a potential with n variables. Assuming that all the variables have the same number of states, i.e. $k = |\Omega_{X_1}| = |\Omega_{X_2}| = \cdots = |\Omega_{X_m}|$, the size of a table $\mathcal{T}$, representing this potential is given by Equation (8.3), which corresponds with the size of the Cartesian product of the variable domains (i.e. the total number of configurations).

$$size(\mathcal{T}) = |\Omega_{X_1}| \cdot |\Omega_{X_2}| \cdot \ldots \cdot |\Omega_{X_m}| = k^m \tag{8.3}$$

In the level 0 of a $\mathcal{NT}$ representing ϕ there is only one node (the root). In a level i with $i = 1, \ldots, m$, there are k^i nodes. Then, the size of $\mathcal{NT}$:

$$size(\mathcal{NT}) = 1 + \sum_{i=1}^{m} k^i = \frac{k^{m+1} - 1}{k - 1} \tag{8.4}$$

A completely expanded $\mathcal{BT}$ representing ϕ has k^m leaves (number of values in ϕ). Thus, from Theorem 2 in page 98 we obtain:

$$size(\mathcal{BT}) = 2 \cdot k^m - 1 \tag{8.5}$$

It can be observed that in general the size of BTs is higher than NTs or tables but the complexity of traversing these representations is similar: exponential $\mathcal{O}(k^m)$. However, as BTs allow a more compact representation encoding more context-specific independencies, it is expected a certain benefit when pruning operations are performed. This point is demonstrated empirically in Section 8.4.

8.3 ID evaluation algorithms with BTs

Evaluation algorithms for IDs can be easily adapted for working with BTs. The global structure of the algorithms is not changed: the main difference is that they require an initialization phase where initial BTs are built from tables and pruned in order to obtain smaller trees. BTs are built and pruned using the procedures explained in Sections 5.4.1 and 5.4.2 respectively. Once all the potentials are transformed into BTs and pruned, the evaluation algorithms are quite similar to the algorithms that use tables: computation is done using operations for BTs (Section 8.2), instead of their counterparts for tables. Herein three ID evaluation algorithms from the literature are described for working with BTs: *variable elimination*, *lazy evaluation* and *symbolic probabilistic inference*.

8.3.1 Variable elimination

The VE algorithm is one of the most common methods used for evaluating IDs. As a reminder of Section 4.5.1, its general scheme for evaluating an ID regardless of the potential representation is as follows: it starts with a set of potentials and it eliminates all the variables one by one. All the variables must be removed in reverse order of the information precedence given by $\prec$. Secondly, chance variables are removed using *sum-marginalization* whereas for decisions *max-marginalization* is used. That is, it first sum-marginalizes $\mathcal{I}_n$, then max-marginalizes D_n, sum-marginalizes $\mathcal{I}_{i-1}$, etc.

The VE algorithm, adapted for working with BTs, is shown in Algorithm 12. We assume that, in the specification of the ID, potentials are initially represented using tables. Thus, before the removal loop, an initialization phase is included where the potentials are transformed into pruned BTs using the methods explained in Sections 5.4.1 and 5.4.2 respectively. The algorithm contains an additional parameter, namely ε, which is the pruning threshold. We will use the same value of ε for probabilities and utilities. Yet, different thresholds for PPs and UPs could be used. In line 2 for PPs and 6 for UPs, the exact BTs are built. In lines 3 and 7 they are pruned and replaced in the potential sets Φ or Ψ (i.e. the sets of PPs and UPs in the ID).

Algorithm 12 Variable Elimination with BTs

input : Φ, Ψ (sets of potentials in the ID), $\{\mathcal{I}_0, D_1, \mathcal{I}_1, \ldots, D_n, \mathcal{I}_n\}$ (partitions of nodes in the ID), ε (pruning threshold).

```
    /*Initialization phase*/
 1: for all $\phi \in \Phi$ do
 2:     Build a sorted binary tree $\mathcal{BT}^{\phi}$ representing $\phi$
 3:     $\Phi \leftarrow \Phi \backslash \{\phi\} \cup \{prune(\mathcal{BT}^{\phi},\ \varepsilon)\}$
 4: end for
 5: for all $\psi \in \Psi$ do
 6:     Build a sorted binary tree $\mathcal{BT}^{\psi}$ representing $\psi$
 7:     $\Psi \leftarrow \Psi \backslash \{\psi\} \cup \{prune(\mathcal{BT}^{\psi},\ \varepsilon)\}$
 8: end for
    /*Removal Loop*/
 9: for $k \leftarrow n$ to $0$ do
10:     while $\mathcal{I}_k \neq \emptyset$ do
11:         Select $X \in \mathcal{I}_k$                          ▷ Pick a chance variable to eliminate
12:         $(\Phi, \Psi) \leftarrow \mathrm{ElimVarBT}(X, \Phi, \Psi)$   ▷ Chance variable elimination
                                                                  (Algorithm 13)
13:         $\mathcal{I}_k \leftarrow \mathcal{I}_k \backslash \{X\}$
14:     end while
15:     if $k > 0$ then
16:         $(\Phi, \Psi) \leftarrow \mathrm{ElimVarBT}(D_k, \Phi, \Psi)$   ▷ Decision variable elimination
                                                                  (Algorithm 13)
17:     end if
18: end for
```

Now potentials in Φ and Ψ are represented as BTs and, as a consequence, operations over BTs must be used in the removal loop. Note that the algorithm *ElimVarBT* is invoked in lines 12 and 16 for removing a chance and a decision variable respectively. This procedure, which is detailed in Algorithm 13, is similar to the one explained for the standard version of VE (see Algorithm 2, page 2) but we explicitly show that operations on potentials are performed over BTs. Note that BTs are not pruned nor sorted after each removal: this would introduce an important overhead and a large error in the approximation.

Algorithm 13 ElimVarBT - Elimination of a single variable

```
input :  $Y$ (variable to remove), $\Phi$, $\Psi$ (sets of current potentials)
output :  $\Phi$, $\Psi$ (updated sets of current potentials without $Y$)
$\Phi_Y \leftarrow \{\mathcal{BT}^{\phi} \in \Phi | Y \in dom(\mathcal{BT}^{\phi})\}$    ▷ Select
$\Psi_Y \leftarrow \{\mathcal{BT}^{\psi} \in \Psi | Y \in dom(\mathcal{BT}^{\psi})\}$
$\mathcal{BT}^{\phi_Y} \leftarrow \prod_{\mathcal{BT}^{\phi} \in \Phi_Y} \mathcal{BT}^{\phi}$    ▷ Combine
$\mathcal{BT}^{\psi_Y} \leftarrow \sum_{\mathcal{BT}^{\psi} \in \Psi_Y} \mathcal{BT}^{\psi}$
if $Y \in \mathcal{U}_C$ then
    $(\mathcal{BT}^{\phi'_Y}, \mathcal{BT}^{\psi'_Y}) \leftarrow (\sum_Y \mathcal{BT}^{\phi_Y}, \frac{\sum_Y \mathcal{BT}^{\phi_Y} \cdot \mathcal{BT}^{\psi_Y}}{\sum_Y \mathcal{BT}^{\phi_Y}})$    ▷ Remove by sum
else
    $(\mathcal{BT}^{\phi'_Y}, \mathcal{BT}^{\psi'_Y}) \leftarrow ((\mathcal{BT}^{\phi_Y})^{R(Y=y)}, \max_Y \mathcal{BT}^{\psi_Y})$    ▷ Remove by max
    $\widehat{\delta}_Y \leftarrow \arg\max_Y \mathcal{BT}^{\psi_Y}$    ▷ Optimal policy
end if
$(\Phi, \Psi) \leftarrow (\Phi \backslash \Phi_Y \cup \{\mathcal{BT}^{\phi'_Y}\}, \Psi \backslash \Psi_Y \cup \{\mathcal{BT}^{\psi'_Y}\})$    ▷ Update
return $(\Phi, \Psi)$
```

8.3.2 Lazy evaluation

As explained in Section 4.5.3, LE is an evaluation algorithm based on message passing in a *strong junction tree*, which is a representation of an ID built by moralization and by triangulating the graph using a strong elimination order [68]. Nodes in the strong junction tree correspond to *cliques* (maximal complete sub-graphs) of the triangulated graph. Two neighbour cliques are connected by a separator which contains the intersection of the variables in both cliques. Initially, each potential is associated to the closest clique to the root containing all its variables. Propagation is performed by message-passing from the leaves to the root. A message consists on a list of potentials from which variables not present in the parent separator has been removed using the VE algorithm.

The general scheme of LE for working with BTs is shown in Algorithm 14. This adaptation includes also an initialization phase (lines 1 to 9) where potentials are transformed into trees and pruned with a given threshold ε passed as a parameter. For the message computation, operations with BTs are used instead of their counterparts for tables. Even though the potentials are transformed into BTs before building the strong junction tree, this auxiliary structure is the same regardless of whether potentials are represented using tables or BT.

Algorithm 14 Lazy Evaluation with BTs

input : Φ, Ψ (sets of potentials in the ID), $\{\mathcal{I}_0, D_1, \mathcal{I}_1, \ldots, D_n, \mathcal{I}_n\}$ (partitions of nodes in the ID), ε (pruning threshold).

```
/*Initialization phase*/
for all $\phi \in \Phi$ do
    Build a sorted binary tree $\mathcal{BT}^{\phi}$ representing $\phi$
    $\Phi \leftarrow \Phi \backslash \{\phi\} \cup \{prune(\mathcal{BT}^{\phi}, \varepsilon)\}$
end for
for all $\psi \in \Psi$ do
    Build a sorted binary tree $\mathcal{BT}^{\psi}$ representing $\psi$
    $\Psi \leftarrow \Psi \backslash \{\psi\} \cup \{prune(\mathcal{BT}^{\psi}, \varepsilon)\}$
end for
Build a strong junction tree with root $C_1$ from the ID
/*Propagation phase*/
Associate each potential in $\Phi \cup \Psi$ to the closest clique to the root containing all its variables
Invoke *CollectMessage* in $C_1$                                  ▷ Algorithm 5
```

8.3.3 Symbolic probabilistic inference

In this dissertation, a new ID evaluation algorithm will be proposed, namely the SPI algorithm (see Chapter 12). In short, this method tries to find the optimal order for the combinations and marginalizations by choosing at each step the best operation. For evaluating IDs, as VE does, SPI removes all variables in the decision problem in reverse order of the partial ordering imposed by the information constraints. Yet, VE is guided by an elimination order while SPI is guided by a combination order. That is, SPI chooses the next pair of potentials to combine and eliminate the variables when possible. In this sense SPI is finer grained than VE.

For adapting this method for working with BTs (see Algorithm 15), we will proceed like in previous algorithms: we include an initialization phase for building the trees and use the corresponding operations with BTs.

Algorithm 15 SPI-algorithm with BTs

input : Φ, Ψ (sets of potentials in the ID), $\{\mathcal{I}_0, D_1, \mathcal{I}_1, \ldots, D_n, \mathcal{I}_n\}$ (partitions of nodes in the ID), ε (pruning threshold).

```
/*Initialization phase*/
for all $\phi \in \Phi$ do
    Build a sorted binary tree $\mathcal{BT}^{\phi}$ representing $\phi$
    Prune $\mathcal{BT}^{\phi}$ using $\varepsilon$ as threshold
    $\Phi \leftarrow \Phi \backslash \{\phi\} \cup \{\mathcal{BT}^{\phi}\}$
end for
for all $\psi \in \Psi$ do
    Build a sorted binary tree $\mathcal{BT}^{\psi}$ representing $\psi$
    Prune $\mathcal{BT}^{\psi}$ using $\varepsilon$ as threshold
    $\Psi \leftarrow \Psi \backslash \{\psi\} \cup \{\mathcal{BT}^{\psi}\}$
end for
/*Removal loop*/
for $k \leftarrow n$ to $0$ do
    $(\Phi_{\mathbf{X}}, \Psi_{\mathbf{X}}) \leftarrow (\{\phi \in \Phi | dom(\phi) \cap \mathcal{I}_k \neq \emptyset\},\ \{\psi \in \Psi | dom(\psi) \cap \mathcal{I}_k \neq \emptyset\})$
    $(\Phi, \Psi) \leftarrow (\Phi \backslash \Phi_{\mathbf{X}},\ \Psi \backslash \Psi_{\mathbf{X}})$
    $(\Phi'_{\mathbf{X}},\ \Psi'_{\mathbf{X}}) \leftarrow RemoveChanceSet(\mathcal{I}_k, \Phi_{\mathbf{X}}, \Psi_{\mathbf{X}})$        ▷ Algorithm 24
    $(\Phi, \Psi) \leftarrow (\Phi \cup \Phi'_{\mathbf{X}},\ \Psi \cup \Psi'_{\mathbf{X}})$
    if $k > 0$ then
        $(\Phi_D, \Psi_D) \leftarrow (\{\phi \in \Phi | D_k \in dom(\phi)\},\ \{\psi \in \Psi | D_k \in dom(\psi)\})$
        $(\Phi, \Psi) \leftarrow (\Phi \backslash \Phi_D,\ \Psi \backslash \Psi_D)$
        $(\Phi'_D,\ \psi'_D) \leftarrow RemoveDecision(D_k, \Phi_D, \Psi_D)$        ▷ Algorithm 26
        $(\Phi, \Psi) \leftarrow (\Phi \cup \Phi'_D,\ \Psi \cup \{\psi'_D\})$
    end if
end for
```

8.4 Experimental work

In this section, the performance of NTs and BTs for IDs inference is analysed. However, we must introduce first some concepts about *multi-objective optimization problems*.

8.4.1 Multi-objective optimization problems

When performing approximate evaluation of IDs there are two objectives to consider: time and error of the approximation. These two objectives can be controlled with the threshold for pruning, ε. The use of low values of ε will produce slow evaluations with low errors. On the hand, high values will lead to fast evaluations and big errors. Thus, the problem of finding the best approximation can be considered as a multi-objective optimization problem (MOP) [66] with two objectives to be minimized. Hence, optimizing means finding a solution with acceptable values for all the objectives.

There are several possible solutions for this optimization problem as there is not additional information about the preferred objective. In MOPs the set of acceptable solutions composes the Pareto set (non-dominated solutions). In order to compare two solution sets (produced by two different representations of potentials) we have used the *hyper-volume* indicator [117]. It is based on representing the solution set in a n-dimensional space being n the number of objectives. In this space a reference point r characterizes the worst possible solution. In our case, the axis of the space represents computation time and the error produced by the approximation. The hyper-volume indicator is an unary value measuring the percentage of area dominated by a certain set of solutions (Pareto-set). Its maximum value is 1 and corresponds to a solution set dominating the rest of possible solutions (the best one). Figure 8.7 shows an example of a Pareto-set containing six different solutions represented by x_1, x_2, x_3, x_4, x_5, and x_6. The hyper-volume measures the portion of area in grey.

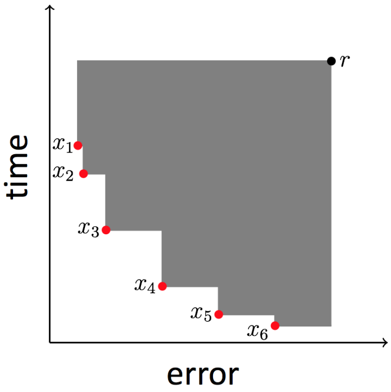


Figure 8.7: Hyper-volume for a minimization problem.

8.4.2 Objectives and procedure

There are several objectives related to the set of experiments performed in this chapter. Given a certain set of IDs whose potentials are represented as BTs, NTs and tables we try:

- To test if the representation as BTs requires less memory space than using NTs or tables. The gain in memory space should imply two benefits: a reduction in the computation time as well as the ability to evaluate more complex models.
- To check what representation gives better performance (respect to memory space, time and error) when computing approximate solutions varying ε.

For the empirical validation, a set of 15 IDs from the literature are used: NHL and Jaundice are two real world IDs used for medical purposes [76, 97]; Appendicitis and ChestClinic are two synthetic diagrams modelling a medical decision problems [69]; two versions of the oil wildcatter's [96, 40, 55]; an ID representing the Car Buyer problem [94]; two IDs representing a decision problem in the poker game [64]; two different IDs used at agriculture for treating mildew [64]; an ID for solving the maze problem [113]; Competitive Asymm. and Theat of Entry are two IDs from the business domain. An ID modelling the reactor decision problem as described by Bielza and Shenoy [3, 4]. The details of these IDs are shown in

Table 8.3, which contains the number of nodes of each kind, the size of the largest partition $\mathcal{I}_i$ of chance nodes and the average potential size.

ID	$\|\mathcal{U}_C\|$	$\|\mathcal{U}_D\|$	$\|\mathcal{U}_V\|$	$\max\|\mathcal{I}_i\|$	average potential size
Appendicitis	4	1	1	2	3.6
Car Buyer	3	3	1	1	64.5
ChestClinic	8	2	2	5	5.2
Competitive Asymm.	10	9	1	10	35.182
Jaundice	21	2	1	10	41.5
Maze	14	2	1	6	3100.2
Mildew 1	6	1	2	6	28.375
Mildew 4	7	2	2	4	32.222
NHL	17	3	1	11	468.111
Oil	2	2	2	1	7.25
Oil Split Costs	2	2	3	1	6.2
Poker	7	1	1	7	94
Poker Extended	9	3	1	4	129.3
Reactor	3	2	3	2	7.667
Threat of Entry	3	9	1	3	46

Table 8.3: Features of the IDs used in the experimentation. More details of these IDs are given in Appendix B.2.

Note that the IDs used for the experimentation must be simple enough to allow an exact evaluation as well. This is the only way to compute the error introduced in the approximate solutions. The analysis of the results is divided in two parts. First, in Section 8.4.3 we analyse in detail the evaluation of the NHL. Secondly, the results obtained with the rest of IDs is analysed in Section 8.4.4.

For the experiments, each ID is evaluated using tables, NTs and BTs with different values of ε in the interval $[0, 1]$. The inference algorithms employed are VE, LE, and SPI (see Section 8.3). For each evaluation it is measured:

- The size of all the potentials[1] is measured before and after the removal of each variable (or a set of variables for the SPI algorithm). That is:

$$totalPotSize_{NT} = \sum_{\phi \in \Phi} size(\mathcal{NT}^{\phi}) + \sum_{\psi \in \Psi} size(\mathcal{NT}^{\psi}) \tag{8.6}$$

$$totalPotSize_{BT} = \sum_{\phi \in \Phi} size(\mathcal{BT}^{\phi}) + \sum_{\psi \in \Psi} size(\mathcal{BT}^{\psi}) \tag{8.7}$$

$$totalPotSize_{tables} = \sum_{\phi \in \Phi} \prod_{X_i \in dom(\phi)} |\Omega_{X_i}| \; + \sum_{\psi \in \Psi} \prod_{X_j \in dom(\psi)} |\Omega_{X_j}| \tag{8.8}$$

where Φ and Ψ are the set of PPs and UPs in a given evaluation stage. The reduction in memory space requirements for NTs and BTs is computed as follows.

$$spaceSavings_{NT} = 1 - \frac{totalPotSize_{NT}}{totalPotSize_{tables}} \tag{8.9}$$

$$spaceSavings_{BT} = 1 - \frac{totalPotSize_{BT}}{totalPotSize_{tables}} \tag{8.10}$$

- The gain respect to computation time is computed with Equation (8.11). It should be noticed that the evaluation time with trees also includes the time required for building the trees from tables and pruning them.

$$speedup = \frac{time_{tables}}{time_{trees}} \tag{8.11}$$

- To analyse the error produced by the approximation, the MEU is calculated using trees and tables (see Equation (4.26)). Then, the absolute error is computed with Equation (8.12) . The error is analysed together with the computation time: all the pairs $(meanPotSize_{trees}, absoluteError)$ for the

[1] The size of a potential represented as a table corresponds with its number of entries. In case of trees, it is the number of nodes (internal ones and leaves).

same kind of representation compose a solution set. Pareto front and hypervolume are computed for every solution set.

$$absoluteError = |MEU_{trees}(\widehat{\Delta}) - MEU_{tables}(\widehat{\Delta})| \tag{8.12}$$

8.4.3 Results for the NHL ID

8.4.3.1 Storage requirements and computation time

Figures 8.8, 8.9 and 8.10 show the storage requirements for handling all the potentials during the evaluation of the NHL ID using the algorithms VE, LE and SPI respectively. Additionally, for each algorithm and data structure, two different values of ε are compared. The vertical axis represents the storage size (i.e., see Equations 8.6, 8.7 and 8.8) using a logarithmic scale. The horizontal axis indicates the evaluation stages. The corresponding space savings are shown in Table 8.4. It can be observed that, using any of the three algorithms, less space is needed with trees (NTs and BTs) than with tables. As long as ε is increased the memory space reduction is more noticeable. We can also observe that, in the final evaluation stages, there are not noticeable differences in storage requirements. This may be due to the combination of the potentials producing another ones where the effect of the initial prune is lost. If we compare the space savings obtained with both kinds of trees, the reduction is higher with BTs than with NTs. Similar space savings values are obtained for all the evaluation algorithms analysed.

	VE		LE		SPI	
	$\varepsilon = 0.0$	$\varepsilon = 0.05$	$\varepsilon = 0.0$	$\varepsilon = 0.05$	$\varepsilon = 0.0$	$\varepsilon = 0.05$
NTs	0.318	0.959	0.431	0.965	0.55	0.924
BTs	**0.525**	**0.997**	**0.666**	**0.997**	**0.592**	**0.98**

Table 8.4: Space savings that results from using trees (NTs and BTs) instead of tables during NHL ID evaluation with two different ε thresholds values and the algorithms VE, LE and SPI.

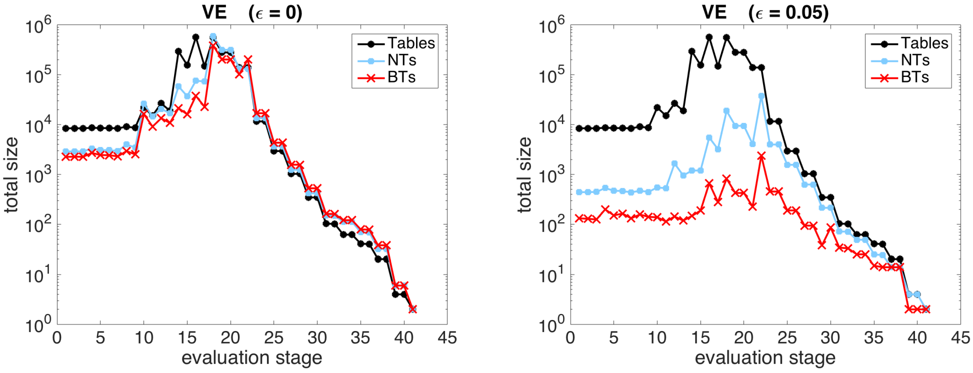


Figure 8.8: Size of the potentials during the NHL ID evaluation with tables, NTs and BTs with two different ε threshold values and the VE algorithm.

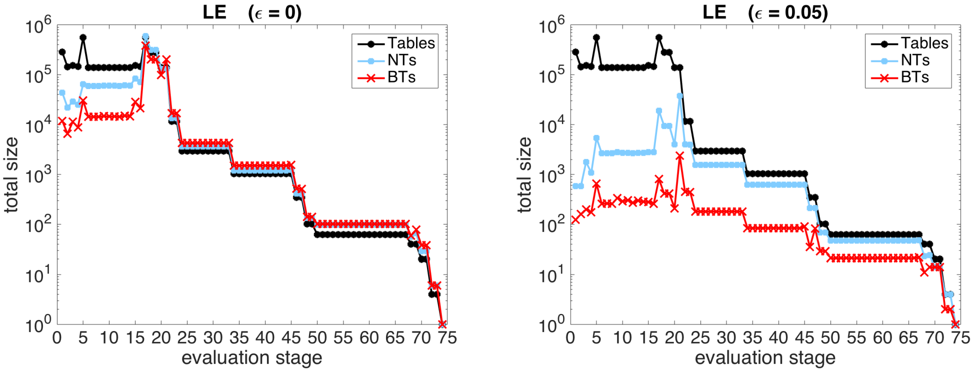


Figure 8.9: Size of the potentials during the NHL ID evaluation with tables, NTs and BTs with two different ε threshold values and the LE algorithm.

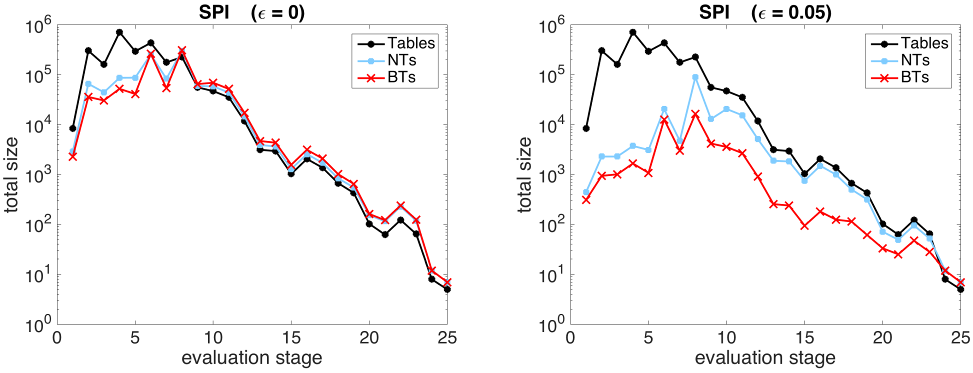


Figure 8.10: Size of the potentials during the NHL ID evaluation with tables, NTs and BTs with two different ε threshold values and the SPI algorithm.

Figures 8.11, 8.12 and 8.13 show the computation time with different ε values. Since the evaluation time using tables is much higher than using trees, this is not shown in the graphic. The evaluation time for each algorithm with tables is approximately 15900 ms, 12040 ms and 9970 ms. As expected, the reduction in the size of the potentials implies a reduction in the computation time: for all the algorithms proposed, the evaluation with BTs is faster that with NTs or tables. It can be observed that the computation with BTs requires less time with a decreasing reduction as long as ε value increases. Perhaps this can be explained due to the behaviour of pruning operation on NTs: the values for all the states are collapsed into a single one. That is, this operation is more drastic on NTs although more error will be introduced in the solutions. If the speed up values obtained with each algorithm are compared, it can be observed that the highest improvements are obtained with the VE algorithm for BTs. The reason for that is that the VE algorithm for tables is the slowest one. We can also observe that the computation time required by LE with trees is higher than the required by VE: the reason for that is that LE has an overhead due to the time needed for building the strong junction tree (which is independent of the potential representation). Thus, another conclusion is that the use of BTs makes the VE algorithm faster than LE.

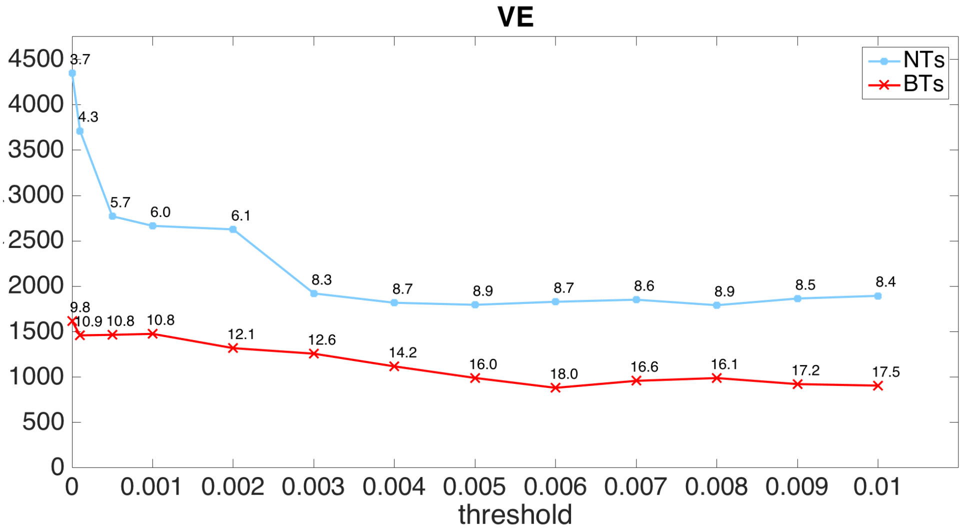


Figure 8.11: Evaluation time and speed up obtained during NHL ID evaluation with NTs and BTs and different values for ε with the VE algorithm. The evaluation time with tables is approximately 15900 ms.

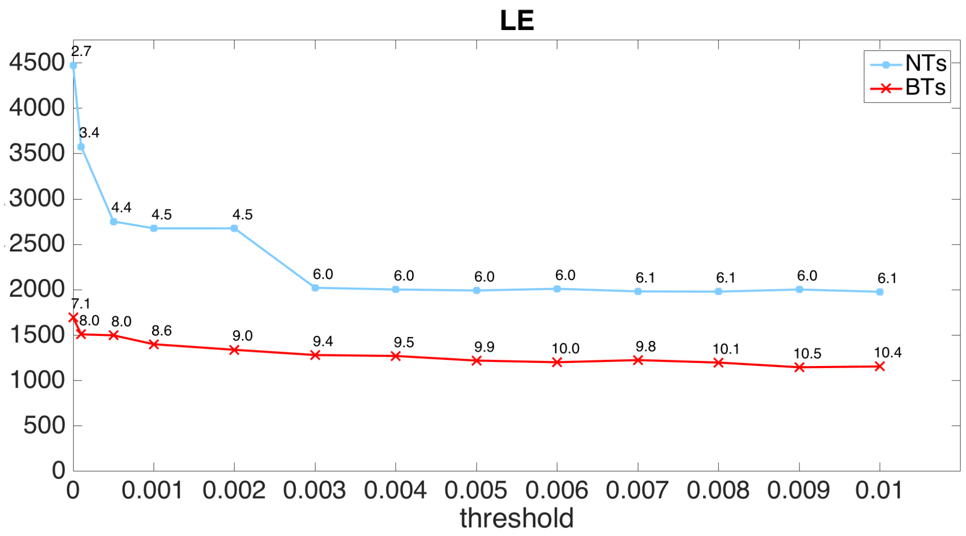


Figure 8.12: Evaluation time and speed up obtained during NHL ID evaluation with NTs and BTs and different values for ε with the LE algorithm. The evaluation with tables is approximately 12040 ms.

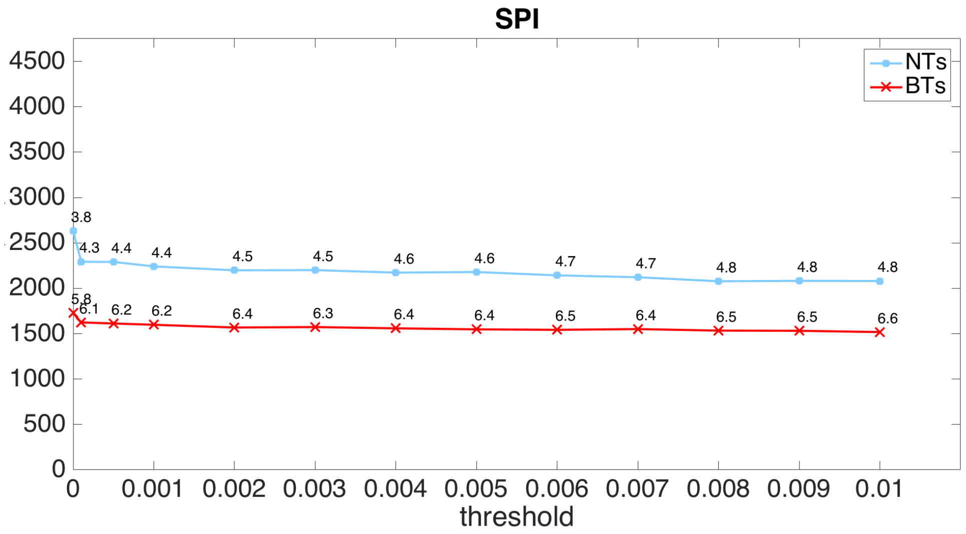


Figure 8.13: Evaluation time and speed up obtained during NHL ID evaluation with NTs and BTs and different values for ε with the SPI algorithm. The evaluation time with tables is approximately 9970 ms.

8.4.3.2 Error against time

The experiments reveal that, using any of the algorithms proposed, BTs produce better approximations than NTs: the same error level is achieved requiring less time. This situation can be shown in Figures 8.14, 8.15 and 8.16. For each algorithm, it is shown one graphic showing a comparison of the absolute error versus the running time required for computing the MEU of the NHL ID. Each point corresponds to a different evaluation with certain value of ε. It can be observed that the dominated area is always bigger for BTs.

All the pairs $(time_{trees}, absoluteError)$ for the same representation compose a solution set. For each solution set the Pareto front and hyper-volume are computed using the reference point r. These numbers are included in Table 8.5. Each column corresponds to one evaluation algorithm whereas each row is used for every kind of potential: NT and BT. It can be observed that the higher hyper-volume values using BTs (H_{BT}) are always larger (better) than those using NTs (H_{BT}). Thus, we conclude that better approximations are achieved using BTs for this ID.

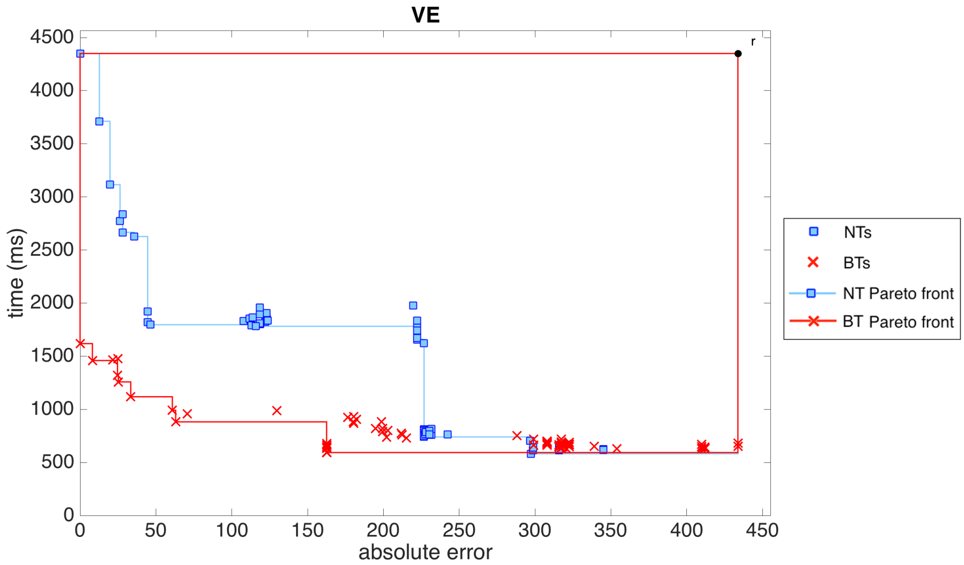


Figure 8.14: Comparison of the absolute error versus the computation time of the NHL ID using the VE algorithm.

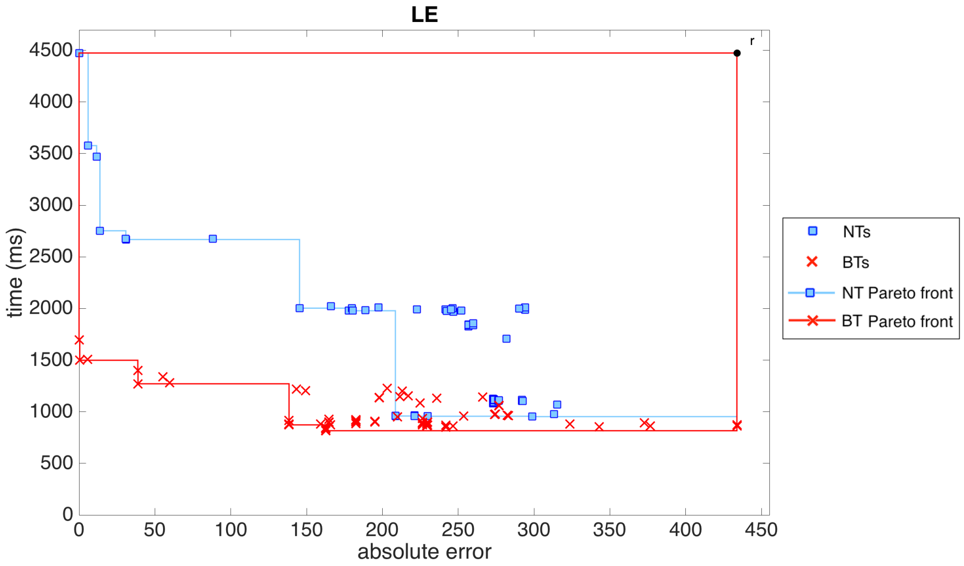


Figure 8.15: Comparison of the absolute error versus the computation time of the NHL ID using the LE algorithm.

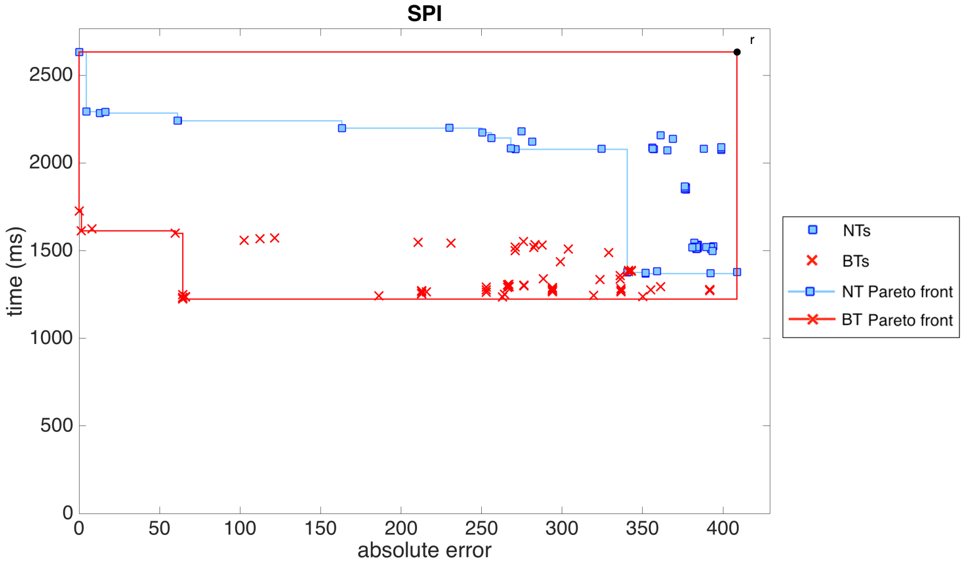


Figure 8.16: Comparison of the absolute error versus the computation time of the NHL ID using the SPI algorithm.

	VE	LE	SPI
H_{NT}	0.68	0.615	0.217
H_{BT}	**0.826**	**0.78**	**0.512**

Table 8.5: Hyper-volume values obtained from points shown in Figures 8.14, 8.15 and 8.16.

8.4.4 Results for the rest of IDs

8.4.4.1 Storage requirements and computation time

Table 8.6 includes the average space savings obtained using NTs and BTs and for different ε values (see Equations (8.9) and (8.10)). When exact evaluation is carried out ($\varepsilon = 0$) tables require less space than trees: for small IDs with a low number of context-specific independencies, the representation with tables is more efficient due to the additional space required for storing the internal nodes in tree representations. Yet, when approximate evaluation is performed ($\varepsilon > 0$) the representation with BTs requires less space. Similar space savings values are obtained for all the evaluation algorithms analysed.

		$\varepsilon = 0.0$	$\varepsilon = 0.05$	$\varepsilon = 0.5$	$\varepsilon = 1.0$
VE	NTs	**−0.418**	0.132	0.384	0.483
	BTs	−0.461	**0.521**	**0.893**	**0.938**
LE	NTs	**−0.348**	0.15	0.368	0.47
	BTs	−0.358	**0.521**	**0.877**	**0.934**
SPI	NTs	**−0.515**	0.019	0.266	0.371
	BTs	−0.581	**0.538**	**0.87**	**0.88**

Table 8.6: Average space saving obtained using trees instead of tables with different ε values.

The improvement in computation time can be analysed using the speedup (Equation (8.11)). Table 8.7 contains the speedup values for tables, NTs and BTs with different values of ε. It can be seen the improvements achieved by BTs: the algorithms VE and SPI are faster when using BTs than with NTs or tables. However, the LE algorithm has a better performance with tables if exact evaluation is carried out. When this method is used with trees, the overhead introduced by pruning and sorting the trees consumes all the benefits for reduced size potentials.

		$\varepsilon = 0.0$	$\varepsilon = 0.05$	$\varepsilon = 0.5$	$\varepsilon = 1.0$
VE	NTs	1.177	1.436	1.679	1.718
	BTs	**1.412**	**2.2**	**2.503**	**2.507**
LE	NTs	0.868	0.947	1.089	1.077
	BTs	**0.909**	**1.117**	**1.286**	**1.406**
SPI	NTs	1.087	1.261	1.385	1.397
	BTs	**1.153**	**1.605**	**1.674**	**1.676**

Table 8.7: Average speedup for IDs using tables and trees (NTs and BTs).

8.4.4.2 Error against time

Table 8.8 contains the results of performing a Wilcoxon signed-rank test with the hyper-volumes for all the random IDs evaluated with the VE, LE and SPI algorithms. The null hypothesis states that there is no difference between the hyper-volumes for BTs and NTs (both share a similar performance). The significance level for rejecting the hypothesis is 5%. For each algorithm the table contains the $p - value$, the percentage of IDs where BTs outperform NTs and finally the conclusion of the test. It can be observed that the null hypothesis is rejected for the methods VE and SPI. As BT outperforms in a higher number of IDs, we can conclude that, for such algorithms, better aproximate solutions are obtained using BTs than NTs.

	p-value	BTs wins	rejected
VE	0.0245	66.67%	*yes*
LE	0.4263	53.33%	*no*
SPI	0.0085	86.67%	*yes*

Table 8.8: Results of the Wilcoxon test for the results using NTs and BTs.

8.5 Conclusions

In this chapter we have explained how BTs are used during the evaluation of IDs using the three different algorithms (VE, LE and SPI). For that, detailed methods for operating with potentials represented as BTs are given.

The experimental work shows that, in general, less memory space is required for storing potentials as a BT than using a NT or a table. As a consequence, the ID evaluation is faster using BTs. However, for some IDs it is necessary to use a threshold higher to 0 for pruning in order to obtain any benefits from the use of BTs. Another conclusion is that using BTs for evaluating IDs offers better approximate solutions than using NTs: the same error level is achieved with a lower computation time. If the three evaluation algorithms are compared, the best improvements achieved by BTs are obtained with the VE and SPI algorithms. By contrast, the required time for building the strong junction tree limits the improvement by the LE algorithm.

In Chapter 9 we propose some heuristics considering that potentials are represented as BTs. Besides, in Chapter 10 of this dissertation, we consider the evaluation of ID with restrictions represented as BTs (i.e., BCTs). This would allow addressing asymmetric decision problems.

As regards future directions of research, we can study alternative ways of applying the pruning operation during the evaluation. In the approach presented here, all the BTs are pruned before the evaluation. Instead, we might consider applying an on-line operation that only prunes the BTs involved in a specific operation if this is extremely costly. In doing so, we might minimize the error of the approximation as the pruning is only done for a few potentials. We could also consider using data structures representing a full model such as RPTs Section 5.2.2 for evaluating IDs.

Chapter 9

Elimination Heuristics with BTs

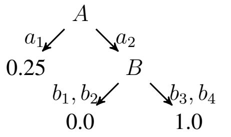
9.1 Introduction

In Chapter 8, BTs have been shown as an effective data structure for representing the potentials in IDs. VE is one of the evaluation algorithms adapted for evaluating IDs when potentials are represented as BTs. This method starts with a set of potentials and it eliminates one variable at each time. As it happens with the corresponding algorithm for Bayesian networks (BNs) [116], the efficiency of VE depends heavily on the optimality of the elimination order. In fact, any method for finding an optimal ordering in a BN, which is an NP-hard problem [68], can be adapted for IDs. Some of the most efficient methods for BNs are greedy algorithms that choose at each step the next variable to remove using a heuristic procedure (see Section 4.5.1.1). These heuristics try to minimize the complexity of the operations involved in the evaluation.

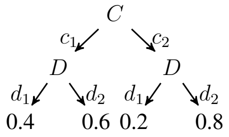
The problem of traditional heuristics is that they assume that potentials are represented using tables, but the complexity of the operations might be different if potentials are represented using pruned BTs. In the present chapter two new heuristics that consider that potentials are represented using BTs are proposed. These heuristics estimate the size of the generated BTs during the removal of a variable.

9.2 Motivation

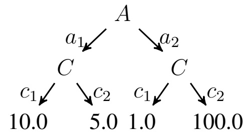
In order to illustrate the need for new elimination heuristics, let us consider the following example. Let $\{A, B, C, D\}$ be a set of variables in an ID and the sizes of their respective domains are: $|\Omega_A| = 2$, $|\Omega_B| = 4$, $|\Omega_C| = 2$, $|\Omega_D| = 2$. Let us suppose that A and C are the two candidate variables to be removed. Assume the ID contains the following potentials:



$\mathcal{BT}^{\phi_1}(B, A)$



$\mathcal{BT}^{\phi_2}(D, C)$



$\mathcal{BT}^{\psi_3}(A, C)$

The generated clique candidate[1] if variable A is eliminated is $C_A = \{A, B, C\}$. By contrast, if C is the removed variable, the generated clique candidate is $C_C = \{A, C, D\}$. The weights of each clique can be calculated as follows.

$$w(C_A) = |\Omega_A| \cdot |\Omega_B| \cdot |\Omega_C| = 2 \cdot 4 \cdot 2 = 16$$

$$w(C_C) = |\Omega_A| \cdot |\Omega_C| \cdot |\Omega_D| = 2 \cdot 2 \cdot 2 = 8$$

If the *minimum weight* heuristic is used, then variable C is selected as the next variable to eliminate. The weights $w(C_A)$ and $w(C_A)$ correspond to the sizes of the tables representing the potentials $\psi_A = \phi_1 \cdot \psi_3$ and $\psi_C = \phi_2 \cdot \psi_3$ respectively. That is, the resulting potentials from combining all the relevant potentials to remove A or C respectively. However, if the potentials are represented using BTs previously pruned, the weight might not correspond with the size of the resulting potential. Moreover, the variable that generates a clique of minimal weight might not be the variable that generates a minimal BT. For example, Figures 9.1 and

[1] As defined in page 76, a clique candidate is the set of all variables contained in the relevant potentials for the removal of a variable.

9.2 shows the resulting BTs of combining the relevant potentials for removing A and C respectively. The BT representing ψ_A contains 9 nodes whereas the BT representing ψ_C contains 15. Therefore, the best option is to choose the variable A. However, the worst option is chosen (removing variable C) when the *minimum weight* heuristic is used.

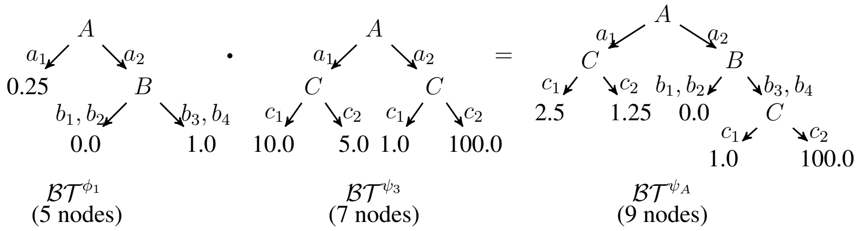


Figure 9.1: Combination of the PPs and UPs if variable A is chosen to be removed

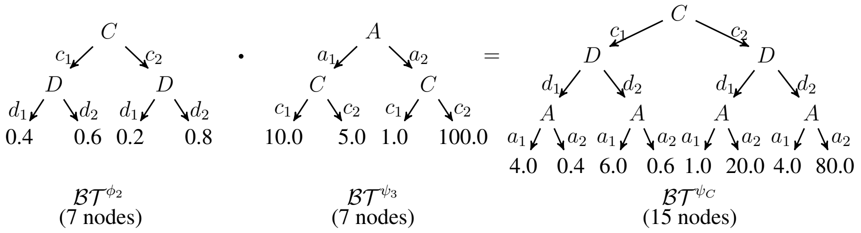


Figure 9.2: Combination of the PPs and UPs performed if variable C is chosen to be removed

9.3 Proposed heuristics

9.3.1 Minimum combined tree

If the VE algorithm is used to evaluate an ID, the highest memory requirements are achieved when combining all the relevant potentials to remove a variable. Here the *minimum combined tree* heuristic is proposed, which is similar to the *minimum weight* since it aims to calculate the size of the resulting potential of the combination. However, this new heuristic considers that potentials are represented using trees.

Let us suppose that we aim to remove a variable Y, the resulting potential of combining all the relevant potentials is $\psi_Y = (((\phi_1)\cdot\phi_2)\cdot\ldots\cdot\phi_n)\cdot(\psi_1+\ldots+\psi_m)$. The generated candidate clique is denoted C_Y. The size of the tree representing ψ_Y can be estimated using Equation (9.1).

$$size(\mathcal{BT}^{\psi_Y}) \simeq size(\mathcal{BT}^{\phi_1}) + leaves(\mathcal{BT}^{\phi_1}) \cdot (2 \cdot \prod_{\substack{X_i \in C_Y \\ X_i \notin dom(\phi_1)}} |\Omega_{X_i}| - 2) \quad (9.1)$$

This heuristic computes the size of $\mathcal{BT}^{\psi_Y}$ from the size of the first potential $\mathcal{BT}^{\phi_1}$ involved in the combination. It checks which of the variables in C_Y are not present in $\mathcal{BT}^{\phi_1}$ and it supposes that a sub-tree with all these variables is added to each leaf of $\mathcal{BT}^{\phi_1}$. With this heuristic, it is selected the variable Y with a minimal $\mathcal{BT}^{\psi_Y}$. For example, let us consider the potentials shown in Section 9.2, then the estimated sizes are:

$$size(\mathcal{BT}^{\psi_A}) \simeq size(\mathcal{BT}^{\psi_1}) + leaves(\mathcal{BT}^{\psi_1}) \cdot (2 \cdot |\Omega_C| - 2) = 5 + 3 \cdot (2 \cdot 2 - 2) = 11$$

$$size(\mathcal{BT}^{\psi_C}) \simeq size(\mathcal{BT}^{\psi_2}) + leaves(\mathcal{BT}^{\psi_2}) \cdot (2 \cdot |\Omega_A| - 2) = 7 + 4 \cdot (2 \cdot 2 - 2) = 15$$

Therefore, using the heuristic *minimum combined tree*, variable A is selected since it generates a minimal tree when combining all the potentials involved. Notice that if trees are completely expanded, the estimated size is the real one. If trees have been previously pruned, it is not possible to compute the real size without combining them, which is not efficient.

9.3.2 Minimum marginalised tree

Here, we propose another heuristic to select a variable to be removed. This heuristic is called *minimum marginalised tree* and it is quite similar to the *Cano and Moral* heuristic (see Section 4.5.1.1). In this case, we aim to choose a variable that generates a minimal tree after the removal. For example, the variable Y that minimises $size(\mathcal{BT}^{\psi_Y})/|\Omega_Y|$ is selected. As it happen with *minimum combined tree*, it computes an approximation of the tree size.

9.4 Experimental work

The aim of the experimental work is to show that the use of specific heuristics for trees reduces the storage size of the potentials during the evaluation. For that purpose two real world IDs are used. First an ID used for the treatment of gastric NHL disease [76] with 3 decisions, 1 utility node and 17 chance nodes. Second, an ID from IctNeo System for jaundice management [97] which contains 2 decisions, 1 utility node and 23 chance nodes. More details about these IDs are given in Appendix B.2. Both IDs are evaluated using BTs for representing the potentials. We will compare the proposed heuristics, namely *minimum combined tree* (MIN_COMB_TREE) and *minimum marginalised tree* (MIN_MARG_TREE), with two of the traditional ones: *minimum weight* (MIN_WEIGHT) and *minimum fill-in arcs weight* (MIN_FILL_WEIGHT). All BTs are pruned before starting evaluation. The threshold ε used for pruning are 0, 0.05 and 0.075.

During the evaluation of both IDs, the storage requirements for representing all potentials are analysed (number of nodes). In particular, the measurements are performed after combining all relevant potentials for removing each variable. It is important to reduce the size of potentials after combinations since at this step largest potentials are generated. Figure 9.3 shows size of all potentials stored in memory during the NHL (left) and Jaundice (right) evaluation using different heuristics and threshold values. The vertical axis indicates the storage size using a logarithmic scale. The horizontal axis indicate the variable removed. It can be observed that when the exact prune is performed (threshold value 0), there is not relevant differences in the storage requirements. In fact, at some stages of the evaluation, MIN_WEIGHT and MIN_FILL_WEIGHT require less storage size. However, using a threshold for pruning greater than 0, specific heuristics for BTs (MIN_COMB_TREE and MIN_MARG_TREE) offer better results. When a high threshold value is used, the size of a BT is much smaller than the size of a table representing the same potential. Since MIN_WEIGHT and MIN_FILL_WEIGHT assume that potentials are represented with tables, it is not able to take advantage of such information.

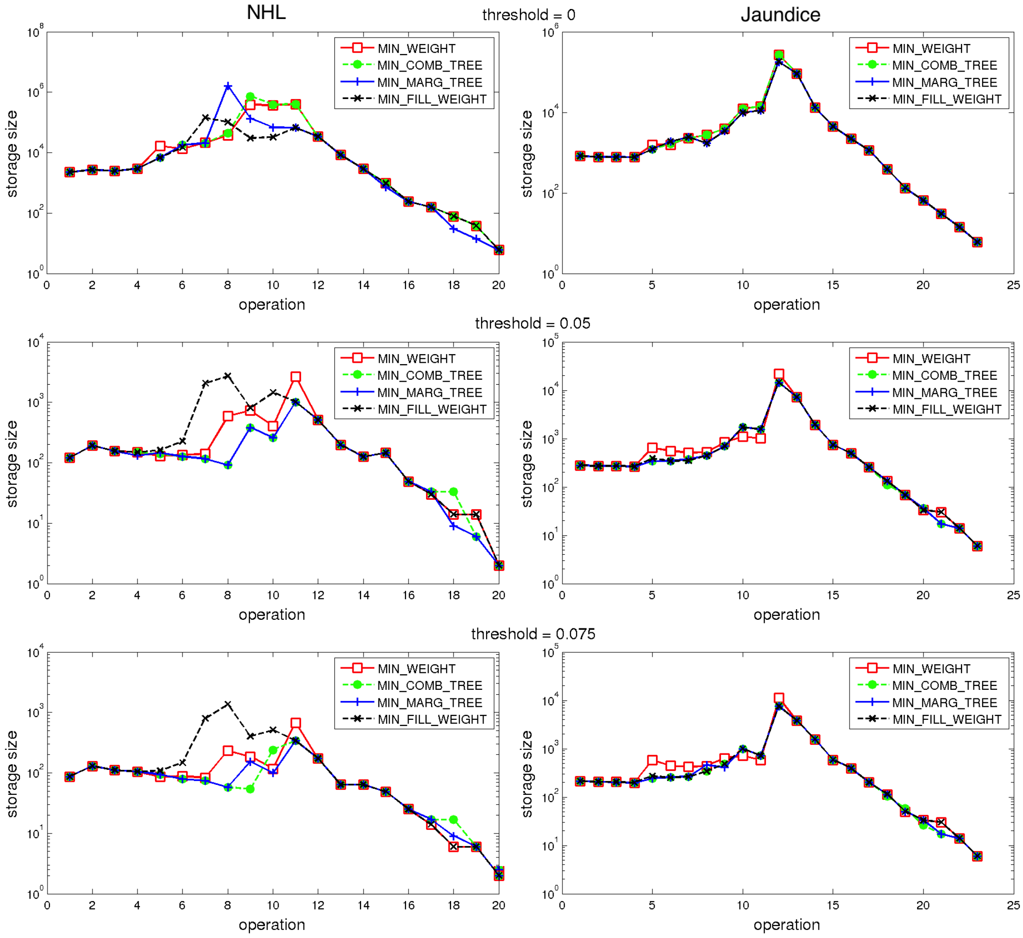


Figure 9.3: Size of all potentials stored in memory during the NHL ID (left) and IctNeo ID (right) evaluation comparing the heuristics *minimum weight*, *minumum combined tree*, *minumum marginalised* and *minimum fill-in arcs weight* using BTs and different threshold values.

The storage requirements can be analysed in more detail with Tables 9.1 and 9.2, which show the mean and maximum storage requirements in number of nodes for evaluating NHL and Jaundice IDs respectively. Specific heuristics (MIN_COMB_TREE and MIN_MARG_TREE) require less storage size if an important prune is performed.

	Mean storage size			Maximum storage size		
	$\varepsilon = 0$	$\varepsilon = 0.05$	$\varepsilon = 0.075$	$\varepsilon = 0$	$\varepsilon = 0.05$	$\varepsilon = 0.075$
MIN_WEIGHT	$6.51 \cdot 10^4$	325	115	$4.03 \cdot 10^5$	2670	676
MIN_COMB_TREE	$8.33 \cdot 10^4$	192	89.2	$7.33 \cdot 10^5$	**1010**	**340**
MIN_MARG_TREE	$9.92 \cdot 10^4$	**190**	**86.8**	$1.61 \cdot 10^6$	**1010**	**340**
MIN_FILL_WEIGHT	$\mathbf{2.25 \cdot 10^4}$	508.15	225.15	$\mathbf{1.41 \cdot 10^5}$	2730	1360

Table 9.1: Mean and maximum storage requirements in number of nodes for the evaluation of the NHL ID

	Mean storage size			Maximum storage size		
	$\varepsilon = 0$	$\varepsilon = 0.05$	$\varepsilon = 0.075$	$\varepsilon = 0$	$\varepsilon = 0.05$	$\varepsilon = 0.075$
MIN_WEIGHT	$1.84 \cdot 10^4$	1690	984	$2.7 \cdot 10^5$	$2.16 \cdot 10^4$	$1.13 \cdot 10^4$
MIN_COMB_TREE	$1.84 \cdot 10^4$	**1390**	**798**	$2.7 \cdot 10^5$	$\mathbf{1.44 \cdot 10^4}$	**7510**
MIN_MARG_TREE	$\mathbf{1.42 \cdot 10^4}$	**1390**	800	$\mathbf{1.8 \cdot 10^5}$	$\mathbf{1.44 \cdot 10^4}$	**7510**
MIN_FILL_WEIGHT	$\mathbf{1.42 \cdot 10^4}$	**1390**	800	$\mathbf{1.8 \cdot 10^5}$	$\mathbf{1.44 \cdot 10^4}$	**7510**

Table 9.2: Mean and maximum storage requirements in number of nodes for the evaluation of the Jaundice ID

9.5 Conclusions

Finding an elimination ordering that minimises the size of potentials during the evaluation is an element of crucial importance for the efficiency of the VE algorithm. Some greedy algorithms use deterministic heuristics for choosing the next variable to remove. However, these heuristics are not appropriate if potentials are represented using BTs. In this chapter, two new heuristics that estimate the sizes of intermediate potentials during the evaluation have been proposed. In the experimentation, it has been shown that the use of these heuristics reduce the storage requirements. In particular, these heuristics offer the best results if high thresholds for pruning the BTs are used.

Related to the optimization of the evaluation process, in Chapter 12 we will study different alternatives for optimizing the evaluation assuming that the potentials of the ID are represented as tables. With the heuristics proposed in this chapter, it is only possible to optimize the order of the marginalizations involved in the evaluation. By contrast, in Chapter 12 we will explore different alternatives for optimizing al the operations with potentials (e.g., combination, sum-marginalization, max-marginalization, etc.) Finding an optimal ordering will reduce the complexity of operations and therefore the efficiency of the evaluation. Some methods such as *symbolic probabilistic inference* (SPI) [104, 74] reorder the combination and marginalization operations to reduce the complexity of computations. Thus, the SPI algorithm will be adapted for evaluating IDs. Additionally, an optimization of VE will be proposed.

Chapter 10

Evaluation of Asymmetric Decision Problems with BTs

10.1 Introduction

As explained in Chapter 6, the evaluation of IDs representing asymmetric decision problem implies that a considerable amount of unnecessary memory space and computation may be involved. For that reason, we proposed representing potentials and asymmetries using BTs. Asymmetries represented as BTs are called binary constraint trees (BCTs).

Here we will explain how to evaluate ID where potentials and asymmetries are represented with BTs. In our approach we try to keep separately qualitative (graph and constraints due to asymmetries) and quantitative (potentials) knowledge, merely because a constraint may affect several potentials, with some of them not being present in the model (i.e. distributions managed during the evaluation process and derived from the initial ones). Potentials are represented using BTs instead of tables. On the other hand, asymmetries are initially represented as constraint rules. However, during the evaluation, these constraint rules are transformed into BCTs making their application possible.

NTs were already used for this purpose in [54], however the use of BTs will improve the efficiency even in those cases where the use of NTs is counterproductive (the application of the asymmetries has an overhead). Our approach is evaluated against the corresponding one with NTs in terms of running time and storage requirements.

10.2 Applying constraints to potentials

10.2.1 Applicability of a constraint rule

In order to take advantage of asymmetries and avoid unnecessary computations, the information about impossible scenarios present in constraint rules should be combined with the information represented in potentials. We say that a constraint rule is applicable to a potential if it meets the conditions given in Proposition 8.

Proposition 8 (applicability of a constraint rule) *Let ψ be a potential (no matter if PP or UP) defined over $\mathbf{X}_I$ and κ a constraint rule for the variables in $\mathbf{X}_J$. The constraint rule κ is applicable to ψ if and only if both the antecedent and the consequent are applicable. To decide if each of them are applicable we will consider the following cases:*

- ***Case A**: An atomic sentence in κ for X_j is applicable if $X_j \in \mathbf{X}_J \cap \mathbf{X}_I$.*
- ***Case B**: The negation of a logical expression is applicable if and only if the logical expression itself is applicable.*
- ***Case C**: A conjunction is applicable if and only if the two conjuncts are applicable.*
- ***Case D**: A disjunction is applicable if and only if at least one of the disjuncts is applicable.*

The applicability is verified by starting with the simplest well-formed formulas in a logical expression, i.e. a atomic sentence. For composed logical expressions, the applicability can be checked if it has been verified for their composing formulas. An example is given below with one of the constraint rules in the reactor problem.

Example 36 (applicability of a constraint rule) *Let us consider the potential $\phi(T|D_1, A)$ and the constraint rule $\kappa_4 : (T = b) \vee (D_2 \in \{c, n\}) \Rightarrow A \in \{\}$ in Examples 26 and 28 respectively. The set of common set of variables is $\{T, A\}$. Considering Proposition 8, we will reason as follows.*

- *The atomic sentence in $T = b$ is applicable because T belongs to the set of common variables (case A).*
- *The atomic sentence $D_2 \in \{c, n\}$ is not applicable because does not belongs to the set of common variables (case A).*
- *The antecedent is applicable since one of the disjuncts, i.e., $(T = b)$, is applicable (case D).*
- *The consequent is applicable because it is an atomic sentence for the variable A, which is in the set of common variables (case A).*

As both the antecedent and consequent are applicable, κ_4 is applicable to $\phi(T|D_1, A)$.

From Proposition 8 it can be deduced that, if a constraint rule is applicable to a potential, then they have at least a variable in common. This is a necessary but not sufficient condition for the applicability of a constraint rule. For example, the constraint rule $(T = b) \wedge (D_2 \in \{c, n\}) \Rightarrow A \in \{\}$ is not applicable to $\phi(T|D_1, A)$ even though the set of common variables is $\{T, A\}$.

10.2.2 Applying BCTs

Even though defining a set of constraint rules is an intuitive way to identify the impossible configurations, they must be transformed into BTs for their application to potentials (represented as BTs) in an ID. A BT representing a constraint rule is called a *binary constraint tree* and was already defined in Section 6.3.2.

The process for applying $\kappa(\mathbf{X}_J)$ to a binary tree $\mathcal{BT}^{\psi}(\mathbf{X}_I)$ representing a potential (no matter if PP or UP) is the following:

1. The applicability must be checked as explained in previous section.
2. A binary tree $\mathcal{BT}^{\kappa}(\mathbf{X}_J)$ from κ is built[1].
3. Non-common variables are removed from $\mathcal{BT}^{\kappa}(\mathbf{X}_J)$ using max-marginalization, and the resulting tree is multiplied by the BT representing the potential

The result of this process is a new BT, denoted $\mathcal{BT}^{\kappa\cdot\psi}(\mathbf{X}_I)$, that represent ψ but it takes the value 0.0 for those configurations that are not allowed due to the asymmetry represented by κ. This process is summarized in Equation (10.1).

$$\mathcal{BT}^{\kappa\cdot\psi}(\mathbf{X}_I) = \left(\max_{\mathbf{X}_J \setminus \mathbf{X}_I} \mathcal{BT}^{\kappa}(\mathbf{X}_J)\right) \cdot \mathcal{BT}^{\psi}(\mathbf{X}_I) \tag{10.1}$$

When evaluating an ID, we will be interested in applying a set of constraints rather than a single one. Algorithm 16 details the process for a applying a set of constraints rules to a binary tree $\mathcal{BT}^{\psi}(\mathbf{X}_I)$. For each rule $\kappa \in \mathcal{K}$, the applicability is checked . If applicable, a BCT is built[2] and multiplicated with the all applicable constraints (lines 2 to 11). Note that non-common variables are max-marginalized out. Finally, the BCT representing all the constraints is combined with the target BT (lines 12 to 16). The output is a BT representing the same potential but taking a 0.0 for the impossible configurations.

[1] The process for building a BCT from a constraint rule was explained in Section 6.3.2

[2] The building process can be avoided if the BCT was previously built during the evaluation.

Algorithm 16 ApplyConstraints

```
input : $\mathcal{K}$ (set of constraint rules), $\mathcal{BT}^{\psi}(\mathbf{X}_I)$ (target BT representing a PP or UP)
output : $\mathcal{BT}^{\mathcal{K}'\cdot\psi}(\mathbf{X}_I)$ (resulting BT of applying constraints)
$\mathcal{BT}^{\mathcal{K}'} \leftarrow null$
for all $\kappa(\mathbf{X}_J) \in \mathcal{K}$ do
    if $\kappa(\mathbf{X}_J)$ is applicable to $\mathcal{BT}^{\psi}(\mathbf{X}_I)$ then
        Build a sorted $\mathcal{BT}^{\kappa}(\mathbf{X}_J)$
        if $\mathcal{BT}^{\mathcal{K}'} = null$ then
            $\mathcal{BT}^{\mathcal{K}'} \leftarrow \max_{\mathbf{X}_J \setminus \mathbf{X}_I} \mathcal{BT}^{\kappa}$
        else
            $\mathcal{BT}^{\mathcal{K}'} \leftarrow \mathcal{BT}^{\mathcal{K}'} \cdot \max_{\mathbf{X}_J \setminus \mathbf{X}_I} \mathcal{BT}^{\kappa}$
        end if
    end if
end for
if $\mathcal{BT}^{\mathcal{K}'} \neq null$ then
    $\mathcal{BT}^{\mathcal{K}'\cdot\psi} \leftarrow \mathcal{BT}^{\mathcal{K}'} \cdot \mathcal{BT}^{\psi}$
else
    $\mathcal{BT}^{\mathcal{K}'\cdot\psi} \leftarrow \mathcal{BT}^{\psi}$
end if
return $\mathcal{BT}^{\mathcal{K}'\cdot\psi}(\mathbf{X}_I)$
```

By simply applying a BCT to a potential, a reduction in the size cannot be assured. For that reason, we will apply the exact pruning operation[3] (i.e. with threshold $\varepsilon = 0.0$) in order to group, as much as possible, all the new leaves labelled with 0.0. Yet, in Section 10.2.3, improved versions of the algorithms for performing the operations over BTs are given. With them, the pruning of the resulting BT is not needed.

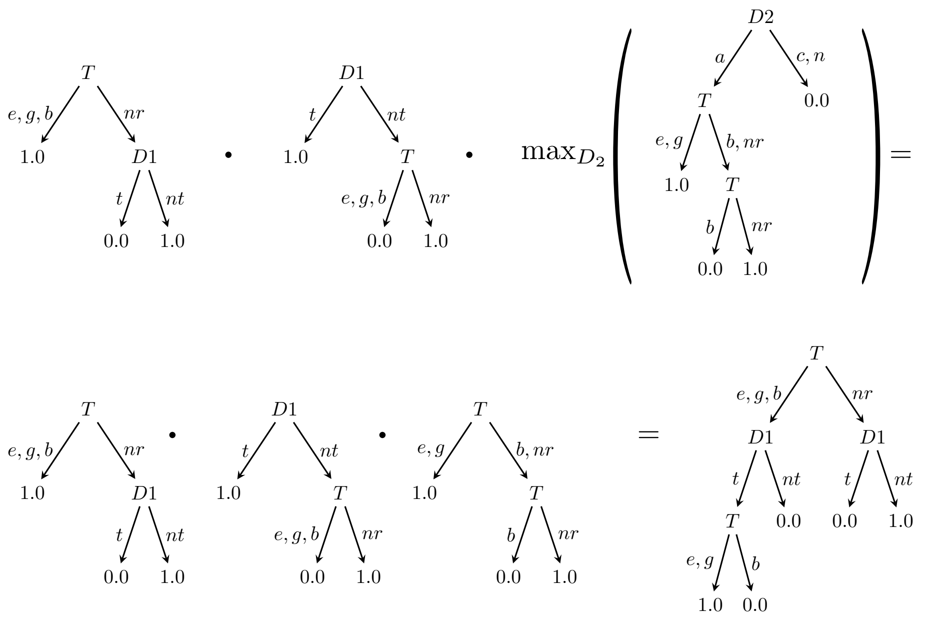
Example 37 (application of a set of constraint rules to a potential) *Let us consider that $\mathcal{BT}^{\phi}(T, D_1, A)$ represents the PP $\phi(T|D_1, A)$ given in Example 26 and the set of constraint rules in Example 28. Then, the process for applying this set of constraints is the following.*

[3] For more details about the pruning operation in BTs, see Section 5.4.2.

1. *The constraint rules that are applicable to $\phi(T|D_1, A)$ are $\kappa_1(D_1, T)$, $\kappa_2(D_1, T)$ and $\kappa_4(T, D_2, A)$. The BCTs representing these constraints rules $\mathcal{BT}^{\kappa_1}(D_1, T)$, $\mathcal{BT}^{\kappa_2}(D_1, T)$ and $\mathcal{BT}^{\kappa_4}(T, D_2, A)$ are built. For $\mathcal{BT}^{\kappa_4}(T, D_2, A)$ the max-marginalization is performed since it includes the variable D_2 which is not present in $\phi(T|D_1, A)$. Then the multiplication of all the relevant BCTs is done:*

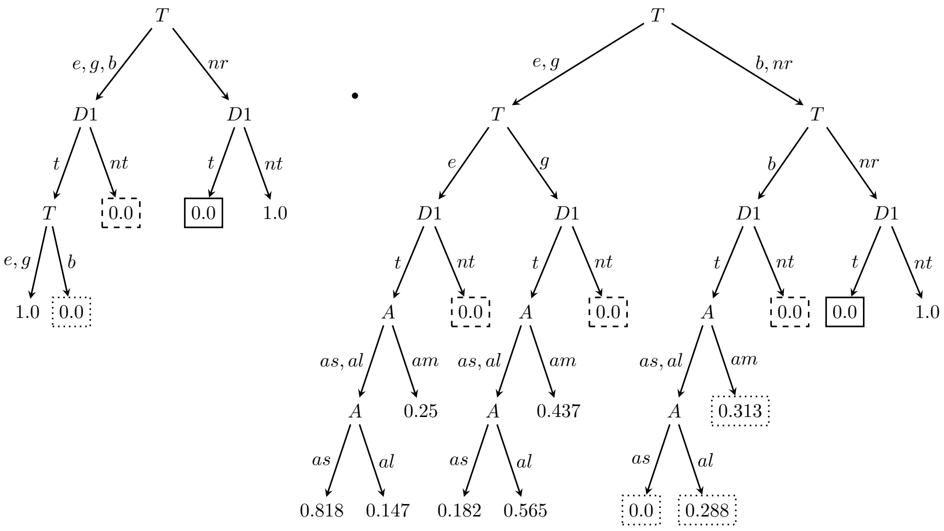
$\kappa_1 : D_1 = t \Rightarrow T \in \{e, g, b\}$ $\kappa_2 : D_1 = nt \Rightarrow T \in \{nr\}$ $\kappa_4 : (T = b) \vee (D_2 \in \{c, n\}) \Rightarrow A \in \{\}$

$\mathcal{BT}^{\kappa_1}(D_1, T)$ $\mathcal{BT}^{\kappa_2}(D_1, T)$ $\mathcal{BT}^{\kappa_4}(T, D_2, A)$



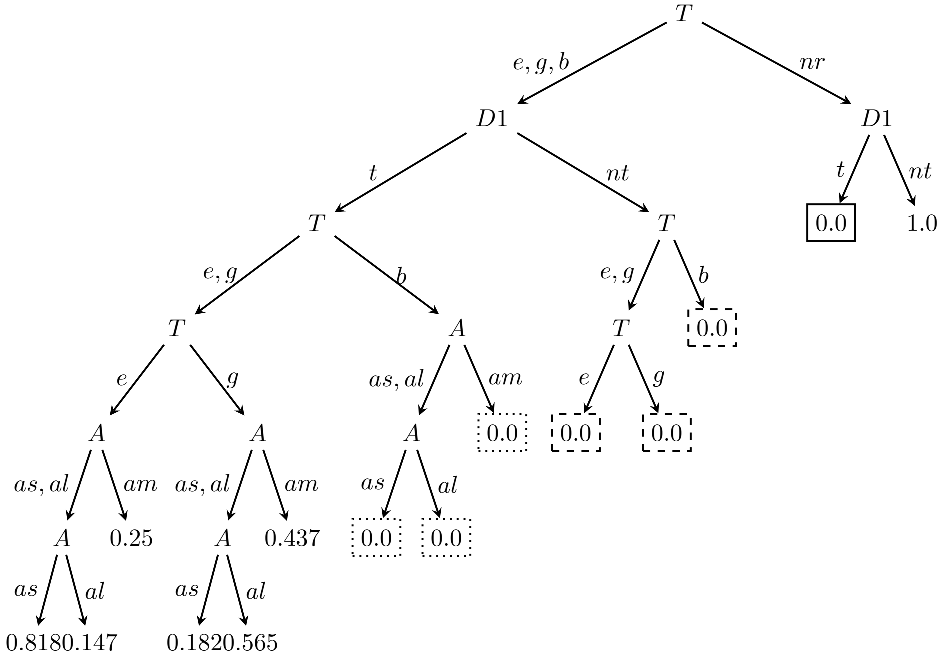
2. *We obtain a BCT representing all the constraint rules, denoted $\mathcal{BT}^{\kappa_1 \cdot \kappa_2 \cdot \kappa_4}$. Such BCT is combined (i.e. multiplied) with the PP:*

$\mathcal{BT}^{\kappa_1 \cdot \kappa_2 \cdot \kappa_4}(D_1, T, A)$ • $\mathcal{BT}^{\phi}(T, D_1, A)$



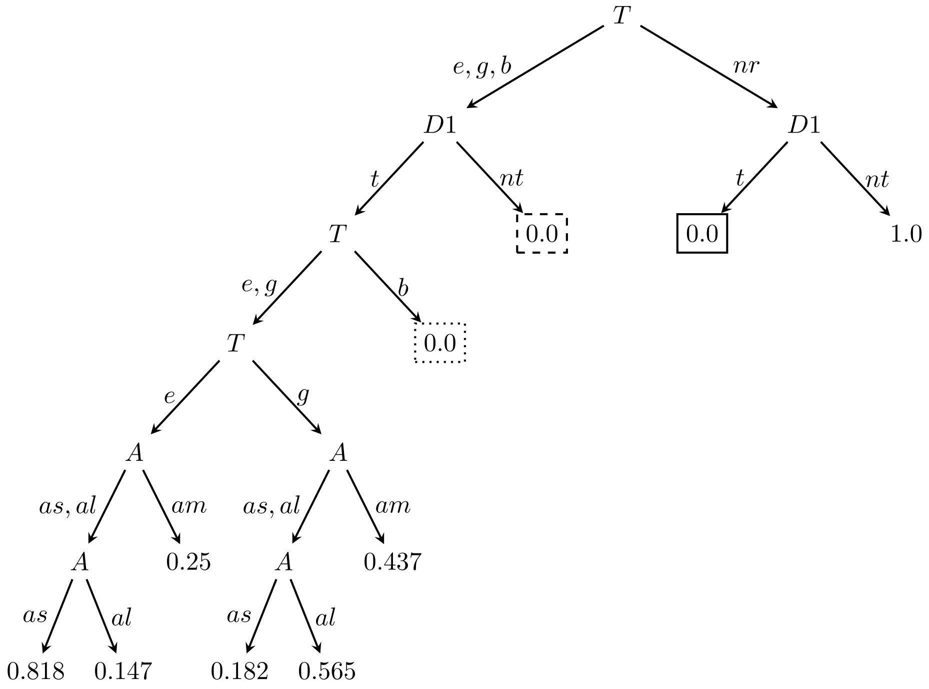
3. *The result of combining the BCT with the PP is a new BT denoted* $\mathcal{BT}^{\kappa_1 \cdot \kappa_2 \cdot \kappa_4 \cdot \phi}$*. This tree represents the same PP but taking* 0.0 *for the configurations leading to impossible scenarios:*

$\mathcal{BT}^{\kappa_1 \cdot \kappa_2 \cdot \kappa_4 \cdot \phi}(T, D_1, A)$



4. *By applying the BCT to the PP, the size of the BT remains the same: it contains 27 nodes. We will apply the exact pruning operation (i.e. with threshold* $\varepsilon = 0.0$*) in order to group all the leaves labelled with* 0.0*. The final result is the following.*

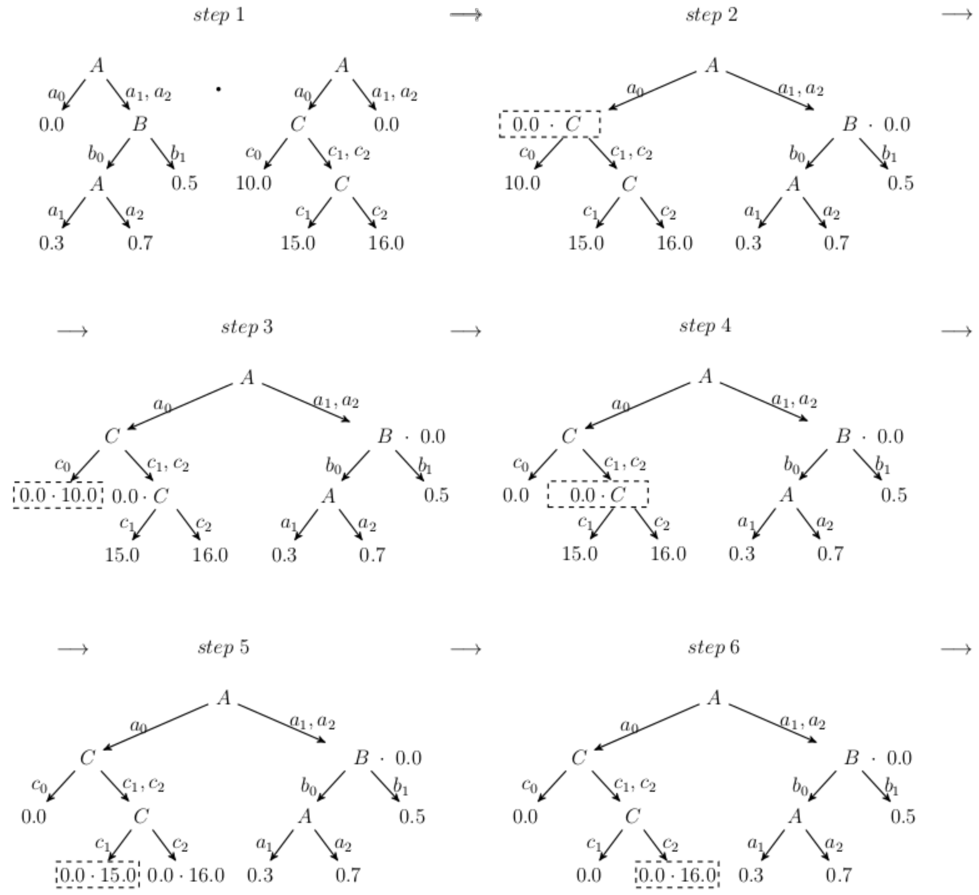
$$prune\left(\mathcal{BT}^{\kappa_1 \cdot \kappa_2 \cdot \kappa_4 \cdot \phi}(T, D_1, A),\ \varepsilon = 0.0\right)$$

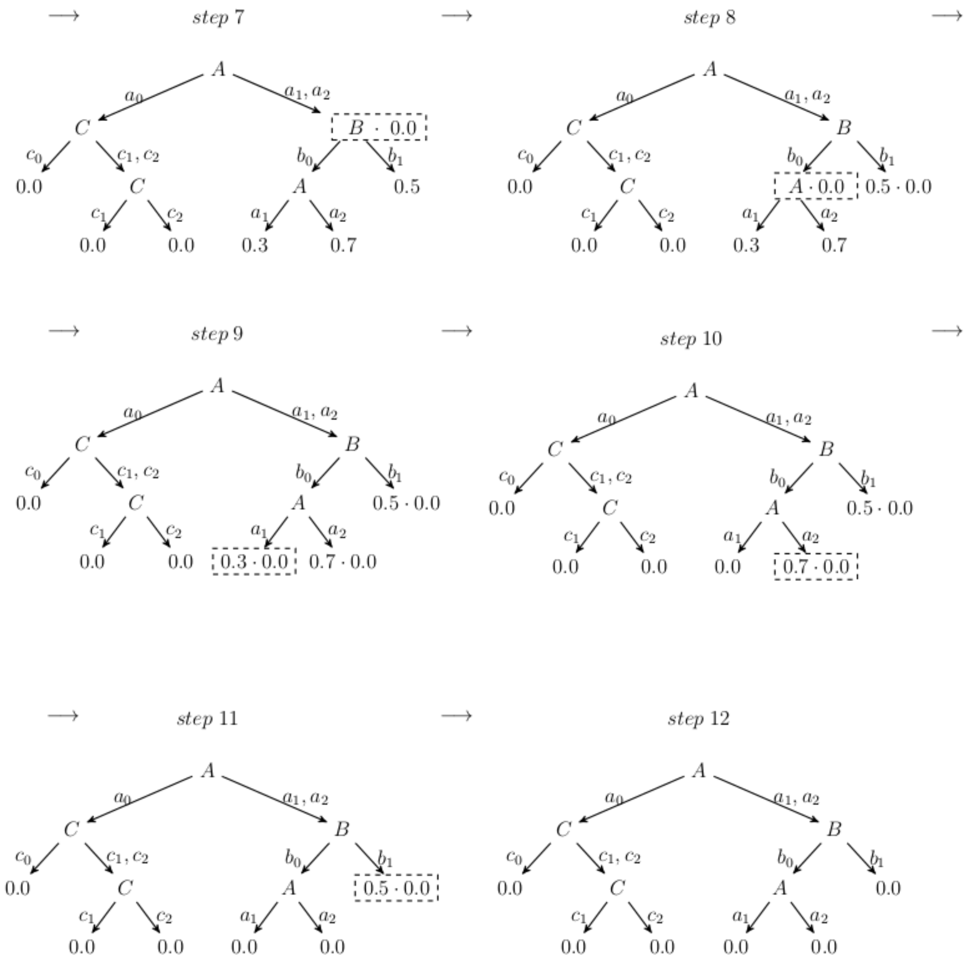


10.2.3 Improved operations

The potentials of IDs representing asymmetric decision problems contain 0.0 values for many configurations. In fact, the application of a constraint rule to potentials consist of replacing some values by 0.0. Yet, the algorithms given in Section 8.2 for operating with BTs do take advantage of that situation. In order to illustrate this problem, let us consider the multiplication shown in the example below.

Example 38 (multiplication of two BTs containing 0.0 values) *Let us consider two binary trees $\mathcal{BT}_1$ and $\mathcal{BT}_2$ representing the potentials $\phi(A|B)$ and $\psi(A, C)$ respectively. If we apply Algorithm 10 (page 150), the process for performing the multiplication $\mathcal{BT}_1 \cdot \mathcal{BT}_2$ is detailed below. Those nodes being processed by the algorithm at each step are highlighted with a dashed pattern.*





Notice that in step 2, the algorithm aims to multiply a leaf node labelled with 0.0 and a BT with variable the C in the root. A similar situation happens in step 7, where the second operand is a leaf node labelled with 0.0. At these steps, we already now that the result will be a BT with all their leaves labelled with 0.0. Thus, there is no need to traverse the BT: the current node can be simply labelled with 0.0 as shown in Figure 10.1.

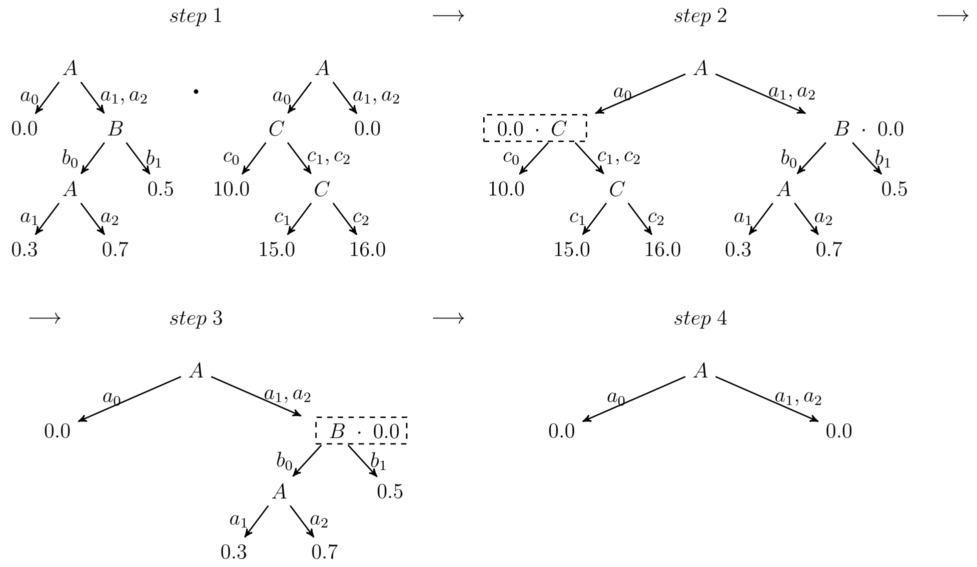


Figure 10.1: Process for multiplying the same BTs than in Example 38 but without the unnecessary computations.

Now we will explain how operations on potentials can be modified in order to avoid the unnecessary computations previously described. The improvement proposed is based on the idea that, when multiplying or dividing two BTs, if any of the operands is a leaf node labelled with the value 0.0, the result is also a leaf node labelled with 0.0. Note that, in any division (with real numbers) performed with the VE algorithm, if the denominator is equal to 0.0, so is the numerator. In that case, the convention $\frac{0}{0} = 0$ is adapted.

In Section 8.2, the algorithms for performing the multiplication and division were described by means of the definition of the generic combinator operator $\otimes$, which also allows describing other operations such as the addition or the maximum. Algorithm 17 shows the improved version of this operation. The new additional code corresponds to lines 3 to 9: the boolean variable *stop*, which is used to control if the combination process can be stopped, is initialized to *false*. If any of the BTs is a leaf node labelled with 0.0, then the label of the new node is set to 0.0 and *stop* is set to $true$ (lines 5 to 8). The algorithm will only perform more recursive calls if *stop* is equal to *false*. This improvement is only included for two types of combinations: *multiplication* and *division*.

Algorithm 17 Improved Generic Combination

input : t_1 and t_2 (root nodes of $\mathcal{BT}_1$ and $\mathcal{BT}_2$)

output : the root of $\mathcal{BT} = \mathcal{BT}_1 \otimes \mathcal{BT}_2$

```
Build a new node $t_n$
$stop \leftarrow false$
/*Check if the process can be stopped*/
if multiplication or division then
    if ($t_1$ is a leaf node and $L_{t_1} = 0$)
or ($t_2$ is a leaf node and $L_{t_1} = 0$) then
        $L_{t_n} \leftarrow 0.0$                                    ▷ Sets the label of $t_n$
        $stop \leftarrow true$
    end if
end if
if $stop = false$ then                                  ▷ The process cannot be stopped
    if $t_1$ is a leaf node then
        if $t_2$ is a leaf node then
            $L_{t_n} \leftarrow f(L_{t_1}, L_{t_2})$                 ▷ Sets the label of the leaf
        else
            $L_{t_n} \leftarrow L_{t_2}$                             ▷ Sets the label of $t_n$
            $L_{lb(t_n)} \leftarrow L_{lb(t_2)}$                     ▷ Sets labels for both branches
            $L_{rb(t_n)} \leftarrow L_{rb(t_2)}$
            $t_{nl} \leftarrow$ combination($t_1, t_{2_l}$)          ▷ Sets children
            $t_{nr} \leftarrow$ combination($t_1, t_{2_r}$)
        end if
    else
        Let $X_i$ be the variable labelling $t_1$
        $L_{t_n} \leftarrow L_{t_1}$                                 ▷ Sets the label of $t_n$
        $L_{lb(t_n)} \leftarrow L_{lb(t_1)}$                         ▷ Sets the labels of both branches
        $L_{rb(t_n)} \leftarrow L_{rb(t1)}$
        $t_{nl} \leftarrow$ combination($t_{1_l}, \mathcal{BT}_2^{R(X_i, L_{lb(t_1)})}$)   ▷ Sets children
        $t_{nr} \leftarrow$combination($t_{1_r}, \mathcal{BT}_2^{R(X_i, L_{rb(t_1)})}$)
    end if
end if
return $t_n$
```

As a reminder of Section 8.2, in line 13 the leaf node of the new tree is labelled with the result of a function f, which will depend on the particular type of combination. For example, in case of multiplying two BTs, this function is defined as $f(L_{t_1}, L_{t_2}) = L_{t_1} \cdot L_{t_2}$. On the other hand, for the division, it is defined as $f(L_{t_1}, L_{t_2}) = \frac{L_{t_1}}{L_{t_2}}$.

A noticeable aspect of using this implementation of the combination operation is that, the size of the resulting BTs can be reduced without pruning them before the application of constraints.

10.3 ID Evaluation with BCTs

This section shows how BCTs can be applied to evaluate IDs. In particular, we work with the VE algorithm, which was explained in its general version in Section 4.5.1. This method for working with BTs was given in Section 8.3. Algorithm 18 adapts it for working with BTs representing potentials and constraints (i.e., BCTs).

The algorithm includes an additional input parameter, which is the set of constraint rules in the ID, denoted as $\mathcal{K}$. If we compare this procedure with the one without BCTs (Algorithm 12, page 161), we observe that such algorithm differs in the initialization phase (lines 1 to 5). After building each BT representing an initial potential, the constraints are applied by invoking *applyConstraints* (previously explained in Section 10.2.2). Note that for their application, constraint rules will be transformed into BCTs. If the improved versions of the operations explained in previous section are used, this application already reduces the size of potentials. However, a greatest reduction can be achieved if the nodes in the BT are sorted and pruned[4].

[4] The methods for sorting and pruning were explained in Section 5.4

Algorithm 18 Variable Elimination with BCTs

```
input : $\Phi, \Psi$ (sets of potentials in the ID), $\{\mathcal{I}_0, D_1, \mathcal{I}_1, \ldots, D_n, \mathcal{I}_n\}$ (partitions of nodes in the ID), $\varepsilon$ (pruning threshold), $\mathcal{K}$ (set of constraint rules).
/*Initialization phase*/
for all $\phi \in \Phi$ do
    Build a binary tree $\mathcal{BT}^{\phi}$ representing $\phi$
    $\mathcal{BT}^{\mathcal{K}' \cdot \phi} \leftarrow \text{applyConstraints}(\mathcal{K}, \mathcal{BT}^{\phi})$    ▷ Algorithm 16
    $\Phi \leftarrow \Phi \backslash \{\phi\} \cup \{prune(sort(\mathcal{BT}^{\mathcal{K}' \cdot \phi}),\ \varepsilon)\}$
end for
for all $\psi \in \Psi$ do
    Build a binary tree $\mathcal{BT}^{\psi}$ representing $\psi$
    $\mathcal{BT}^{\mathcal{K}' \cdot \psi} \leftarrow \text{applyConstraints}(\mathcal{K}, \mathcal{BT}^{\psi})$    ▷ Algorithm 16
    $\Psi \leftarrow \Psi \backslash \{\psi\} \cup \{prune(sort(\mathcal{BT}^{\mathcal{K}' \cdot \psi}),\ \varepsilon)\}$
end for
/*Removal Loop*/
for $k \leftarrow n$ to $0$ do
    while $\mathcal{I}_k \neq \emptyset$ do
        Select $X \in \mathcal{I}_k$    ▷ Pick a chance variable to eliminate
        $(\Phi, \Psi) \leftarrow \text{ElimVarBCT}(X, \Phi, \Psi, \mathcal{K})$    ▷ Chance variable elimination (Algorithm 19)
        $\mathcal{I}_k \leftarrow \mathcal{I}_k \backslash \{X\}$
    end while
    if $k > 0$ then
        $(\Phi, \Psi) \leftarrow \text{ElimVarBCT}(D_k, \Phi, \Psi, \mathcal{K})$    ▷ Decision variable elimination (Algorithm 19)
    end if
end for
```

During the removal loop in the previous algorithm, intermediate potentials might contain values different to 0 but corresponding with impossible configurations. It could happen that some constraint rules are not applicable to any of the initial potentials but they are applicable to some of the generated potentials during the evaluation. For that reason, the constraint application is also required after obtaining such intermediate potentials, i.e., when invoking *ElimVarBCT*, which is shown in Algorithm 19.

Algorithm 19 ElimVarBCT - Elimination of a single variable

```
input :  $Y$ (variable to remove), $\Phi$, $\Psi$ (sets of current potentials), $\mathcal{K}$ (set of constraints rules)
output :  $\Phi$, $\Psi$ (updated sets of current potentials without $Y$)
 1: $\Phi_Y \leftarrow \{\mathcal{BT}^{\phi} \in \Phi | Y \in dom(\mathcal{BT}^{\phi})\}$                                ▷ Select
 2: $\Psi_Y \leftarrow \{\mathcal{BT}^{\psi} \in \Psi | Y \in dom(\mathcal{BT}^{\psi})\}$
 3: $\mathcal{BT}^{\phi_Y} \leftarrow \prod_{\mathcal{BT}^{\phi} \in \Phi_Y} \mathcal{BT}^{\phi}$                                ▷ Combine
 4: $\mathcal{BT}^{\psi_Y} \leftarrow \sum_{\mathcal{BT}^{\psi} \in \Psi_Y} \mathcal{BT}^{\psi}$
 5: $\mathcal{BT}^{\mathcal{K}'\cdot\phi_Y} \leftarrow$ applyConstraints$(\mathcal{K}, \mathcal{BT}^{\phi_Y})$
 6: $\mathcal{BT}^{\mathcal{K}'\cdot\psi_Y} \leftarrow$ applyConstraints$(\mathcal{K}, \mathcal{BT}^{\psi_Y})$
 7: if $Y \in \mathcal{U}_C$ then
 8:     $(\mathcal{BT}^{\phi'_Y}, \mathcal{BT}^{\psi'_Y}) \leftarrow (\sum_Y \mathcal{BT}^{\mathcal{K}'\cdot\phi_Y}, \frac{\sum_Y \mathcal{BT}^{\mathcal{K}'\cdot\phi_Y} \cdot \mathcal{BT}^{\mathcal{K}'\cdot\psi_Y}}{\sum_Y \mathcal{BT}^{\mathcal{K}'\cdot\phi_Y}})$     ▷ Remove by sum
 9: else
10:     $(\mathcal{BT}^{\phi'_Y}, \mathcal{BT}^{\psi'_Y}) \leftarrow ((\mathcal{BT}^{\mathcal{K}'\cdot\phi_Y})^{R(Y=y)}, \max_Y \mathcal{BT}^{\mathcal{K}'\cdot\psi_Y})$     ▷ Remove by max
11:     $\widehat{\delta}_Y \leftarrow \arg\max_Y \mathcal{BT}^{\mathcal{K}'\cdot\psi_Y}$                          ▷ Optimal policy
12: end if
13: $\mathcal{BT}^{\mathcal{K}'\cdot\phi'_Y} \leftarrow$ applyConstraints$(\mathcal{K}, \mathcal{BT}^{\phi'_Y})$
14: $\mathcal{BT}^{\mathcal{K}'\cdot\psi'_Y} \leftarrow$ applyConstraints$(\mathcal{K}, \mathcal{BT}^{\psi'_Y})$
15: $(\Phi, \Psi) \leftarrow (\Phi \backslash \Phi_Y \cup \{\mathcal{BT}^{\mathcal{K}'\cdot\phi'_Y}\}, \Psi \backslash \Psi_Y \cup \{\mathcal{BT}^{\mathcal{K}'\cdot\psi'_Y}\})$     ▷ Update
16: return $(\Phi, \Psi)$
```

In the previous algorithm, the constraint rules are applied after combining all the potentials (lines 5 and 6) and after the marginalization of the variable Y (lines 13 and 14). Note that, in this case, BTs are not sorted and pruned after the application of the constraints: these operations are time-consuming, therefore sorting and pruning the BTs at each iteration is counter-productive. Yet, as the improved versions of the combination algorithms explained in previous section are used, the application of constraints can reduce the size of potentials without the need of sorting and pruning the BTs.

10.4 Experimental work

For an empirical validation of the VE algorithm for IDs proposed in this chapter, we consider a benchmark of six IDs modelling real decision tasks. Table 10.1 details the number of nodes of each type for these models.

Name of the ID	$\vert\mathcal{U}_C\vert$	$\vert\mathcal{U}_D\vert$	$\vert\mathcal{U}_V\vert$
Car Buyer	3	3	1
Dating	12	4	7
Diabetes	4	3	3
Reactor	3	2	3
NHL	17	3	1
Maze	14	2	1

Table 10.1: Number of chance, decision and utility nodes for the benchmark IDs

Some of these IDs have often been used in the literature for illustrating the problematic related to the evaluation of asymmetric decision problems: the car buyer ID [94]; an ID representing the dating decision problem [71]; the reactor ID, which was detailed in Chapter 6 for introducing the asymmetries in decision problems. Some other IDs that are not usually related to the literature about asymmetries but being highly asymmetric are: the NHL ID, which is a large real world IDs used for medical purposes [76], and the ID for solving the maze problem [113]. More details of these IDs are given in Appendix B.2. The set of constraint rules of each ID is detailed in Appendix B.3.

We compare the ID evaluation including constraint rules represented as NTs and BTs (NTWC and BTWC schemes in the graphics) with their counterparts without including them (NT and BT schemes). The ID evaluation with BTs for representing potentials without the application of constraint rules was explained in Chapter 8. For simplicity, we only consider the pruning thresholds $\varepsilon = 0.0$, $\varepsilon = 0.05$ and $\varepsilon = 0.5$. In addition, we compare our approach with the evaluation using tables, which does not admit the application of constraint rules.

Figure 10.2 shows the storage requirements using each one of the considered schemas. In particular, we show the average total potential size (i.e. number of nodes of all the potentials and BCTs) along the evaluation. The measurements were performed before and after the removal of each variable, i.e. in steps 7 and 16 of Algorithm 19. For a better understanding and interpretation of the results, each bar in the graphic is labelled with the space savings[5] obtained w.r.t. the evaluation with tables (which does not admit the application of constraint rules).

As expected, the application of constraints represented as BT implies a reduction in the storage requirements (i.e., BTWC w.r.t. BT) during the evaluation of most of the IDs. Only for a small one, namely Reactor, the storage requirements are increased: the reduction due to the asymmetries is insignificant compared with the additional space required for storing the BCTs. This is not the case of the largest IDs (i.e. NHL and Maze) where, indeed, the highest reductions are obtained. If we consider the evaluation with NTs (i.e. NTWC and NT schemes), we can observe that there are not great differences for all the IDs but for NHL. With this kind of tree, many of the impossible configurations (labelled with 0.0) in the potentials of the small IDs cannot be grouped together.

A reduction in the size of the potentials should lead to a reduction in the computation time. Figure 10.3 shows the running times and the relative durations (i.e., speed up[6]) when compared with the evaluation with tables. We consider as computation time as the time required for building all the trees from tables and the

[5] The concept of *space savings* was introduced in page 168. In short, a positive value implies a reduction in the storage requirements.

[6] The concept of *speed up* was introduced in page 168. A value higher to 1 implies a reduction in the computation time.

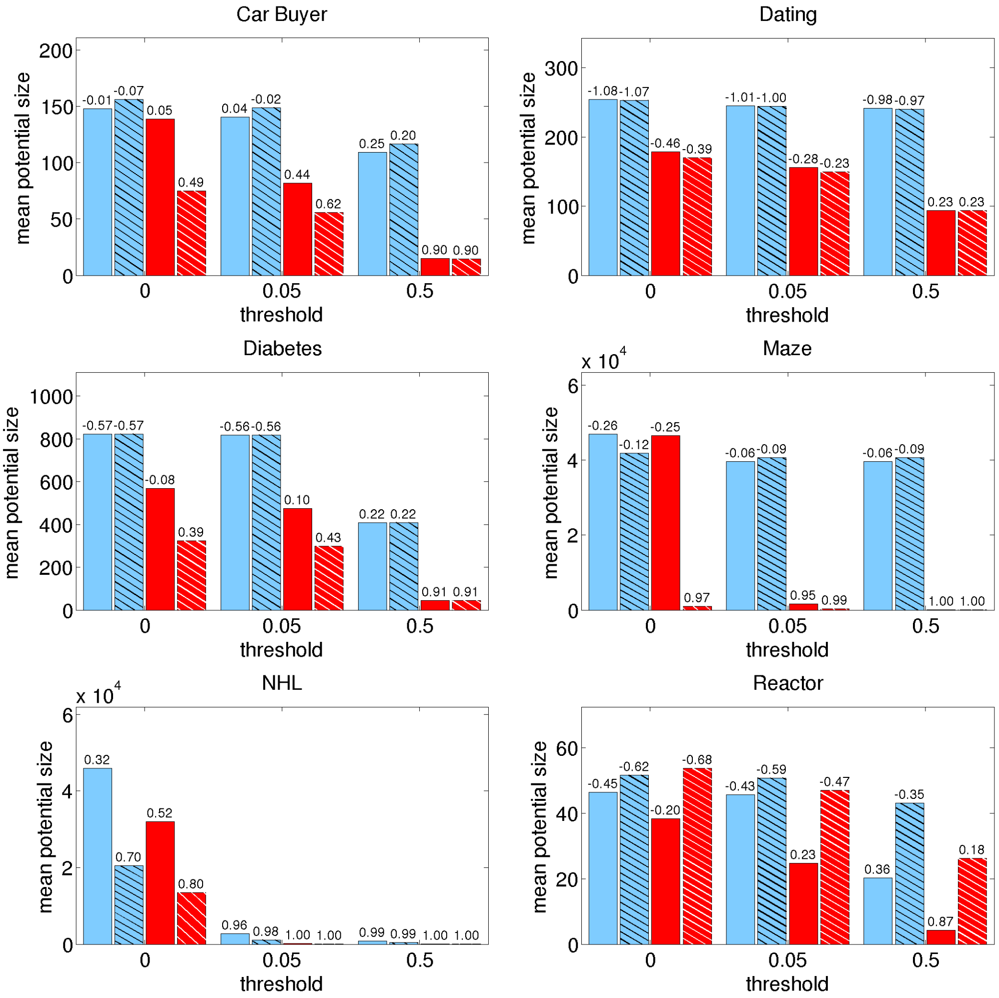


Figure 10.2: Storage requirements for evaluating six IDs where potentials are represented using as trees (NTs and BTs) with and without constraint rules.

time for performing the evaluation (i.e. removal loop). For most the IDs, the best results are obtained with the scheme BTWC, i.e., the application of constraints represented as BTs improves the efficiency of the evaluation. Only for the ID representing the Reactor problem, the computation time is increased: as this is a small ID, the reduction in the number of scenarios to consider does not compensate the overhead introduced by the application of constraints. Additionally, we can observe that the reduction in the computation time is not as important as the reduction in the storage requirements: the application of constraints does not reduce the initialization time (it is actually increased). The same holds for the evaluation with high pruning thresholds values. Thus, great reductions will be obtained when evaluating large networks (such as the NHL and Maze IDs). If we compare the running time for the schemes NT and NTWC, we see that the computation time is, in most of the cases, increased: as the size of potentials is barely reduced, the application of constraints is counter-productive due to the overhead introduced.

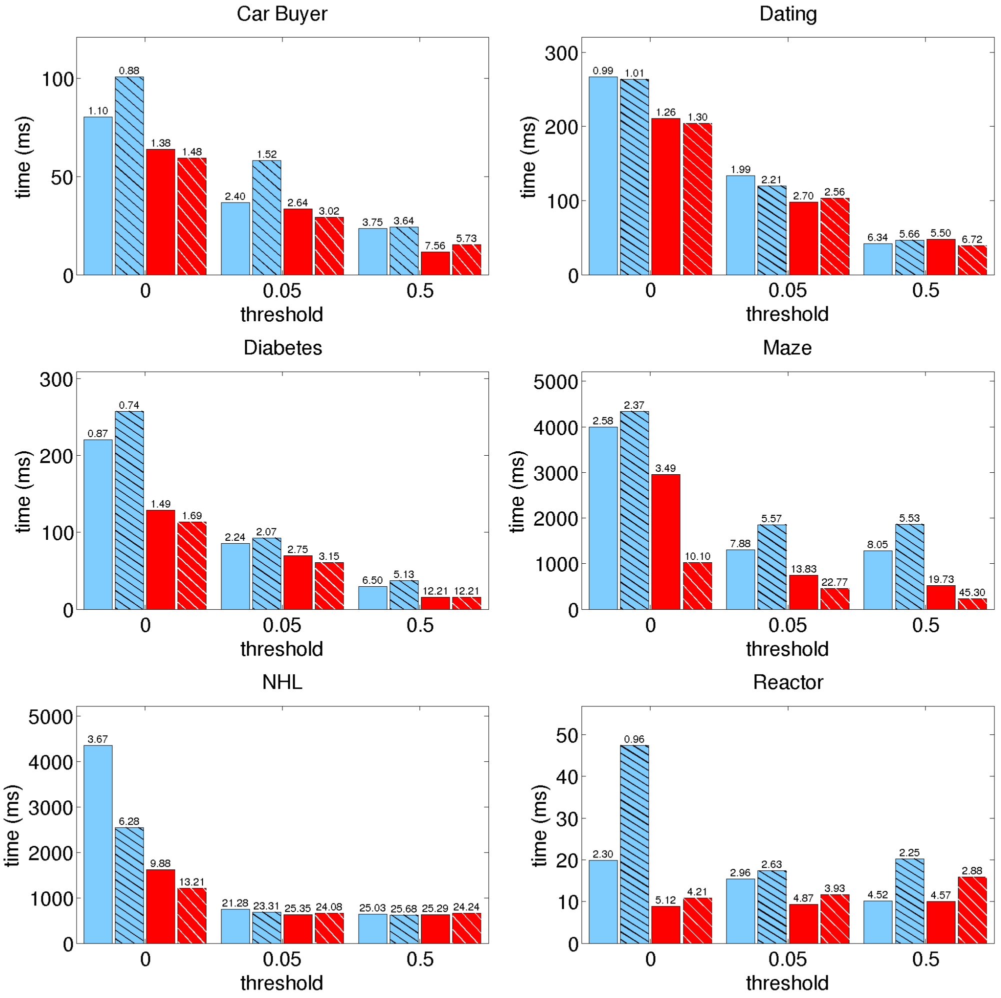


Figure 10.3: Running time for evaluating six IDs where potentials are represented using as trees (NTs and BTs) with and without constraint rules.

10.5 Conclusions

In the present chapter, we have proposed a new method for representing and evaluating asymmetric decision problems with IDs: potentials and asymmetries (constraint trees) are represented using BTs. Using this kind of representation allows to reduce the number of scenarios to consider and also to approximate the potentials. This chapter shows how constraints can be used to improve the efficiency of the VE algorithm. In the experimental work, it has been proved that evaluating IDs with BCTs improves the efficiency and reduces the storage requirements, specially for large IDs. By contrast, if the ID is too small, the application of the constraints might be counter-productive (the application of asymmetries has an overhead). If we compare BTs and NTs for representing asymmetries, we have seen that the efficiency is improved even in many cases where the use of NTs is counter-productive.

As regards to future directions of research, we shall study the behaviour of BTs with constraints using alternatives to the VE inference algorithm, like *arc reversal* [102], *lazy evaluation* [79], etc.

Chapter 11

Evaluation with Interval-valued Potentials

11.1 Introduction

In Chapter 7 we formalized the concept of *interval-valued potentials*. Here We extend to the interval-valued case the formalism of IDs by keeping the same sensitivity-analysis interpretation of credal networks [37]. Such generalized ID is therefore equivalent to a collection of classical (i.e., "precise") IDs whose parameters are consistent with the interval constraints. In this framework, the expected utility of a policy becomes interval-valued. A decision criterion to detect the optimal decision when comparing intervals is therefore needed. We adopt a conservative approach, called *interval dominance* in the imprecise-probability jargon [109], which rejects all the decisions leading to certainly sub-optimal strategies.

In this chapter, some standard approaches to IDs evaluation, namely *variable elimination* [64, 116] and *arc reversal* [102], are generalized in order to cope with the interval-valued case. The extension to intervals does not increase the computational complexity which remains the same as with sharp parameters for both the algorithms. This is achieved at the price of an outer approximation in the inferences, which is required to preserve the interval-valued modelling. An experimental comparison against the arc reversal technique proposed by Fertig and Breese in [51, 50, 8] (i.e., the only practical approach proposed so far these models) shows a clear improvement in terms of both evaluation time and accuracy.

Additionally, the proposed algorithms can be also used for practical sensitivity analysis in (standard) IDs. By replacing the sharp values of some parameters with intervals, we can decide whether or not the original optimal strategy is robust with respect to a perturbation consistent with the intervals. The maximal level of perturbation leaving the strategy unchanged can be therefore regarded as a robustness descriptor. This allows to identify the more critical parameters of the model and, for instance, deciding which ones deserve a more careful elicitation.

The chapter is organized as follows. Section 11.2 defines interval-valued influence diagrams and the basic operations for their evaluations corresponding are detailed in Section 11.3. The algorithms to evaluate interval-valued IDs are in Section 11.2, while the procedure for sensitivity analysis is in Section 11.5. The empirical analysis is presented in Section 11.6.

11.2 Interval-valued influence diagrams

IDs can be extended to intervals just by replacing the PPs and UPs with an equal number of IPPs and IUPs defined over the same domains. A model of this kind is called an *interval-valued influence diagram* (IID). As an example, the interval-valued potentials in Example 31 (page 139) can be used to transform the oil wildcatter's ID into an IID as follows.

Example 39 (the oil wildcatter's IID) *Figure 11.1 depicts the graph of an IID modelling the decision problem described in Example 10. The set of chance variables is $\mathcal{U}_C = \{S, O\}$, while the set of decisions is $\mathcal{U}_D = \{T, D\}$. The utility nodes P and C describe the profit possibly obtained from the presence of oil and the cost of the tests. The sets of interval-valued potentials are $\underline{\overline{\Phi}} = \{\underline{\overline{\phi}}(O), \underline{\overline{\phi}}(S|O,T)\}$, and $\underline{\overline{\Psi}} = \{\underline{\overline{\psi}}(T), \underline{\overline{\psi}}(O,D)\}$. The lower and upper bounds these of potentials are* *reported below in a table form.*

$$\underline{\overline{\phi}}(O) = \begin{array}{c c} \begin{bmatrix} [.475,.525] \\ [.285,.335] \\ [.190,.240] \end{bmatrix} & \begin{array}{c} e \\ w \\ s \end{array} \end{array} \qquad \underline{\overline{\psi}}(T) = \begin{array}{c c} \begin{bmatrix} [-10,-5] \\ [-5,5] \end{bmatrix} & \begin{array}{c} t \\ nt \end{array} \end{array}$$

$$\underline{\overline{\psi}}(O,D) = \begin{array}{c c} \begin{array}{cc} d & nd \end{array} & \\ \begin{bmatrix} [-75,-65] & [-5,5] \\ [45,55] & [-5,5] \\ [195,205] & [-5,5] \end{bmatrix} & \begin{array}{c} e \\ w \\ s \end{array} \end{array}$$

$$\underline{\overline{\phi}}(S|O,T) = \begin{array}{c c} \begin{array}{cccccc} & t & & & nt & \\ e & w & s & e & w & s \end{array} & \\ \begin{bmatrix} [.095,.145] & [.285,.335] & [.475,.525] & [.317,.367] & [.317,.367] & [.317,.367] \\ [.288,.335] & [.380,.430] & [.380,.430] & [.317,.367] & [.317,.367] & [.317,.367] \\ [.570,.620] & [.285,.335] & [.095,.145] & [.317,.367] & [.317,.367] & [.317,.367] \end{bmatrix} & \begin{array}{c} c \\ o \\ d \end{array} \end{array},$$

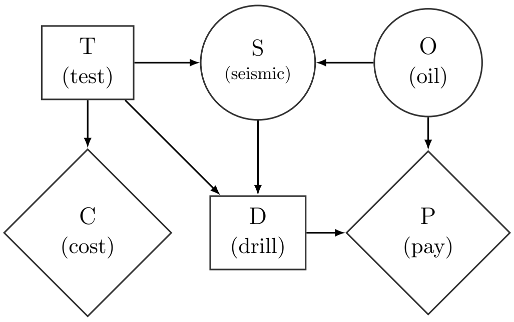


Figure 11.1: Graph of an IID modelling the oil wildcatter's decision problem

IIDs offer a direct sensitivity analysis interpretation. An IID can be regarded as a collection of so-called *consistent* IDs, all with the same graph and set of variables, with PPs and UPs taking their values from the extensions of the IPPs and IUPs of the IID. Note that the ID described in Example 11 is one of the IDs represented by the previous IID.

Considering this sensitivity analysis interpretation, IID evaluation is therefore intended as the calculation of the interval spanned by the MEU values of the consistent IDs. We similarly define the optimal policies of an IID as the union of those optimal, in the sense of Equation (4.24), for at least a consistent ID. This set of policies will be called *credal policy*. An example of the credal policy for the oil wildcatter's IID will be given in Section 11.4.1.3.

11.3 Basic operations for evaluating IIDs

To evaluate an IID, operations considered in Section 4.3.2.1 for precise potentials need to be extended to intervals. For such definitions, we will keep the same sensitivity analysis interpretation: an interval-valued potential represents a set of precise potentials consistent with the interval constraints. Thus, the result of operating over such interval-valued potential will be a new imprecise potential that represents (among others) all possible precise resulting potentials.

We will start by extending the combination operation to intervals. Like in the precise case (Definition 16 in page 59), this operation represents aggregation of knowledge. Given two interval-valued potentials (no matter whether IUPs or IPPs) their combination gives as a result a new interval potential whose domain is the union of the domains of both potentials. In the evaluation of an ID with interval constraints, depending on the potentials involved, we can distinguish two types of combination: *multiplication* and *addition*. Two IPPs are combined using the multiplication while two IUPs are combined with the addition instead (we assume an additive model). When combining an IPP with an IUP, we will proceed as in the case of two IPPs. Thus, the combination can be defined as follows:

Definition 43 (combination of interval-valued potentials) *The combination (addition)* $\overline{\underline{\psi + \psi'}}$ *of two IUPs, say* $\overline{\underline{\psi}}(\mathbf{X}_I)$ *and* $\overline{\underline{\psi}}'(\mathbf{X}_J)$*, is an IUP over* $\mathbf{X}_{I\cup J}$ *such that*

$$\overline{\psi + \psi'}(\mathbf{x}_{I\cup J}) := \overline{\psi}(\mathbf{x}_I) + \overline{\psi}'(\mathbf{x}_J)\,, \tag{11.1}$$

$$\underline{\psi + \psi'}(\mathbf{x}_{I\cup J}) := \underline{\psi}(\mathbf{x}_I) + \underline{\psi}'(\mathbf{x}_J)\,, \tag{11.2}$$

for each $\mathbf{x}_{I\cup J} \in \Omega_{\mathbf{X}_{I\cup J}}$*, with* $\mathbf{x}_I, \mathbf{x}_J \sim \mathbf{x}_{I\cup J}$*. The combination (multiplication)* $\overline{\underline{\phi \cdot \psi}}$ *of an IPP* $\overline{\underline{\phi}}(\mathbf{X}_I|\mathbf{X}_J)$ *with an IUP* $\overline{\underline{\psi}}(\mathbf{X}_K)$ *is an IUP over* $\mathbf{X}_{I\cup J\cup K}$ *such that*

$$\overline{\phi\cdot\psi}(\mathbf{x}_{I\cup J\cup K}) := \left\{ \begin{array}{ll} \overline{\phi}(\mathbf{x}_I|\mathbf{x}_J)\cdot\overline{\psi}(\mathbf{x}_K) & if\ \ \overline{\psi}(\mathbf{x}_K) > 0 \\ \underline{\phi}(\mathbf{x}_I|\mathbf{x}_J)\cdot\overline{\psi}(\mathbf{x}_K) & otherwise \end{array} \right\} \tag{11.3}$$

$$\underline{\phi\cdot\psi}(\mathbf{x}_{I\cup J\cup K}) := \left\{ \begin{array}{ll} \underline{\phi}(\mathbf{x}_I|\mathbf{x}_J)\cdot\underline{\psi}(\mathbf{x}_K) & if\ \ \underline{\psi}(\mathbf{x}_K) > 0 \\ \overline{\phi}(\mathbf{x}_I|\mathbf{x}_J)\cdot\underline{\psi}(\mathbf{x}_K) & otherwise \end{array} \right\} \tag{11.4}$$

for each $\mathbf{x}_{I\cup J\cup K} \in \Omega_{\mathbf{X}_{I\cup J\cup K}}$*, with* $\mathbf{x}_I, \mathbf{x}_J, \mathbf{x}_K \sim \mathbf{x}_{I\cup J\cup K}$*. Finally, the combination (multiplication)* $\overline{\underline{\phi\cdot\phi'}}$ *of two IPPs, say* $\overline{\underline{\phi}}(\mathbf{X}_I|\mathbf{X}_J)$ *and* $\overline{\underline{\phi}}'(\mathbf{X}_K|\mathbf{X}_L)$ *is an IPP over* $\mathbf{X}_{I\cup K}$ *given* $\mathbf{X}_{(J\cup L)\setminus(I\cup K)}$ *such that*

$$\overline{\phi\cdot\phi'}(\mathbf{x}_{I\cup K}|\mathbf{x}_{(J\cup L)\setminus(I\cup K)}) := \overline{\phi}(\mathbf{x}_I|\mathbf{x}_J)\cdot\overline{\phi'}(\mathbf{x}_K|\mathbf{x}_L)\,, \tag{11.5}$$

$$\underline{\phi\cdot\phi'}(\mathbf{x}_{I\cup K}|\mathbf{x}_{(J\cup L)\setminus(I\cup K)}) := \underline{\phi}(\mathbf{x}_I|\mathbf{x}_J)\cdot\underline{\phi'}(\mathbf{x}_K|\mathbf{x}_L)\,, \tag{11.6}$$

for each $\mathbf{x}_{I\cup K} \in \Omega_{\mathbf{X}_{I\cup K}}$ *and* $\mathbf{x}_{(J\cup L)\setminus(I\cup K)} \in \Omega_{\mathbf{X}_{(J\cup L)\setminus(I\cup K)}}$*, with* $\mathbf{x}_I, \mathbf{x}_J, \mathbf{x}_K, \mathbf{x}_L \sim \mathbf{x}_{I\cup K}, \mathbf{x}_{(J\cup L)\setminus(I\cup K)}$.

Example 40 (interval-valued potential combination) *Consider the IPPs and IUPs in Example 31 associated to the oil wildcatter's ID. It is a straightforward exercise to check that the following combined potentials*

$$\begin{array}{rcl} \overline{\underline{\psi}}(T,O,D) & := & \overline{\underline{\psi}}(T) + \overline{\underline{\psi}}(O,D)\,, \\ \overline{\underline{\phi}}(S,O|T) & := & \overline{\underline{\phi}}(O)\cdot\overline{\underline{\phi}}(S|O,T)\,, \\ \overline{\underline{\psi}}(S,O,T,D) & := & \overline{\underline{\phi}}(S,O|T)\cdot\overline{\underline{\psi}}(O,D)\,, \end{array}$$

have the following numerical values

$$\overline{\underline{\psi}}(T,O,D) = \begin{array}{ccccc} t & & nt & & \\ d & nd & d & nd & \\ \left[\begin{array}{r} [-90,-70] \\ [30,50] \\ [180,200] \end{array}\right. & \begin{array}{r} [-20,0] \\ [-20,0] \\ [-20,0] \end{array} & \begin{array}{r} [-80,-60] \\ [40,60] \\ [190,210] \end{array} & \left.\begin{array}{r} [-10,10] \\ [-10,10] \\ [-10,10] \end{array}\right] & \begin{array}{c} e \\ w \\ s \end{array} \end{array}\,,$$

$$\overline{\underline{\phi}}(S,O|T) = \begin{array}{cccccc|c} & t & & & nt & & \\ e & w & s & e & w & s & \\ \hline [.045,.076] & [.081,.112] & [.090,.126] & [.150,.192] & [.090,.123] & [.060,0.088] & c \\ [.135,.176] & [.108,.144] & [.072,.103] & [.150,.192] & [.090,.123] & [.060,0.088] & o \\ [.271,.326] & [.081,.112] & [.018,.035] & [.150,.192] & [.090,.123] & [.060,0.088] & d \end{array} ,$$

$$\overline{\underline{\psi}}(S,O,T,D) = \begin{array}{cccc|cc} t & & nt & & & \\ d & nd & d & nd & & \\ \hline [-5.709,-2.933] & [-.381,.381] & [-14.437,-9.777] & [-.962,.962] & e & \\ [3.655,6.172] & [-.561,.561] & [4.061,6.756] & [-.614,.614] & w & c \\ [17.599,25.83] & [-.630,.630] & [11.732,18.040] & [-.440,.440] & s & \\ [-13.191,-8.799] & [-.879,.879] & [-14.437,-9.777] & [-.962,.962] & e & \\ [4.873,7.923] & [-.720,.720] & [4.061,6.756] & [-.614,.614] & w & o \\ [14.079,21.156] & [-.516,.516] & [11.732,18.040] & [-.440,.440] & s & \\ [-24.413,-17.599] & [-1.627,1.627] & [-14.437,-9.777] & [-.962,.962] & e & \\ [3.655,6.172] & [-.561,.561] & [4.061,6.756] & [-.614,.614] & wd & \\ [3.520,7.134] & [-.174,.174] & [11.732,18.040] & [-.440,.440] & s & \end{array} .$$

The above combination operator generalizes the combination of precise potentials in Definition 16 by keeping the same commutative and associative properties. A deeper characterization is provided by the following result, which provides a sensitivity-analysis justification for the proposed generalization of the combination operator.

Proposition 9 *Given two potentials (no matter whether IUPs or IPPs)* $\overline{\underline{\psi}}$ *and* $\overline{\underline{\phi}}$*, the combination of the elements of their extensions is included in the extension of their combination, i.e.,*

$$\left\{ \psi \otimes \phi \,\middle|\, \psi \in \overline{\underline{\psi}}^{*}, \phi \in \overline{\underline{\phi}}^{*} \right\} \subseteq \overline{\underline{\psi \otimes \phi}}^{*} . \tag{11.7}$$

The proof easily follows from the fact that potentials consistent with the bounds on the right-hand side of Equations (11.2-11.6) cannot produce bounds not consistent with those on the left-hand side. The other operations over sharp-valued potentials required by the usual evaluation algorithms (division, sum-marginalization and max-marginalization) can also be generalized to intervals and similar characterizations provided.

Definition 44 (dividing interval-valued potentials) *The ratio between an IUP $\overline{\underline{\psi}}(\mathbf{X}_I)$ and an IPP $\overline{\underline{\phi}}(\mathbf{X}_J)$ is an IUP $\overline{\underline{\psi/\phi}}$ over $\mathbf{X}_{I\cup J}$ such that, for each $\mathbf{x}_{I\cup J} \in \Omega_{\mathbf{X}_{I\cup J}}$,*

$$\overline{\psi/\phi}(\mathbf{x}_{I\cup J}) := \overline{\psi}(\mathbf{x}_I)/\underline{\phi}(\mathbf{x}_J)\,, \tag{11.8}$$

$$\underline{\psi/\phi}(\mathbf{x}_{I\cup J}) := \underline{\psi}(\mathbf{x}_I)/\overline{\phi}(\mathbf{x}_J)\,, \tag{11.9}$$

with $\mathbf{x}_I, \mathbf{x}_J \sim \mathbf{x}_{I\cup J}$.

The ratio of two IPPs is analogously defined. With zero denominators, the result is set to $+\infty$ for positive numerators and $-\infty$ for negative ones. When both numerator and denominator are zero, we set $\frac{0}{0} = 0$.

Definition 45 (sum-marginalization) *The sum-marginalization $\overline{\underline{\sum_X \psi}}$ of an IUP $\overline{\underline{\psi}}(X, \mathbf{X}_I)$ is an IUP over $\mathbf{X}_I$ such that*

$$\overline{\sum_X \psi}(\mathbf{x}_I) := \sum_{x\in\Omega_X} \overline{\psi}(x, \mathbf{x}_I)\,, \tag{11.10}$$

$$\underline{\sum_X \psi}(\mathbf{x}_I) := \sum_{x\in\Omega_X} \underline{\psi}(x, \mathbf{x}_I)\,, \tag{11.11}$$

for each $\mathbf{x}_I \in \Omega_{\mathbf{X}_I}$.

The sum-marginalization of an IPP is analogously defined.

Definition 46 (max-marginalization) *The max-marginalization $\overline{\underline{\max_D \psi}}$ of an IUP $\overline{\underline{\psi}}(D, \mathbf{X}_I)$ is an IUP over $\mathbf{X}_I$ such that*

$$\overline{\max_D \psi}(\mathbf{x}_I) := \max_{d\in\Omega_D} \overline{\psi}(d, \mathbf{x}_I)\,, \tag{11.12}$$

$$\underline{\max_D \psi}(\mathbf{x}_I) := \max_{d\in\Omega_D} \underline{\psi}(d, \mathbf{x}_I)\,, \tag{11.13}$$

for each $\mathbf{x}_I \in \Omega_{\mathbf{X}_I}$.

The max-marginalization of an IPP is analogously defined. Envelope theorems analogous to that in Proposition 9 can be proved for all the operators defined in this section.

Example 41 (marginalization and division) *Consider the interval-valued potentials $\underline{\overline{\psi}}(S,O,T,D)$ and $\underline{\overline{\phi}}(S,O|T)$ obtained in Example 40. By sum-marginalizing out the variable O, we obtain*

$$\underline{\overline{\psi}}(S,T,D) = \textstyle\sum_O \underline{\overline{\psi}}(S,O,T,D) =$$

$$\begin{array}{cccccc} & t & & nt & & \\ & d & nd & d & nd & \\ \left[\begin{array}{c} \\ \\ \\ \end{array}\right. & \begin{array}{r} [15.544, 29.069] \\ [5.762, 20.279] \\ [-17.238, -4.292] \end{array} & \begin{array}{c} [-1.572, 1.572] \\ [-2.116, 2.116] \\ [-2.363, 2.363] \end{array} & \begin{array}{c} [1.356, 15.019] \\ [1.356, 15.019] \\ [1.356, 15.019] \end{array} & \begin{array}{c} [-2.017, 2.017] \\ [-2.017, 2.017] \\ [-2.017, 2.017] \end{array} & \left.\begin{array}{c} \\ \\ \\ \end{array}\right] \begin{array}{c} c \\ o \\ d \end{array} \end{array},$$

$$\underline{\overline{\phi}}(S|T) = \sum_O \underline{\overline{\phi}}(S,O|T) = \begin{array}{cc} t & nt \\ \left[\begin{array}{rc} [.217, .314] & [.301, .403] \\ [.316, .423] & [.301, .403] \\ [.37, .473] & [.301, .403] \end{array}\right] & \begin{array}{c} c \\ o \\ d \end{array} \end{array}.$$

The division of the previously obtained IUP and IPP gives the following IUP

$$\underline{\overline{\psi}}_2(S,T,D) = \frac{\underline{\overline{\psi}}(S,T,D)}{\underline{\overline{\phi}}(S|T)} =$$

$$\begin{array}{cccccc} & t & & nt & & \\ & d & nd & d & nd & \\ \left[\begin{array}{c} \\ \\ \\ \end{array}\right. & \begin{array}{r} [49.45, 134.207] \\ [13.617, 64.201] \\ [-46.585, -9.084] \end{array} & \begin{array}{c} [-7.256, 7.256] \\ [-6.698, 6.698] \\ [-6.385, 6.385] \end{array} & \begin{array}{c} [3.363, 49.924] \\ [3.363, 49.924] \\ [3.363, 49.924] \end{array} & \begin{array}{c} [-6.704, 6.704] \\ [-6.704, 6.704] \\ [-6.704, 6.704] \end{array} & \left.\begin{array}{c} \\ \\ \\ \end{array}\right] \begin{array}{c} c \\ o \\ d \end{array} \end{array},$$

Finally, the max-marginalization of D from $\underline{\overline{\psi}}_2(S,T,D)$ *is*

$$\underline{\overline{\psi}}(S,T) = \max_D \underline{\overline{\psi}}_2(S,T,D) = \begin{array}{cc} t & nt \\ \left[\begin{array}{cc} [49.45, 134.207] & [3.363, 49.924] \\ [13.617, 64.201] & [3.363, 49.924] \\ [-6.385, 6.385] & [3.363, 49.924] \end{array}\right. & \left.\right] \end{array} \begin{array}{c} \\ c \\ o \\ d \end{array} .$$

11.4 New evaluation algorithms for IIDs

Both the VE and AR schemes can be adopted for IIDs evaluation by replacing the operations over sharp potentials with the analogous operations for interval-valued potentials defined in Section 11.3. We show that this approach might produce unnecessarily large outer approximations. To avoid that, we propose a sophistication of these algorithms based on linear programming (Section 11.4.1 for VE and Section 11.4.3 for AR) as well as an alternative VE which gives faster but less accurate inferences (Section 11.4.2). The latter approach gives an outer approximation analogous to the generalization of the AR algorithm proposed by Fertig and Breese [51, 50, 8].

11.4.1 Variable elimination in IIDs by linear programming

Consider the VE scheme outlined by Algorithm 1 (page 74). The procedure to eliminate a variable, detailed by Algorithm 2, is based on two sequential steps: first the potentials including the variable to eliminate in their arguments are combined (line 2), then the elimination is performed on the combined potential (lines 4 or 6). When coping with IIDs, we perform the last combination together with the elimination. This corresponds to a linear program, that avoids unnecessary additional approximations.

11.4.1.1 Chance variables elimination from IPPs

Here we will explain how to eliminate a chance variable, say Y, from a set of IPPs. Because of Definition 15, only one of the PPs to be combined in line 2 of Algorithm 2 has Y on the left-hand side of its argument. The same holds with the IPPs. When eliminating a chance variable from the IPPs of an IID, we proceed as follows. First we combine (Definition 43) all the IPPs apart from the one having Y on the left. The resulting IPP is combined indeed with the only IPP having Y on the left-hand side and, simultaneously, the variable is sum-marginalized (first term in line 4 of Algorithm 2) as described by Definition 45. The procedure is detailed here below.

Definition 47 (eliminating chance variables from IPPs) *Consider the elimination of the chance variable Y during VE. Let $\overline{\underline{\phi}}(\mathbf{X}_I|\mathbf{X}_J, Y)$ denote the IPP obtained by combining all the IPPs with Y on the right-hand side, and $\overline{\underline{\phi}}(Y, \mathbf{X}_K|\mathbf{X}_L)$ the only IPP with Y on the left. The elimination of Y from the combination of these two IPPs generates an IPP $\overline{\underline{\phi}}(\mathbf{X}_K, \mathbf{X}_I|\mathbf{X}_L, \mathbf{X}_J)$. For each $\mathbf{x}_{I\cup K} \in \Omega_{\mathbf{X}_{I\cup K}}$ and $\mathbf{x}_{L\cup J} \in \Omega_{\mathbf{X}_{L\cup J}}$, an outer approximation of the lower bound $\underline{\phi}(\mathbf{x}_{K\cup I}|\mathbf{x}_{L\cup J})$ is the solution of the following task:*

$$\begin{aligned}
\textit{minimize} \quad & \sum_{y\in\Omega_Y} \phi(\mathbf{x}_I|\mathbf{x}_J, y) \cdot \phi(y, \mathbf{x}_K|\mathbf{x}_L)\,, \\
\textit{subject to} \quad & \underline{\phi}(\mathbf{x}_I|\mathbf{x}_J, y) \leq \phi(\mathbf{x}_I|\mathbf{x}_J, y) \leq \overline{\phi}(\mathbf{x}_I|\mathbf{x}_J, y)\,, \\
& \underline{\phi}(y, \mathbf{x}_K|\mathbf{x}_L) \leq \phi(y, \mathbf{x}_K|\mathbf{x}_L) \leq \overline{\phi}(y, \mathbf{x}_K|\mathbf{x}_L), \forall y \in \Omega_Y\,.
\end{aligned}$$

Analogously, an outer approximation of the upper bound $\overline{\phi}(\mathbf{x}_{K\cup I}|\mathbf{x}_{L\cup J})$ can be calculated by maximizing the previous objective function instead.

In the previous linear program, the optimization variables[1] $\{\phi(\mathbf{x}_I|\mathbf{x}_J, y)\}_{y\in\Omega_Y}$ are free to vary one independently of the other. Each one of these variables is in a different term of the objective function. Thus, in case of computing the lower bound,

[1] The optimization variables should not be confused with the variables in an ID. A linear program is solved when the best values of the variables (given an objective function) have been identified.

we can easily minimize with respect to these variables and replace $\phi(\mathbf{x}_I|\mathbf{x}_J,y)$ with the lower bound $\underline{\phi}(\mathbf{x}_I|\mathbf{x}_J,y)$. This reduces the task to a linear program over the optimization variables $\{\phi(y,\mathbf{x}_K|\mathbf{x}_L)\}_{y\in\Omega_Y}$. Analogously, when computing the upper bound, we can replace $\phi(\mathbf{x}_I|\mathbf{x}_J,y)$ with $\overline{\phi}(\mathbf{x}_I|\mathbf{x}_J,y)$.

It should be noticed that the optimization variables $\{\phi(y,\mathbf{x}_K|\mathbf{x}_L)\}_{y\in\Omega_Y}$ are not only required to satisfy the separate constraints reported in the above task, but also the normalization constraint of the PPs consistent with the IPP $\overline{\underline{\phi}}(y,\mathbf{x}_K|\mathbf{x}_L)$. These are constraints among the different tasks corresponding to the different values of $\mathbf{x}_K$. By considering the reachability constraints for $\overline{\underline{\phi}}(\mathbf{X}_K|\mathbf{X}_L) := \sum_Y \overline{\underline{\phi}}(\mathbf{X}_K,Y|\mathbf{X}_L)$ (see Definition 18), we have

$$1-\sum_{\mathbf{x}'_K\neq\mathbf{x}_K,y}\overline{\phi}(y,\mathbf{x}'_K|\mathbf{x}_L)\leq\sum_y\phi(y,\mathbf{x}_K|\mathbf{x}_L)\leq 1-\sum_{\mathbf{x}'_K\neq\mathbf{x}_K,y}\underline{\phi}(y,\mathbf{x}'_K|\mathbf{x}_L)\,. \tag{11.14}$$

Note that if $X_K=\emptyset$ the constraint in Equation (11.14) degenerates in the trivial normalization of the potential and becomes useless.

Example 42 *Consider the elimination of the chance variable O from the IID associated to the graph in Figure 4.2 with the IPPs and IUPs as in Example 31. This consists in the combination $\overline{\underline{\phi}}(S,O|T):=\overline{\underline{\phi}}(O)\cdot\overline{\underline{\phi}}(S|O,T)$, and then the sum-marginalization $\overline{\underline{\phi}}(S|T):=\sum_o\overline{\underline{\phi}}(S,o|T)$. To show how this works, let us compute the upper bound $\overline{\phi}(S=c|T=t)$. The corresponding linear program is:*

$$\begin{aligned}
\textit{maximize}\quad & 0.145\cdot\phi(e)+0.335\cdot\phi(w)+0.525\cdot\phi(s)\,,\\
\textit{subject to}\quad & 0.475\leq\phi(e)\leq 0.525\,,\\
& 0.285\leq\phi(w)\leq 0.335\,,\\
& 0.190\leq\phi(s)\leq 0.240\,,\\
& \phi(e)+\phi(w)+\phi(s)=1\,.
\end{aligned}$$

The objective function is maximized when $\phi(e)=0.475$, $\phi(w)=0.285$ and $\phi(s)=0.240$, which gives $\overline{\phi}(c|t)\simeq 0.290$. By iterating this procedure for all the values of S and T and for the lower bounds too, the following IPP is obtained:

$$\underline{\overline{\phi}}(S|T) = \begin{array}{cc} t & nt \\ \left[\begin{array}{cc} [.221, .290] & [.317, .367] \\ [.330, .385] & [.317, .367] \\ [.375, .444] & [.317, .367] \end{array}\right] & \begin{array}{c} c \\ o \\ d \end{array} \end{array}$$

It is worth noticing that not including the additional constraints in Equation (11.14) *would have produced larger intervals.*

11.4.1.2 Chance variables elimination from IUPs

Let us consider here how the second term of line 4 of Algorithm 2, i.e., the elimination of a chance variable Y from the utility potentials, can be achieved with IIDs. We first combine all the IUPs including Y in their arguments as in Definition 43. For the IPPs, we proceed as in the previous section by first combining all the IPPs with Y on the right-hand side. The remaining combinations and the division are performed simultaneously as described in the following definition.

Definition 48 (eliminating chance variables from IUPs) *Let $\underline{\overline{\phi}}(\mathbf{X}_I|\mathbf{X}_J, Y)$ be the IPP obtained by combining all the IPP with Y on the right-hand side, $\underline{\overline{\phi}}(Y, \mathbf{X}_K|\mathbf{X}_L)$ be the only IPP with Y on the left-hand side and $\underline{\overline{\psi}}(Y, \mathbf{X}_M)$ the combination of all the IUPs with Y in the argument. The elimination of a chance variable Y from the combination of these potentials produces a new IPP $\underline{\overline{\psi}}(\mathbf{X}_I, \mathbf{X}_J, \mathbf{X}_K, \mathbf{X}_L, \mathbf{X}_M)$. For each $\mathbf{x}_{I\cup J\cup K\cup L\cup M} \in \Omega_{\mathbf{X}_{I\cup J\cup K\cup L\cup M}}$, an outer approximation of the lower bound $\underline{\psi}(\mathbf{x}_{I\cup J\cup K\cup L\cup M})$ is the solution of the task*

$$\begin{array}{ll} \textit{minimize} & \dfrac{\sum_{y\in\Omega_Y} \phi(\mathbf{x}_I|\mathbf{x}_J, y) \cdot \phi(y, \mathbf{x}_K|\mathbf{x}_L) \cdot \psi(y, \mathbf{x}_M)}{\sum_{y\in\Omega_Y} \phi(\mathbf{x}_I|\mathbf{x}_J, y) \cdot \phi(y, \mathbf{x}_K|\mathbf{x}_L)}\,, \\ \textit{subject to} & \underline{\phi}(\mathbf{x}_I|\mathbf{x}_J, y) \leq \phi(\mathbf{x}_I|\mathbf{x}_J, y) \leq \overline{\phi}(\mathbf{x}_I|\mathbf{x}_J, y)\,, \\ & \underline{\phi}(y, \mathbf{x}_K|\mathbf{x}_L) \leq \phi(y, \mathbf{x}_K|\mathbf{x}_L) \leq \overline{\phi}(y, \mathbf{x}_K|\mathbf{x}_L)\,, \\ & \underline{\psi}(y, \mathbf{x}_M) \leq \psi(y, \mathbf{x}_M) \leq \overline{\psi}(y, \mathbf{x}_M)\,. \end{array}$$

Analogously, an approximation of the upper bound $\overline{\psi}(\mathbf{x}_{I\cup J\cup K\cup L\cup M})$ can be calculated by maximizing the previous objective function instead.

The task has a linearly constrained cubic fractional objective function. When calculating the lower bound, the minimization with respect to the optimization variables associated to an IUP can be trivially achieved by setting $\psi(y, \mathbf{x}_M) = \underline{\psi}(y, \mathbf{x}_M)$. Similarly, in case of computing the upper bound, $\psi(y, \mathbf{x}_M)$ can be replaced with $\overline{\psi}(y, \mathbf{x}_M)$.

Unlike the task in Definition 47, the optimization with respect to the optimization variables $\{\phi(\mathbf{x}_I|\mathbf{x}_J, y)\}$ is not trivial as the variables appear both in the numerator and in the denominator of the objective function. Nevertheless, we can regard the product $\phi(y, \mathbf{x}_K|\mathbf{x}_L) \cdot \phi(\mathbf{x}_I|\mathbf{x}_J, y)$ as a single optimization variable subject to

$$\underline{\phi}(y, \mathbf{x}_K|\mathbf{x}_L) \cdot \underline{\phi}(\mathbf{x}_I|\mathbf{x}_J, y) \leq \phi(y, \mathbf{x}_K|\mathbf{x}_L) \cdot \phi(\mathbf{x}_I|\mathbf{x}_J, y) \leq \overline{\phi}(y, \mathbf{x}_K|\mathbf{x}_L)\overline{\phi}(\mathbf{x}_I|\mathbf{x}_J, y)\,. \tag{11.15}$$

In this way the task becomes a linear-fractional program which can be reduced to a linear program using the classical Charnes-Cooper transformation [30]. This introduces an outer approximation, which can be partially mitigated by additional reachability constraints as in the previous section. In this case the constraints are

$$\begin{aligned}
&1 - \sum_{\{\mathbf{x}'_K, \mathbf{x}'_I\} \neq \{\mathbf{x}_K, \mathbf{x}_I\}, y} \overline{\phi}(y, \mathbf{x}'_K|\mathbf{x}_L) \cdot \overline{\phi}(\mathbf{x}_I|, \mathbf{x}_J, y) \leq \sum_y \phi(y, \mathbf{x}_K|\mathbf{x}_L) \cdot \phi(\mathbf{x}_I|\mathbf{x}_J, y) \\
&\leq 1 - \sum_{\{\mathbf{x}'_K, \mathbf{x}'_I\} \neq \{\mathbf{x}_K, \mathbf{x}_I\}, y} \underline{\phi}(y, \mathbf{x}'_K|\mathbf{x}_L) \cdot \underline{\phi}(\mathbf{x}'_I|\mathbf{x}_J, y)\,.
\end{aligned} \tag{11.16}$$

As in the previous section, if $\mathbf{X}_K = \mathbf{X}_I = \emptyset$ the constraint becomes ineffective and the problem becomes linear instead of linear-fractional.

Example 43 *Consider the VE scheme applied to the oil wildcutter's IID. To remove the chance variable O from the IUPs, we should consider the IPPs* $\overline{\underline{\phi}}(O)$ *and* $\overline{\underline{\phi}}(S|O,T)$ *and the IUP* $\overline{\underline{\psi}}(O,D)$*. A new IUP* $\overline{\underline{\psi}}(S,T,D)$ *is obtained. The upper bound* $\overline{\psi}(e,t,d)$ *requires the solution of the fractional task*

$$\begin{array}{ll}
\textit{maximize} & \dfrac{-65\cdot\phi(e)\cdot\phi(c|e,t)+55\cdot\phi(w)\cdot\phi(c|w,t)+205\cdot\phi(s)\cdot\phi(c|s,t)}{\phi(e)\cdot\phi(c|e,t)+\phi(w)\cdot\phi(c|w,t)+\phi(s)\cdot\phi(c|s,t)}\,,\\
\textit{subject to} & .475\cdot .095 \leq \phi(e)\cdot\phi(c|e,t) \leq .525\cdot .145\,,\\
 & 0.285\cdot 0.285 \leq \phi(w)\cdot\phi(c|w,t) \leq 0.335\cdot 0.335\,,\\
 & 0.190\cdot 0.475 \leq \phi(s)\cdot\phi(c|s,t) \leq 0.240\cdot 0.525\,,\\
 & 0.217 \leq \phi(e)\cdot\phi(c|e,t)+\phi(w)\cdot\phi(c|w,t)+\phi(s)\cdot\phi(c|s,t) \leq 0.314\,.
\end{array}$$

The maximum is 108.44 *which is achieved when the first two variables takes their minimum value and the third its maximum. By solving similar tasks for each joint state in* $\Omega_S\times\Omega_T\times\Omega_D$*, we obtain the IUP*

$$\overline{\underline{\psi}}(S,T,D) = \begin{array}{c} \\ \left[\begin{array}{cccc} \textit{(t,d)} & \textit{(t,nd)} & \textit{(nt,d)} & \textit{(nt,nd)} \\ {[60.8,108.44]} & [-5.0,5.0] & [3.96,41.57] & [-5.0,5.0] \\ {[16.17,53.0]} & [-5.0,5.0] & [3.96,41.57] & [-5.0,5.0] \\ {[-40.58,-10.27]} & [-5.0,5.0] & [3.96,41.57] & [-5.0,5.0] \end{array}\right] \end{array} \begin{array}{c} \\ c \\ o \\ d \end{array}\,.$$

In this particular case, as $0.475\cdot 0.095+0.285\cdot 0.285+0.19\cdot 0.475\simeq 0.217$*, not including the additional constraints in Equation* (11.16) *does not make the result less accurate.*

11.4.1.3 Decision variables elimination

Here we discuss how to extend the operations in lines 6 and 7 of Algorithm 2 to IIDs. The $\arg\max$ operation is intrinsically related to the fact that a UP has sharp values. To decide the optimal options when comparing intervals, we adopt a conservative approach, called *interval dominance* in the imprecise-probability jargon [109], which rejects all the decisions leading to certainly sub-optimal strategies. The procedure is described here below.

Definition 49 (interval optimality) *Let $\overline{\underline{\psi}}$ be an IUP over $Y \cup \mathbf{X}_I$. An element $y \in \Omega_Y$ is* interval-optimal *given $\mathbf{x}_I \in \Omega_{\mathbf{X}_I}$ if there is no $y' \in \Omega_Y \setminus \{y\}$ such that $\underline{\psi}(y', \mathbf{x}_I) > \overline{\psi}(y, \mathbf{x}_I)$.*

Let D be a decision variable to be eliminated from $\overline{\underline{\psi}}(D, \mathbf{X}_I)$ during the VE (if there are multiple potentials they are combined using Definition 43). To detect the optimal policy $\widehat{\delta}_D(\mathbf{X}_I)$ we compute the interval-optimal states of D given each $\mathbf{x}_I \in \Omega_{\mathbf{X}_I}$. This corresponds to a so-called *credal* policy allowing for indecision between two or more possible options. Finally the maximization of the IUP giving as result a new IUP $\overline{\underline{\psi}}(\mathbf{X}_I)$ is done as explained in Definition 46, i.e., by acting separately on the two bounds. Note that the elimination of a decision does not involve any computation over IPPs.

Example 44 *In the oil wildcatter's IID, the removal of the decision D involves the IUP $\overline{\underline{\psi}}(S, T, D)$ obtained in Example 43. The resulting IUP and the corresponding optimal policy are*

$$\overline{\underline{\psi}}(S,T) = \begin{array}{cc} \begin{array}{cc} t & nt \end{array} & \\ \begin{bmatrix} [60.8, 108.44] & [3.96, 41.57] \\ [16.17, 53.0] & [3.96, 41.57] \\ [-5.0, 5.0] & [3.96, 41.58] \end{bmatrix} & \begin{array}{c} c \\ o \\ d \end{array} \end{array}, \quad \widehat{\delta}_D(S,T) = \begin{array}{cc} \begin{array}{cc} t & nt \end{array} & \\ \begin{bmatrix} \{d\} & \{d, nd\} \\ \{d\} & \{d, nd\} \\ \{nd\} & \{d, nd\} \end{bmatrix} & \begin{array}{c} c \\ o \\ d \end{array} \end{array}.$$

Note that for some configurations, the interval resulting from the maximization corresponds with only one of the alternatives given as optimal.

11.4.2 A faster outer approximation

In this section we propose an alternative approach to the generalization of VE to IIDs, which does not require any linear program to be solved. This corresponds to a heuristic approach to the solution of those optimizations, that introduces an additional outer approximation.

First, the chance variable elimination from IPPs is done like in the precise case but using the operations for interval-valued potentials (Section 11.3). On the other hand, for the chance variable elimination from IUPs, we propose the following procedure: suppose that we aim to remove a variable Y from an IPP $\overline{\underline{\phi}}(Y, \mathbf{X}_I|\mathbf{X}_J)$ and an IUP $\overline{\psi}(Y, \mathbf{x}_K)$. The elimination of Y generates a new IUPs $\overline{\underline{\psi}}(\mathbf{X}_{I\cup J\cup K})$ such that, for each $\mathbf{x}_{I\cup J\cup K} \in \Omega_{\mathbf{X}_{I\cup J\cup K}}$,

$$\overline{\psi}(\mathbf{x}_{I\cup J\cup K}) := \sum_{y\in\Omega_Y} \frac{\overline{\phi}(y, \mathbf{x}_I|\mathbf{x}_J)}{\overline{\phi}(y, \mathbf{x}_I|\mathbf{x}_J) + \sum_{y'\neq y} \underline{\phi}(y', \mathbf{x}_I|\mathbf{x}_J)} \cdot \overline{\psi}(y, \mathbf{x}_K) \tag{11.17}$$

$$\underline{\psi}(\mathbf{x}_{I\cup J\cup K}) = \sum_{y\in\Omega_Y} \frac{\underline{\phi}(y, \mathbf{x}_I|\mathbf{x}_J)}{\underline{\phi}(y, \mathbf{x}_I|\mathbf{x}_J) + \sum_{y'\neq y} \overline{\phi}(y', \mathbf{x}_I|\mathbf{x}_J)} \cdot \underline{\psi}(y, \mathbf{x}_K) \tag{11.18}$$

If there is more than an IPP with Y (either on the left or on right), all the IPPs are combined using Definition 43. Compared to what is done by Definition 48, the above considered potential provides an outer approximation based on the *one-sided* potentials defined in Section 7.3. For the removal of a decision we proceed as described in Section 11.4.1.3. An example of this heuristic approach is shown here below.

Example 45 *The removal of the chance variable O (from the IUPs) involves the IPPs* $\overline{\underline{\phi}}(O)$ *and* $\overline{\underline{\phi}}(S|O,T)$ *and the IUP* $\overline{\underline{\psi}}(O,D)$. *A new IUP* $\overline{\underline{\psi}}(S,T,D)$ *is obtained. E.g., according to Equation.* (11.18), $\overline{\psi}(c,t,d) = 107.302$*, which corresponds to*

$$\overline{\psi}(c,t,d) = \frac{0.525{\cdot}0.145{\cdot}(-65)}{0.525{\cdot}0.145 + 0.285{\cdot}0.285 + 0.19{\cdot}0.475} + \\ + \frac{0.335{\cdot}0.335{\cdot}55}{0.475{\cdot}0.095 + 0.335{\cdot}0.335 + 0.19{\cdot}0.475} + \\ + \frac{0.24{\cdot}0.525{\cdot}205}{0.475{\cdot}0.095 + 0.285{\cdot}0.285 + 0.24{\cdot}0.525} = 107.302$$

By similarly proceeding for all the joint states in $\Omega_S \times \Omega_T \times \Omega_D$, *we obtain the IUP shown below. Note that the intervals in Example 43 are included in those of the current IUP.*

$$\underline{\overline{\psi}}(S,T,D) = \begin{array}{c} \\ \\ \\ \\ \end{array}\begin{array}{cccc|c} \textit{(t,d)} & \textit{(t,nd)} & \textit{(nt,d)} & \textit{(nt,nd)} & \\ \hline [64.124, 107.302] & [-3.849, 6.3] & [10.971, 38.662] & [-4.1, 5.988] & c \\ [21.95, 51.444] & [-4.088, 6.003] & [10.971, 38.662] & [-4.1, 5.988] & o \\ [-32.605, -15.972] & [-4.358, 5.681] & [10.971, 38.662] & [-4.1, 5.988] & d \end{array}$$

11.4.3 Arc reversal in IIDs by linear programming

The AR scheme outlined in Algorithm 3 (page 80) evaluates IDs by performing three basic operations: elimination of chance (Transformation 1) and decision (Transformation 2) variables and arc reversal (Transformation 3). The first operation is just a particular case of the chance variable elimination from IUPs explained in Section 11.4.1.2. Yet, in AR, the removal of a chance variable Y always involves an IPP $\underline{\overline{\phi}}(Y|\mathbf{X}_I)$ such that Y is the only variable on the left-hand side and an IUP $\underline{\overline{\psi}}(Y, \mathbf{X}_J)$. Because of the normalization constraint, $\underline{\overline{\sum_Y \phi(Y|\mathbf{X}_J)}}$ is a constant potential always equal to one and the removal corresponds to a simple linear program without fractional terms. Similarly, the second operation consists in removing a decision variable D from an IUP, say $\underline{\overline{\psi}}(D, \mathbf{X}_I)$, and deciding the optimal policy $\widehat{\delta}_{D_i}$. This is done exactly in the same way as for the VE algorithm (see Section 11.4.1.3). Finally, to reverse arcs, we generalize Transformation 3 to intervals as follows.

Transformation 4 (interval arc reversal) *Assume that the chance nodes Y and X of an IID are directly connected by an arc, but not by other directed paths. Let $\underline{\overline{\phi}}(Y|\mathbf{X}_I)$ and $\underline{\overline{\phi}}(X|Y, \mathbf{X}_J)$ be the relative PPs, which means that $\mathbf{X}_I$ are the direct predecessors of Y and $\mathbf{X}_J$ those of X others than Y. Change the orientation of the arc and add arcs from $\mathbf{X}_I$ towards X and from $\mathbf{X}_J$ towards Y. Then replace the original IPPs with $\underline{\overline{\phi}}(X|\mathbf{X}_I, \mathbf{X}_J)$ and $\underline{\overline{\phi}}(Y|X, \mathbf{X}_I, \mathbf{X}_J)$, where the first IPP is obtained by sum-marginalization and the second is such that $\underline{\phi}(y|x, \mathbf{x}_I, \mathbf{x}_J)$ is the minimum of*

$$\frac{\phi(y|\mathbf{x}_I) \cdot \phi(x|y, \mathbf{x}_J)}{\sum_{y' \in \Omega_Y} \phi(y'|\mathbf{x}_I) \cdot \phi(x|y', \mathbf{x}_J)}, \tag{11.19}$$

with respect to the interval constraints induced by the two original IPPs, for each $x \in \Omega_X$, $y \in \Omega_Y$, and $\mathbf{x}_{I \cup J} \in \Omega_{\mathbf{X}_{I \cup J}}$.

To solve the above optimization, we can use the same optimization strategy considered by Zaffalon in his naive credal classifier [115]. Accordingly, we divide the denominator by the numerator and rewrite Equation (11.19) as

$$\left[1 + \sum_{y' \neq y} \frac{\phi(y'|\mathbf{x}_I) \cdot \phi(x|y', \mathbf{x}_J)}{\phi(y|\mathbf{x}_I) \cdot \phi(x|y, \mathbf{x}_J)}\right]^{-1} . \tag{11.20}$$

Then, as $f(t) = [1 + t]^{-1}$ is a monotone decreasing function of $t \in \mathbb{R}$, we reduce the minimization of the objective function in Equation (11.19) or Equation (11.20), to the maximization of

$$\sum_{y' \neq y} \frac{\phi(y'|\mathbf{x}_I) \cdot \phi(x|y', \mathbf{x}_J)}{\phi(y|\mathbf{x}_I) \cdot \phi(x|y, \mathbf{x}_J)} . \tag{11.21}$$

As there are no constraints over the optimization variables $\{\phi(x|y, \mathbf{x}_J)\}_{y \in \Omega_Y}$ we perform the optimization with respect to these variable and obtain the objective function

$$\sum_{y' \neq y} \frac{\phi(y'|\mathbf{x}_I) \cdot \overline{\phi}(x|y', \mathbf{x}_J)}{\phi(y|\mathbf{x}_I) \cdot \underline{\phi}(x|y, \mathbf{x}_J)} , \tag{11.22}$$

which is a linearly constrained linear-fractional objective function. The task can be therefore reduced to a linear program.

Example 46 *The reversal of the arc from O to S involves the IPPs $\underline{\overline{\phi}}(S|O, T)$ and $\underline{\overline{\phi}}(O)$. The resulting IPPs are $\underline{\overline{\phi}}(S|T)$ and $\underline{\overline{\phi}}(O|S, T)$. Computing the upper bound $\overline{\phi}(e|c, t)$ of the IPP attached to O corresponds to the following linear program*

$$\begin{aligned}
\textit{minimize} \quad & \frac{\phi(w) \cdot 0.285}{\phi(e) \cdot 0.145} + \frac{\phi(s) \cdot 0.475}{\phi(e) \cdot 0.145} , \\
\textit{subject to} \quad & 0.475 \leq \phi(e) \leq 0.525 , \\
& 0.285 \leq \phi(w) \leq 0.335 , \\
& 0.19 \leq \phi(s) \leq 0.24 , \\
& \phi(e) + \phi(w) + \phi(s) = 1 .
\end{aligned}$$

The minimum value of the previous function is 2.2525*, and hence*

$$\overline{\phi}(e|c,t) = \frac{1}{1+2.2525} \simeq 0.307\,.$$

By solving similar linear programs for each bound and state of $\Omega_O \times \Omega_S \times \Omega_T$*, we obtain the IPP*

$$\underline{\overline{\psi}}(S,T,D) = \begin{array}{ccccccc} \textit{(e,t)} & \textit{(e,nt)} & \textit{(w,t)} & \textit{(w,nt)} & \textit{(s,t)} & \textit{(s,nt)} & \\ [.169,.307] & [.439,.561] & [.375,.494] & [.439,.561] & [.66,.766] & [.439,.561] & c \\ [.294,.453] & [.256,.368] & [.292,.41] & [.256,.368] & [.187,.28] & [.256,.368] & o \\ [.333,.499] & [.168,.268] & [.192,.298] & [.168,.268] & [.041,.09] & [.168,.268] & d \end{array} \,.$$

The upper bound $\overline{\phi}(c|t)$ *of the IPP attached to the final parent is the solution of a linear program with the same constraints as in the previous optimization and objective function to be maximized:* $\phi(e) \cdot 0.145 + \phi(w) \cdot 0.335 + \phi(s) \cdot 0.525$*. This function is maximized for* $\phi(e) = 0.475$*,* $\phi(w) = 0.285$ *and* $\phi(s) = 0.24$*. Thus,* $\overline{\phi}(c|t) = 0.29$*. If similar programs are solved for each bound and state of* $\Omega_S \times \Omega_T$*, the following IPP is obtained*

$$\underline{\overline{\phi}}(S|T) = \begin{array}{cccc} c & o & d & \\ [.221,.290] & [.330,.385] & [.375,.449] & t \\ [.317,.367] & [.317,.367] & [.317,.367] & nt \end{array} \,.$$

11.4.4 Complexity analysis

The asymptotic complexity of the algorithms proposed in this section for IIDs evaluation is just the same as that of their IDs counterparts. Roughly speaking, with intervals, we perform the double of the number of arithmetic operations required with sharp values. The time required to run the linear programs is polynomial in the number of variables and constraints, which in turn depends on the arity of the local potentials involved during the elimination.

11.5 Sensitivity analysis

The algorithms for IIDs evaluation developed in Section 11.2 can be used for practical sensitivity analysis in standard IDs. The sharp-valued potentials of an ID can be replaced by interval-valued potentials whose extensions include the original potentials (e.g., as the extensions of the interval-valued potentials in Example 31 contain the potentials in Example 11). These sets of potentials are intended to describe the possible effects of a perturbation of the sharp values of the parameters. In particular, we want to parametrize the level of perturbation as described in the following definition.

Definition 50 (nested perturbations) *Given a potential ψ, no matter whether PP or UP, a* parametrized nested perturbation *of ψ is denoted as $\overline{\underline{\psi}}_\varepsilon$. For each $\varepsilon \geq 0$, $\overline{\underline{\psi}}_\varepsilon$ is an interval-valued potential. We require that: (i) $\varepsilon \leq \varepsilon' \Rightarrow \overline{\underline{\psi}}^*_\varepsilon \subseteq \overline{\underline{\psi}}^*_{\varepsilon'}$; and (ii) $\overline{\underline{\psi}}^*_{\varepsilon=0} = \psi$.*

Let us describe some practical ways to implement nested perturbations as in Definition 50. For UPs, we perform a *rectangular* perturbation symmetrical with respect to the original sharp values. In other words, if $\psi(\mathbf{X}_I)$ is a UP over the set of variables $\mathbf{X}_I$, then $\overline{\underline{\psi}}_\epsilon(\mathbf{X}_I)$ is an IPP such that, for each $\mathbf{x}_I \in \Omega_{\mathbf{X}_I}$,

$$\begin{aligned} \underline{\psi}_\varepsilon(\mathbf{x}_I) &:= \psi(\mathbf{x}_I) - \varepsilon\,, & (11.23)\\ \overline{\psi}_\varepsilon(\mathbf{x}_I) &:= \psi(\mathbf{x}_I) + \varepsilon\,. & (11.24) \end{aligned}$$

Rectangular perturbations cannot be applied to PPs, because of the normalization and non-negativity constraints. We consider instead nested perturbations in the form of ϵ-contaminations. Given an (unconditional) PP ϕ over $\mathbf{X}_I$, the perturbed IPP $\overline{\underline{\phi}}_\epsilon$ is such that, for each $\mathbf{x}_I \in \Omega_{\mathbf{X}_I}$,

$$\begin{aligned} \underline{\phi}_\varepsilon(\mathbf{x}_I) &:= (1-\varepsilon)\cdot\phi(\mathbf{x}_I)\,, & (11.25)\\ \overline{\phi}_\varepsilon(\mathbf{x}_I) &:= (1-\varepsilon)\cdot\phi(\mathbf{x}_I) + \varepsilon\,. & (11.26) \end{aligned}$$

Such perturbation is only defined for $0 \leq \varepsilon \leq 1$ and, for $\varepsilon = 1$ the extension of the PP coincide with the so-called *vacuous* set, including any possible PP specification over $\mathbf{X}_I$. We can similarly perturb a conditional PP $\phi(\mathbf{X}_I|\mathbf{X}_J)$, by applying the above procedure for each $\mathbf{x}_J \in \Omega_{\mathbf{X}_J}$.

Finally, let us introduce the notion of *critical level of perturbation* ε^*. We define ε^* as the maximum value of ϵ such that all the optimal policies of the corresponding IID (obtained according to Definition 49) return single decisions. The value of ε^* can be obtained with a bracketing over the parameter ε by running the IID evaluation algorithms described in Section 11.2 for different perturbation levels. Alternatively, we can also characterize the robustness of the model by computing the *failure level of perturbation* ε^{**}, which is intended as the minimum value of ϵ such that all the optimal policies of the corresponding IID are *vacuous*, i.e., all the decisions are returned. The perturbation can be simultaneously applied to all the potentials in the IID or restricted to a specific IPPs or IUPs. In the latter case it is possible to determine which one of the potentials of an ID has a higher impact on the MEU. This gives important information about the parameters deserving a more careful elicitation. A demonstrative example is reported here below.

Example 47 (sensitivity analysis of the oil wildcatter's problem) *Consider the ID in Example 11. To evaluate the corresponding IID obtained by perturbation of this model the VE algorithm with linear programs is considered. We perturb the PPs associated to S (Seismic) and to O (Oil), i.e. $\phi(S|O,T)$ and $\phi(O)$. Figure 11.2 depicts the size of the interval-valued* MEU *for increasing level of perturbation ϵ of these two potentials. The result is clear: perturbing $\phi(S|O,T)$ has a stronger effect than perturbing $\phi(O)$. Accordingly, we might conclude that the PP of S deserves a more careful quantification than that of O. Similar results are obtained by computing the critical perturbation levels ($\varepsilon^* = 0.0082$ for S and $\varepsilon^* = 0.0089$ for O) and the failure perturbation levels ($\varepsilon^{**} = 0.3749$ for S and $\varepsilon^{**} = 0.7499$ for O). Similar values and conclusions are obtained with the AR algorithm.*

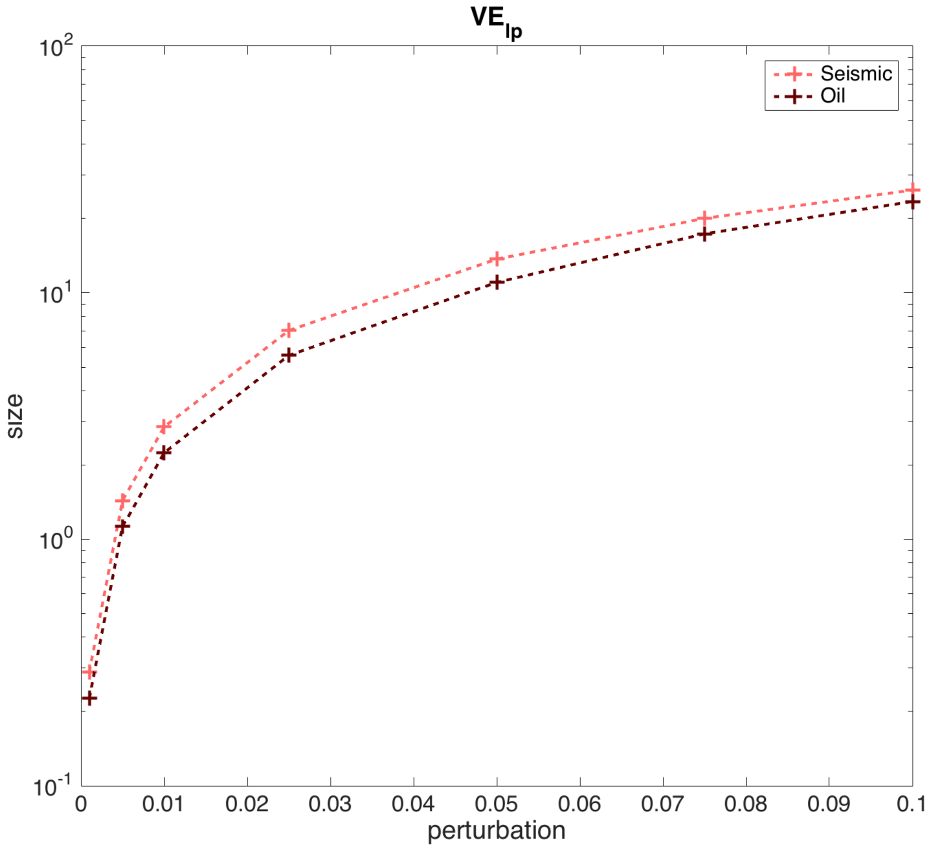


Figure 11.2: Size of the interval-valued MEU as a function of the perturbation level

11.6 Experimental work

For an empirical validation of the VE and AR algorithms extended for IIDs proposed in Section 11.2, we consider a benchmark of nine IDs modelling real decision tasks. Table 11.1 details the number of nodes of each type for these models. These IDs are transformed in IIDs by a perturbation of the original parameters (i.e., potentials) based on the procedure described in Section 11.5. Besides the three algorithms proposed in Section 11.2, we also consider the generalization of the AR to IIDs as proposed by Fertig and Breese [51]. We denote as $\mathrm{VE}_{\mathrm{lp}}$ our VE scheme based on linear programming, as $\mathrm{VE}_{\mathrm{outer}}$ the faster version proposed in Section 11.4.2, as $\mathrm{AR}_{\mathrm{lp}}$ our AR scheme (Section 11.4.3), and as $\mathrm{AR}_{\mathrm{fb}}$ the algorithm of Fertig and Breese.

Name of the ID	$\lvert\mathcal{U}_C\rvert$	$\lvert\mathcal{U}_D\rvert$	$\lvert\mathcal{U}_V\rvert$
Appendicitis	4	1	1
Chest Clinic	8	2	2
Comp. Assym.	3	5	1
Jaundice	21	2	1
NHL	17	3	1
Oil	2	2	2
Oil Split Costs	2	2	3
Thinkbox	5	2	4
Threat of Entry	3	9	1

Table 11.1: Number of chance, decision and utility nodes for the benchmark IIDs. More details about the corresponding precise models are given in Appendix B.2.

Figure 11.3 shows the running times and the relative durations when compared with those of the precise counterparts (VE or AR for IDs). As expected, the two simplest approaches ($\mathrm{AR}_{\mathrm{fb}}$ and $\mathrm{VE}_{\mathrm{outer}}$) roughly take the double of the time required by the precise evaluations (both upper and lower bounds are computed). Methods using linear programs ($\mathrm{AR}_{\mathrm{lp}}$ and $\mathrm{VE}_{\mathrm{lp}}$) are slower due to the time required by the linear solver. In particular, the evaluation might be demanding if there are chance variables with many states, this being the case of NHL (which has a chance variable with twelve states). If we compare the interval versions of AR against VE, we see that the differences are typically small for small ID/IIDs, while with large models such as NHL or Jaundice AR might be very slow. In fact the reversal of an arc might introduce very large potentials, this being a very well-known issue even with standard IDs.

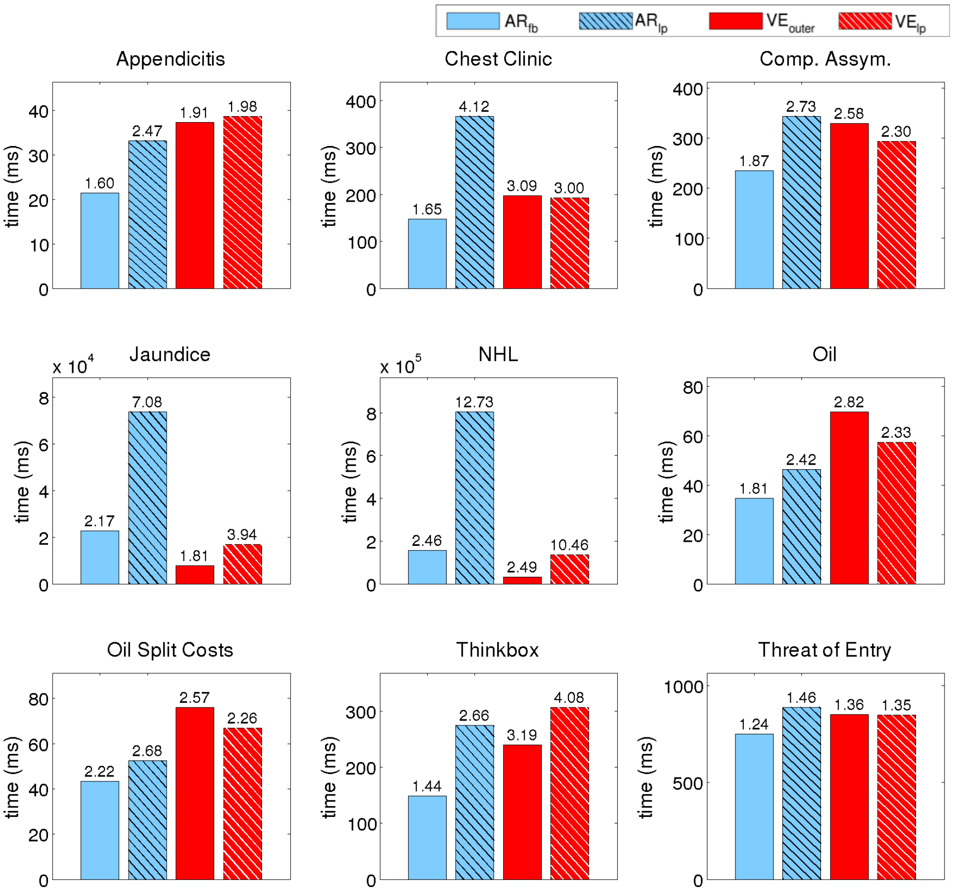


Figure 11.3: Absolute (y-axis) and relative (numbers over the bars) running times for the IIDs in Table 11.1.

We also analyse the size of the interval-valued MEU as a function of the size of the intervals in the initial potentials (parametrized by the perturbation level ϵ). The results obtained with precise utilities are depicted in Figure 11.4. As expected, the results based on the linear programming ($\mathrm{AR_{lp}}$ and $\mathrm{VE_{lp}}$) are the most informative ones for all the IIDs. $\mathrm{AR_{fb}}$ is much less accurate with *Comp. Assym.* and *Threat of Entry*. This might be due to the high number of decisions: a weakness of $\mathrm{AR_{fb}}$ is that the maximization of an IUP is done by taking the highest upper bounds and the lowest upper bounds (instead of the highest lower bounds). $\mathrm{VE}_{\mathrm{outer}}$ is also generally inaccurate and should be regarded as the algorithm of choice only if very severe constraints are posed on the running time.

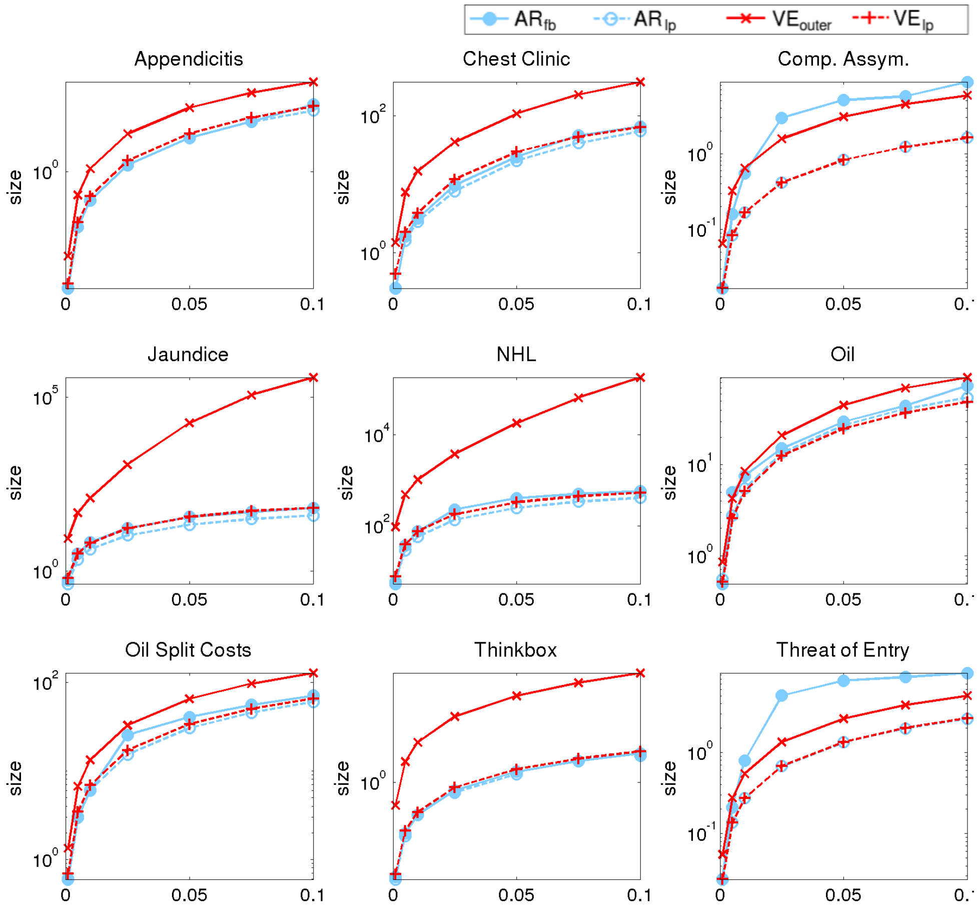


Figure 11.4: Size of interval-valued MEU as a function of the perturbation level ε of the IPPs.

Similar results, with a sharp specification of the IPPs and intervals only in the IUPs are reported in Figure 11.5. The two VE methods, which differs only in the treatment of the IPPs, produce the same results. In general, when only utilities are imprecise, $\mathrm{AR}_{\mathrm{lp}}$ offers the best results. Finally, by comparing the y-scales in Figures 11.4 and 11.5, it can be observed that the imprecision in the IPPs seems to have a stronger effect than that in the IUPs on the size of the interval MEU.

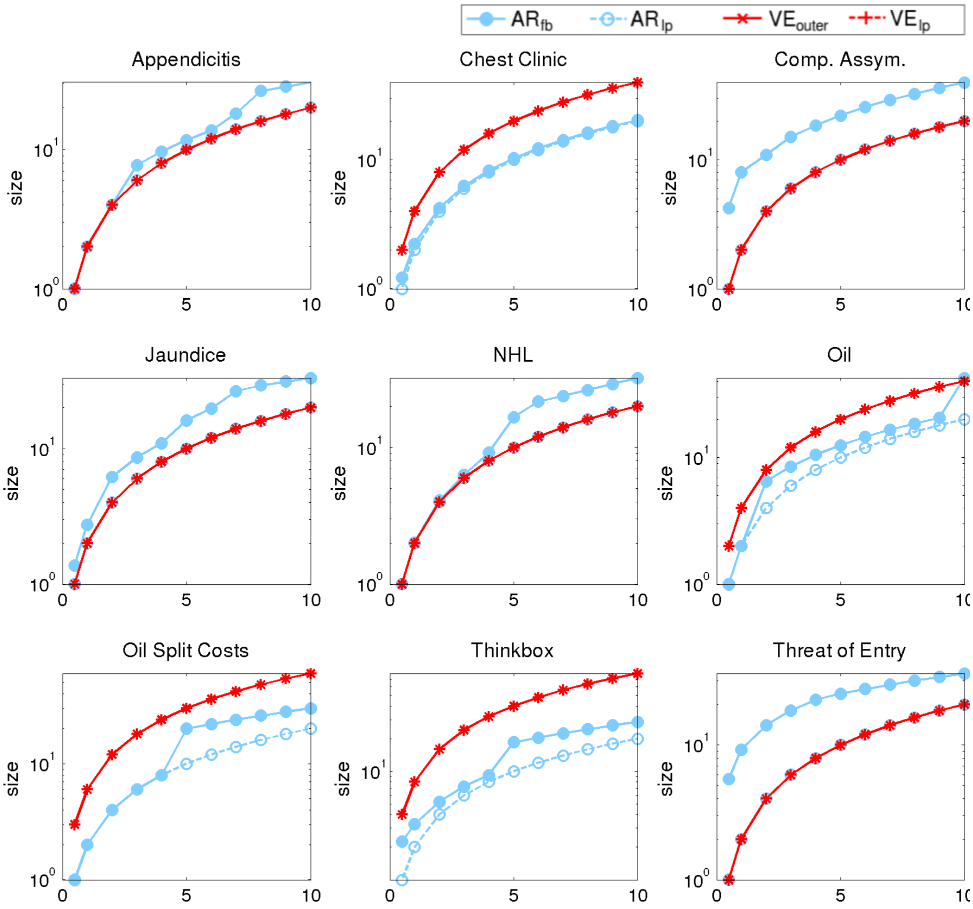


Figure 11.5: Size of interval-valued MEU as a function of the perturbation level ε of the IUPs.

11.7 Conclusions

We have extended the ID formalism to support an interval-valued specification of the potentials. The corresponding models, called IIDs, have a direct sensitivity-analysis interpretation: an IID is equivalent to a collection of precise IDs whose potentials are consistent with the constraints induced by the intervals. Consequently, the set-valued (so-called *credal*) optimal policies of an IID include the single-valued optimal policies of the consistent IDs, as well as the interval-valued MEU of an IID contains all the MEU of the consistent IDs. Moreover, we extended to IIDs the classical *variable elimination* and *arc reversal* evaluation schemes for IDs. These two extensions are achieved by local optimization tasks, reduced to linear programs. For VE, a faster but less accurate procedure, that does not require linear programming, is also proposed. The latter approach introduces an additional outer approximation similar to that characterizing the generalization of the AR algorithm proposed by Fertig and Breese [51, 50]. All these algorithms have the same asymptotic complexities of their classical, sharp-valued, counterparts. The empirical analysis showed that the approximations we introduce to keep the same complexity as with IDs did not compromise the informativeness of the inferences. In particular, the new methods based on linear programming are clearly more accurate than the algorithm of Fertig and Breese. Finally, we also proposed a possible application of IIDs to practical sensitivity analysis in precise IDs. Computing the maximal level of perturbation, no matter whether local or global, might allow to decide which are the potential/variables deserving a more careful elicitation process.

As a future work we intend to extend this formalism to more general imprecise frameworks, e.g., credal sets represented by extreme points or generic linear constraints. This should affect to the computational complexity of the evaluation process, thus making necessary the development of specific approximate algorithms. We also intend to extensively test the procedure we proposed for sensitivity analysis in practical IDs and compare it against the methods proposed so far with the same goal. Additionally, we could also consider using BTs for representing imprecise potentials. In doing so, we will be able to prune the BTs in order to obtain approximate solutions.

Chapter 12

Efficient Evaluation with Tables

12.1 Introduction

In Chapters 8 and 10, we showed how the potentials in an ID can be represented as BTs in order to improve the efficiency of the evaluation. In the present chapter, by contrast, we explore different alternatives for optimizing the evaluation assuming that the potentials of the ID are represented as tables. We base on the following idea: the evaluation algorithms reviewed in Section 4.5 (i.e. VE, AR and LE) require performing several combinations and marginalizations on the potentials attached to the ID. Finding an optimal order for these operations is a NP-hard problem [5] and it is an element of crucial importance for the efficiency of the evaluation. The evaluation of an ID can be considered as a combinatorial optimization problem, that is the problem of finding an optimal order in which combinations and marginalizations are performed. This idea was already used to make inference in BNs with the first version of the *Symbolic Probabilistic Inference algorithm (SPI)* [104] and with an improved algorithm in the SPI family called *set-factoring* [74]. In a related work [39] some experiments with SPI were performed to evaluate decision networks, however no details of the algorithm were provided.

In this dissertation, different approaches for optimizing the order of the operations involved in the evaluation of IDs are considered. First, we adapt the SPI algorithm for evaluating IDs taking into account the differences of an ID compared to a BN: two kinds of potentials, the temporal order of decisions, etc. Secondly, an optimization of the VE algorithm based on [106] is also proposed. This optimization consists of using a greedy algorithm for minimizing the cost of the combination of all the potentials involved in the removal of a variable. This optimization can be seen as an extension of the *binary join trees* of P.P. Shenoy [106] to IDs. Both algorithms are described for the direct evaluation of IDs (without using any auxiliary structure) and for the computation of clique-to-clique messages in Lazy Evaluation (LE) of IDs. In the experimental work, we analyse the behaviour of all these algorithms using a set of IDs from the literature. It is demonstrated that the proposed algorithms can improve the efficiency of the evaluation. Moreover, SPI outperforms VE in many instances. We also propose a pre-analysis algorithm based on the number of arithmetic operations that could help to predict which of the algorithms is the most appropriate one for evaluating each ID.

This chapter is organized as follows: in Section 12.2 the motivation of this work is explained; the SPI algorithm for the direct evaluation of IDs is explained in Section 12.3 while the use of this algorithm for computing clique-to-clique messages in LE is described in Section 12.5; the description of the optimization of VE is given in Section 12.6; Section 12.7 includes the experimental work and results. Finally, Section 12.8 details our conclusions and lines for future work.

12.2 Motivation

In order to explain the motivation, let us consider the ID shown in Figure 12.1. The optimal policy for D_1 can be calculated directly from Equation (4.24) as stated in Equation (12.1).

$$\widehat{\delta}_{D_1}(A) = \arg\max_{D_1} \sum_{G,F,E,C,B} \phi(G)\phi(F)\phi(E)\phi(C)\phi(B|C,E,F,G)\phi(A|B)\left(\psi_1(G,F,D_1)+\psi_2(E,C,D_1)\right) \tag{12.1}$$

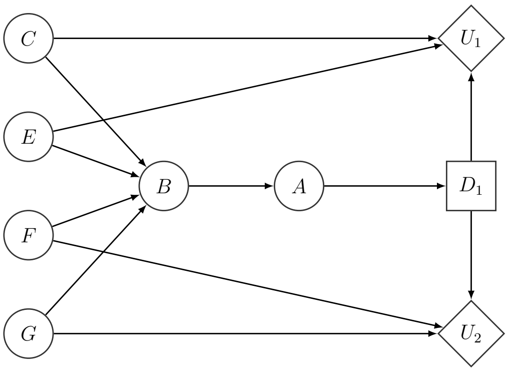


Figure 12.1: An example of an ID whose partial order is the following: $\{A\} \prec D_1 \prec \{B, C, E, F, G\}$.

The table representing the joint probability of all the chance variables might be too large. For that reason, some evaluation algorithms such as VE for IDs iteratively removes variables following a strong elimination order (see Section 4.5.1). The advantage of VE is that the removal of a variable X_i only involves computations with those potentials with X_i in their domain. Assuming that all the variables are binary, the optimal order for removing variables in $\mathcal{I}_1$ is G, F, E, C, B and the computations done are:

1. Removal of G (96 multiplications, 48 additions, 32 divisions):

$$\sum_G \phi(G) \cdot \phi(B|C, E, F, G) = \sum_G \phi(G, B|C, E, F) = \phi(B|C, E, F)$$

$$\frac{\sum_G \phi(G,B|C,E,F) \cdot \psi_1(G,F,D_1)}{\phi(B|C,E,F)} = \psi(D_1, F, B, C, E)$$

2. Removal of F (48 multiplications, 24 additions, 16 divisions):

$$\sum_F \phi(F) \cdot \phi(B|C, E, F) = \sum_G \phi(F, B|C, E) = \phi(B|C, E)$$

$$\frac{\sum_F \phi(F,B|C,E) \cdot \psi(D_1,F,B,C,E)}{\phi(F,B|C,E)} = \psi(D_1, B, C, E)$$

3. Removal of E (24 multiplications, 28 additions, 8 divisions):

$$\sum_E \phi(E) \cdot \phi(B|C, E) = \sum_E \phi(E, B|C) = \phi(B|C)$$

$$\frac{\sum_E \phi(E,B|C) \cdot (\psi(D_1,B,C,E) + \psi_2(E,C,D_1))}{\phi(B|C)} = \psi(D_1, B, C)$$

4. Removal of C (12 multiplications, 15 additions, 4 divisions):

$$\sum_C \phi(C) \cdot \phi(B|C) = \sum_C \phi(B,C) = \phi(B)$$

$$\frac{\sum_C \phi(B,C) \cdot \psi(D_1,B,C)}{\phi(C)} = \psi(D_1,B)$$

5. Removal of B (12 multiplications, 15 additions, 4 divisions):

$$\sum_B \phi(B) \cdot \phi(A|B) = \sum_B \phi(A,B) = \phi(A)$$

$$\frac{\sum_B \phi(A,B)\psi(D_1,B)}{\phi(A)} = \psi(D_1,A)$$

We can see that the removal of the variables in $\mathcal{I}_1$ using VE requires 366 arithmetic operations (192 multiplications, 110 additions and 64 divisions). Independently of the elimination order used to solve this ID, VE will always have to combine the marginal potentials with a large potential. However, with a re-order of the operations, this situation can be avoided:

1. Removal of F, G (112 multiplications, 40 additions, 16 divisions):

$$\sum_{F,G} \phi(B|C,E,F,G) \cdot (\phi(F) \cdot \phi(G)) = \sum_{F,G} \phi(B,F,G|C,E) = \phi(B|C,E)$$

$$\frac{\sum_{F,G} \phi(B,F,G|C,E) \cdot \psi_1(G,F,D_1)}{\phi(B|C,E)} = \psi(B,C,E,D_1)$$

2. Removal of B, C, E (68 multiplications, 42 additions, 4 divisions):

$$\sum_{B,C,E} (\phi(A|B) \cdot (\phi(E) \cdot \phi(C))) \cdot \phi(B|C,E) \ = \sum_{B,C,E} \phi(A,E,C,B) = \phi(A)$$

$$\frac{\sum_{B,C,E} \phi(A,E,C,B)(\psi(B,C,E,D_1) + \psi_2(E,C,D_1))}{\phi(A)} = \psi(D_1,A)$$

Now the removal of $\mathcal{I}_1$ requires 282 arithmetic operations (180 multiplications, 82 additions and 20 divisions). From this example we can deduce that sometimes it could be better to combine small potentials even if they do not share any variable (e.g., $\phi(E)$ and $\phi(C)$). This combination will never be performed using VE since it is guided by the elimination order. Thus, the efficiency of the evaluation can be improved if an optimization of the order of both operations, marginalization and combination, is performed simultaneously[74].

12.2.1 Definition of the problem

Evaluating an ID can be seen as an optimization problem in which we try to find an order that minimizes the cost of the operations involved in the evaluation. Moreover, due to temporal restrictions, the problem can be divided into two sub-problems. The first one consists of finding the optimal order of the operations involved in the removal of a set of chance variables from a set of PPs and UPs. Similarly, the second sub-problem consists of finding the optimal order of all the operations involved in the removal of a decision from a set of UPs.

For the first problem, assume that we shall remove the set of chance variables $\mathbf{X}$ from the sets of PPs and UPs $\Phi_{\mathbf{X}}$ and $\Psi_{\mathbf{X}}$ (potentials with any variable of $\mathbf{X}$ in the domain). That is, we shall calculate Equation (12.2).

$$\sum_{\mathbf{X}} \prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}}} \phi_{\mathbf{X}} \left(\sum_{\psi_{\mathbf{X}} \in \Psi_{\mathbf{X}}} \psi_{\mathbf{X}} \right) = \sum_{\mathbf{X}} \prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}}} \phi_{\mathbf{X}} \frac{\sum_{\mathbf{X}} \left(\prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}}} \phi_{\mathbf{X}} \left(\sum_{\psi_{\mathbf{X}} \in \Psi_{\mathbf{X}}} \psi_{\mathbf{X}} \right) \right)}{\sum_{\mathbf{X}} \left(\prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}}} \phi_{\mathbf{X}} \right)} \tag{12.2}$$

Previous expression is a factorization of potentials, thus we should calculate the set of potentials $\Phi'_{\mathbf{X}}$ and $\Psi'_{\mathbf{X}}$ such that:

$$\prod_{\phi' \in \Phi'_{\mathbf{X}}} \phi' = \sum_{\mathbf{X}} \prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}}} \phi_{\mathbf{X}} \qquad \sum_{\psi' \in \Psi'_{\mathbf{X}}} \psi' = \frac{\sum_{\mathbf{X}} \left(\prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}}} \phi_{\mathbf{X}} \left(\sum_{\psi_{\mathbf{X}} \in \Psi_{\mathbf{X}}} \psi_{\mathbf{X}} \right) \right)}{\sum_{\mathbf{X}} \left(\prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}}} \phi_{\mathbf{X}} \right)} \tag{12.3}$$

To compute previous expression we should find an optimal order for the operations of sum-marginalization, multiplication, addition and division. Let $Y \in \mathbf{X}$ be a chance variable, let Φ_Y and Ψ_Y be the set of PPs and UPs containing Y in the domain, $\Phi^* = \Phi_{\mathbf{X}} \backslash \Phi_Y$ and $\Psi^* = \Psi_{\mathbf{X}} \backslash \Psi_Y$. Applying the distributive law, the removal of Y can be performed using Equation (12.4).

$$\sum_{\mathbf{X}\setminus Y}\left(\prod_{\phi^*\in\Phi^*}\phi^*\left(\sum_{Y}\prod_{\phi_Y\in\Phi_Y}\phi_Y\right)\left(\sum_{\psi^*\in\Psi^*}\psi^*+\frac{\sum_Y\left(\prod_{\phi_Y\in\Phi_Y}\phi_Y\left(\sum_{\psi_Y\in\Psi_Y}\psi_Y\right)\right)}{\sum_Y\left(\prod_{\phi_Y\in\Phi_Y}\phi_Y\right)}\right)\right) \tag{12.4}$$

From Equation (12.4) we get that a variable Y can only be removed if the product of all the potentials in Φ_Y has been calculated. Moreover, the removal must be performed at the same time from the UPs and PPs.

For the second sub-problem, assume that we aim to remove a decision variable D from the set of UPs Ψ_D (potentials with D in the domain). Then we should calculate Equation (12.5). In this case, the decision variable is removed using max-marginalization.

$$\psi'_D = \max_D \sum_{\psi_D\in\Psi_D} \psi_D \tag{12.5}$$

Note that, when removing a decision, usually there are not PPs containing it and if any, the decision is not affecting the values of such PP since any decision is d-separated from its predecessors (see Proposition 5 in page 71) and any successor has already been removed.

In both sub-problems, we try to find the optimal order for all the operations involved. This optimization will reduce the size of intermediate potentials and therefore the evaluation should be more efficient.

12.3 Symbolic probabilistic inference for IDs

12.3.1 Overview

The *Symbolic Probabilistic Inference* algorithm (SPI) was already used for making inference in BNs [104, 74]. This is also a greedy algorithm that considers the removal of a set of variables as a combinatorial factorization problem. That is, SPI tries to find the optimal order for the combinations and marginalizations by choosing at each step the best operation. Herein we describe how the SPI algorithm can be used for directly evaluating IDs. The correctness and complexity analysis is shown in Section 12.4. For evaluating IDs, as VE does, SPI removes all the variables in reverse order of the partial ordering imposed by the information constraints (called a strong elimination order [68]). That is, it first sum-marginalizes $\mathcal{I}_n$, then max-marginalizes D_n, sum-marginalizes $\mathcal{I}_{k-1}$, etc. The general scheme of the SPI algorithm as presented in this dissertation is shown in Algorithm 20.

Algorithm 20 SPI-algorithm

input : Φ, Ψ (sets of potentials in the ID), $\{\mathcal{I}_0, D_1, \mathcal{I}_1, \ldots, D_n, \mathcal{I}_n\}$ (partitions of nodes in the ID)

```
for $k \leftarrow n$ to $0$ do
    $(\Phi_{\mathbf{X}}, \Psi_{\mathbf{X}}) \leftarrow (\{\phi \in \Phi | dom(\phi) \cap \mathcal{I}_k \neq \emptyset\},\ \{\psi \in \Psi | dom(\psi) \cap \mathcal{I}_k \neq \emptyset\})$
    $(\Phi, \Psi) \leftarrow (\Phi \backslash \Phi_{\mathbf{X}},\ \Psi \backslash \Psi_{\mathbf{X}})$
    $(\Phi'_{\mathbf{X}},\ \Psi'_{\mathbf{X}}) \leftarrow RemoveChanceSet(\mathcal{I}_k, \Phi_{\mathbf{X}}, \Psi_{\mathbf{X}})$    ▷ Algorithm 24
    $(\Phi, \Psi) \leftarrow (\Phi \cup \Phi'_{\mathbf{X}},\ \Psi \cup \Psi'_{\mathbf{X}})$
    if $k > 0$ then
        $(\Phi_D, \Psi_D) \leftarrow (\{\phi \in \Phi | D_k \in dom(\phi)\},\ \{\psi \in \Psi | D_k \in dom(\psi)\})$
        $(\Phi, \Psi) \leftarrow (\Phi \backslash \Phi_D,\ \Psi \backslash \Psi_D)$
        $(\Phi'_D,\ \psi'_D) \leftarrow RemoveDecision(D_k, \Phi_D, \Psi_D)$    ▷ Algorithm 26
        $(\Phi, \Psi) \leftarrow (\Phi \cup \Phi'_D,\ \Psi \cup \{\psi'_D\})$
    end if
end for
```

SPI and VE algorithms differ in the way they solve these problems: VE chooses at each step a variable to remove while SPI chooses a pair of potentials to combine and eliminate variables when possible. In this sense SPI is more fine-grained than VE. The latter only considers the next variable to eliminate, and not the order in which potentials are combined.

12.3.2 Combination candidate set

The SPI algorithm uses a data structure for storing the candidate pairs of potentials to combine in the next iteration. Herein we introduce this data structure and detail the related algorithms used in posterior sections.

Given a set of potentials $\Phi_{\mathbf{X}}$, we define the *combination candidate set* B as the set of all possible pairs of potentials in $\Phi_{\mathbf{X}}$. That is $B := \{\{\phi_i, \phi_j\} | \phi_i, \phi_j \in \Phi_{\mathbf{X}}, \phi_i \neq \phi_j\}$. Algorithm 21 takes two input arguments: a set of potentials $\Phi_{\mathbf{X}}$ and an existing combination candidate set B. If B is empty, this algorithm returns a set B with all pairwise combinations of elements of $\Phi_{\mathbf{X}}$. Otherwise, the algorithm updates B by adding any missing pair with elements from $\Phi_{\mathbf{X}}$ (without removing those pairs already in B). Notice that, if potentials are represented as tables, $\{\phi_i, \phi_j\}$ is equivalent to $\{\phi_j, \phi_i\}$. Thus, the pair $\{\phi_j, \phi_i\}$ is not added to B if $\{\phi_i, \phi_j\}$ is already present (line 4). For example, given a set of potentials $\Phi_{\mathbf{X}} = \{\phi_1, \phi_2, \phi_3\}$, the combination candidate set is $\{\{\phi_1, \phi_2\}, \{\phi_1, \phi_3\}, \{\phi_2, \phi_3\}\}$.

Algorithm 21 addPairwiseCombinations

input : $\Phi_{\mathbf{X}} = \{\phi_1, \phi_2, \ldots \phi_m\}$ (set of m potentials), B (existing combination candidate set)

output : B (updated combination candidate set)

```
1: for $i \leftarrow 1$ to $m - 1$ do
2:     for $j \leftarrow i + 1$ to $m$ do
3:         $p \leftarrow \{\phi_i, \phi_j\}$
4:         if $p \notin B$ then
5:             $B \leftarrow B \cup \{p\}$
6:         end if
7:     end for
8: end for
9: return $B$
```

A pair is a set of two potentials, thus any operation with sets can be used with pairs. For example, given the pairs $p = \{\phi_1, \phi_2\}, p' = \{\phi_1, \phi_3\}$ and $p'' = \{\phi_3, \phi_4\}$, the intersection $p \cap p'$ is $\{\phi_1\}$ while the intersection $p \cap p''$ is $\emptyset$. Thus, given a combination candidate set B and a pair $p \in B$, the removal of any pair $p' \in B$ containing at least one potential in common with p is denoted as $\{p' \in B | p' \cap p = \emptyset\}$. Similarly, the removal of both potentials in p from a potential set $\Phi_{\mathbf{X}}$ is denoted as $\Phi_{\mathbf{X}} \backslash p$.

Once the pairwise combination candidate set is built, a pair of potentials should be selected to combine. Algorithm 22 shows the procedure for selecting a pair from B minimizing any score or heuristic. Some examples of these heuristics are later explained in Section 12.3.5.

Algorithm 22 selectBest

input : $B = \{p_1, \ldots, p_l\}$ (combination candidate set with l pairs)

output : $bestPair$ (the best pair in B minimizing any score)

```
$minScore \leftarrow +\infty$
for $i \leftarrow 1$ to $l$ do
    $s \leftarrow score(p_i)$
    if $s < minScore$ then
        $bestPair \leftarrow p_i$
        $minScore \leftarrow s$
    end if
end for
return $bestPair$
```

In the original proposal of SPI algorithm for BNs, the combination candidate set B does not contain singletons. That is B is composed only of pairs of potentials. In our approach, the set B can also contain singleton potentials containing any variable that can be directly removed. That is a variable which is only present in one potential. With this improvement, the algorithm is not forced to combine at least two potentials in order to remove the first variable. On the other hand, these variables are not directly removed because it cannot be assured that the cost of this removal is lower that the cost of selecting a pair of potentials.

For example, let $\mathbf{X} = \{A, B, C\}$, be a set of chance variables that we want to remove from the set of probability potentials $\Phi_{\mathbf{X}} = \{\phi(A|B), \phi(B|C, E), \phi(C)\}$, then the candidate combination set generated is:

$$B = \left\{ \{\phi(A|B), \phi(B|C,E)\},\ \{\phi(A|B), \phi(C)\},\ \{\phi(B|C,E), \phi(C)\},\ \{\phi(A|B)\} \right\}$$

Previous set B contains the singleton $\{\phi(A|B)\}$ because variable A only belongs to the domain of this potential and thereby it can be sum-marginalized without the need of performing first a combination. Although variable E only belongs to $\phi(B|C,E)$, this potential is not added as a singleton because E is not contained in the set $\mathbf{X}$ of variables we aim to remove. The procedure for adding the singletons to an existing combination candidate set B, given a set of variables $\mathbf{X}$ to remove from $\Phi_{\mathbf{X}}$ is shown in Algorithm 23.

Algorithm 23 addSingletons

input : $\Phi_{\mathbf{X}} = \{\phi_1, \phi_2, \dots \phi_{|\Phi|}\}$ (set of potentials), $\mathbf{X}$ (set of variables to remove), B (existing combination candidate set)

output : B (updated combination candidate set)

```
for $i \leftarrow 1$ to $|\Phi_{\mathbf{X}}|$ do
    if $\exists Y \in dom(\phi_i) \cap \mathbf{X} | \forall \phi \in \Phi_{\mathbf{X}} \backslash \{\phi_i\} : X \notin dom(\phi)$ then
        $B \leftarrow B \cup \{\{\phi_i\}\}$                    ▷ $\{\phi_i\}$ is a singleton
    end if
end for
return $B$
```

12.3.3 Removal of chance variables

In order to remove a subset of chance variables $\mathbf{X}$ from $\Phi_{\mathbf{X}}$ and $\Psi_{\mathbf{X}}$, SPI considers PPs and UPs separately: first, SPI tries to find the best order for combining all potentials in $\Phi_{\mathbf{X}}$ as shown in Algorithm 24. For that purpose, all possible pairwise combinations between the PPs are stored in the combination candidate set B (line 3). Besides, B also contains those PPs or singletons (line 4) that contain any variable of $\mathbf{X}$ which is not present in any other potential of $\Phi_{\mathbf{X}}$.

Algorithm 24 RemoveChanceSet

input : $\mathbf{X}$ (subset of chance variables), $\Phi_{\mathbf{X}}$ (set of PPs relevant for removing $\mathbf{X}$), $\Psi_{\mathbf{X}}$ (set of UPs relevant for removing $\mathbf{X}$)

output : Sets of potentials $\Phi'_{\mathbf{X}}$ and $\Psi'_{\mathbf{X}}$ resulting from removing $\mathbf{X}$ (Eq. (12.3)))

```
 1: $B \leftarrow \emptyset$                                        ▷ Empty combination candidate set
 2: repeat
 3:     $B \leftarrow addPairwiseCombinations(\Phi_{\mathbf{X}}, B)$
 4:     $B \leftarrow addSingletons(\Phi_{\mathbf{X}}, \mathbf{X}, B)$
 5:     $p \leftarrow selectBest(B)$
 6:     if $p$ is a pair then
 7:         $\phi_{ij} \leftarrow \phi_i \cdot \phi_j$                          ▷ $p$ is a pair $\{\phi_i, \phi_j\}$
 8:     else
 9:         $\phi_{ij} \leftarrow \phi_i$                                ▷ $p$ is a singleton $\{\phi_i\}$
10:     end if
11:     $\mathbf{W} \leftarrow \{W \in dom(\phi_{ij}) \cap \mathbf{X} | \forall \phi \in \Phi_{\mathbf{X}} \backslash p : W \notin dom(\phi)\}$
12:     $\Psi_{\mathbf{W}} \leftarrow \{\psi \in \Psi_{\mathbf{X}} | \mathbf{W} \cap dom(\psi) \neq \emptyset\}$
13:     if $\mathbf{W} \neq \emptyset$ then
14:         $(\phi'_{ij}, \Psi'_{\mathbf{W}}) \leftarrow Sum\text{-}marginalize(\mathbf{W}, \phi_{ij}, \Psi_{\mathbf{W}})$
15:     else
16:         $(\phi'_{ij}, \Psi'_{\mathbf{W}}) \leftarrow (\phi_{ij}, \Psi_{\mathbf{W}})$
17:     end if
18:     $B \leftarrow \{p' \in B | p' \cap p = \emptyset\}$                        ▷ Update
19:     $\mathbf{X} \leftarrow \mathbf{X} \backslash \mathbf{W}$
20:     $\Phi_{\mathbf{X}} \leftarrow (\Phi_{\mathbf{X}} \backslash p) \cup \{\phi'_{ij}\}$
21:     $\Psi_{\mathbf{X}} \leftarrow (\Psi_{\mathbf{X}} \backslash \Psi_{\mathbf{W}}) \cup \Psi'_{\mathbf{W}}$
22: until $\mathbf{X} = \emptyset$
23: $(\Phi'_{\mathbf{X}}, \Psi'_{\mathbf{X}}) \leftarrow (\Phi_{\mathbf{X}}, \Psi_{\mathbf{X}})$
24: return $(\Phi_{\mathbf{X}}, \Psi_{\mathbf{X}})$
```

At each iteration, an element of B is selected (line 5). If it is a pair (line 7), both potentials are combined. The procedure stops when all variables in $\mathbf{X}$ have been removed. A variable can be removed at the moment it only appears in a single PP. Thus, after each combination it is computed the set of variables $\mathbf{W} \subseteq \mathbf{X}$ satisfying such condition (line 11) and the set of UPs $\Psi_{\mathbf{W}}$ containing any variable in $\mathbf{W}$ (line 12). If there is any variable that can be removed, a similar procedure is performed for combining utilities and sum-marginalizing the variables in $\mathbf{W}$ (line 14). At the end of each iteration sets $B, \mathbf{X}, \Phi_{\mathbf{X}}$ and $\Psi_{\mathbf{X}}$ are updated. Notice that this algorithm produces a factorization of potentials as stated in Equation (12.3).

Example 48 *To illustrate Algorithm 24, let us suppose we aim to remove the set of variables* $\mathbf{X} = \{A, C\}$ *from the sets of potentials* $\Phi_{\mathbf{X}} = \{\phi(A), \phi(C), \phi(B|A, C)\}$ *and* $\Psi_{\mathbf{X}} = \{\psi(A), \psi(A, B, C)\}$*. Initially, the combination candidate set B is* $\{\{\phi(A), \phi(C)\}, \{\phi(A), \phi(B|A, C)\}, \{\phi(C), \phi(B|A, C)\}\}$*. Suppose that our heuristic chooses the pair* $\{\{\phi(A), \phi(C)\}$ *in line 5, then we combine both potentials obtaining* $\phi(A, C)$ *as a result in line 7. Notice that these two potentials are never combined by the VE algorithm since they do not share any variable. The set of removable variables* $\mathbf{W}$ *is equal to* $\emptyset$ *and the sum-marginalize algorithm is not invoked in line 14. At the end of this iteration, set* B *is now empty and* $\Phi_{\mathbf{X}}$ *is* $\{\phi(A, C), \phi(B|A, C)\}$*. The sets* $\mathbf{X}$ *and* $\Psi_{\mathbf{X}}$ *have not changed. In the second iteration the single pair in* $B = \{\{\phi(A, C), \phi(B|A, C)\}\}$ *are combined, obtaining* $\phi(A, B, C)$ *as a result. Now* $\mathbf{W}$ *is equal to* $\{A, C\}$ *and the sum-marginalization algorithm is invoked in line 14.*

In Algorithm 24 only PPs are combined while UPs are not. The UPs must be combined with ϕ_{ij} which is the resulting potential of combining all potentials containing X. Thus, in order to avoid additional computations, the utilities are only combined when a variable can be removed. That is the moment when ϕ_{ij} has been calculated. The procedure for sum-marginalizing a set of variables (Algorithm 25) involves finding a good order for summing the UPs. The procedure for that is quite similar to the one for combining probabilities, the main difference is that at the moment a variable can be removed, the PPs and UPs resulting from the marginalization are computed (line 16). Notice that this procedure is invoked on $\Psi_{\mathbf{W}} \subseteq \Psi_{\mathbf{X}}$.

Algorithm 25 Sum-marginalize

input : $\mathbf{W}$ (subset of chance variables), ϕ (probability potential relevant for removing $\mathbf{W}$), $\Psi_{\mathbf{W}}$ (set of utility potentials relevant for removing $\mathbf{W}$)

output : Potential ϕ' and set of potential $\Psi'_{\mathbf{W}}$ resulting from removing $\mathbf{W}$

```
$B' \leftarrow \emptyset$                                              ▷ Empty combination candidate set
if $\Psi_{\mathbf{W}} = \emptyset$ then
    return $(\sum_{\mathbf{W}} \phi,\ \emptyset)$
end if
repeat
    $B' \leftarrow addPairwiseCombinations(\Psi_{\mathbf{W}}, B')$
    $B' \leftarrow addSingletons(\Psi_{\mathbf{W}}, \mathbf{W}, B')$
    $q \leftarrow selectBest(B')$
    if $q$ is a pair then
        $\psi_{ij} \leftarrow \psi_i + \psi_j$                            ▷ $q$ is a pair $\{\psi_i, \psi_j\}$
    else
        $\psi_{ij} \leftarrow \psi_i$                                   ▷ $q$ is a singleton $\{\psi_i\}$
    end if
    $\mathbf{V} \leftarrow \{V \in dom(\psi_{ij}) \cap \mathbf{W} | \forall \psi \in \Psi_{\mathbf{W}} \backslash q : V \notin dom(\psi)\}$
    if $\mathbf{V} \neq \emptyset$ then
        $(\phi'_{\mathbf{V}}, \psi'_{\mathbf{V}}) \leftarrow (\sum_{\mathbf{V}} \phi,\ \frac{\sum_{\mathbf{V}}(\phi \otimes \psi_{ij})}{\phi'_{\mathbf{V}}})$
    else
        $(\phi'_{\mathbf{V}}, \psi'_{\mathbf{V}}) \leftarrow (\phi, \psi_{ij})$
    end if
    $B' \leftarrow \{q' \in B' | q' \cap q = \emptyset\}$                         ▷ Update
    $\mathbf{W} \leftarrow \mathbf{W} \backslash \mathbf{V}$
    $\phi = \phi'_{\mathbf{V}}$
    $\Psi_{\mathbf{W}} \leftarrow (\Psi_{\mathbf{W}} \backslash q) \cup \{\psi'_{\mathbf{V}}\}$
until $\mathbf{W} = \emptyset$
$(\phi',\ \Psi'_{\mathbf{W}}) \leftarrow (\phi,\ \Psi_{\mathbf{W}})$
return $(\phi,\ \Psi_{\mathbf{W}})$
```

Example 49 *Suppose Algorithm 25 is invoked to remove* $\mathbf{W} = \{A, C\}$ *from* $\phi = \phi(A, B, C)$ *and* $\Psi_{\mathbf{W}} = \{\psi(A), \psi(A, B, C)\}$*. Initially, B' is equal* *to* $\{\{\psi(A), \psi(A, B, C)\}, \{\psi(A, B, C)\}\}$*. Notice that now the combination candidate set contains the singleton* $\psi(A, B, C)$ *because variable* C *is in only one potential. Suppose that in the first iteration we select this singleton, then* $\mathbf{V}$ *is equal to* $\{C\}$*. Thus in line 16 variable* C *is sum-marginalize out obtaining as a result the potentials* $\phi(A, B)$ *and* $\psi(A, B)$*. Now the sets are updated as* $\mathbf{W} = \{A\}$ *and* $\Psi_{\mathbf{W}} = \{\psi(A), \psi(A, B)\}$*. In the second iteration,* B' *is* $\{\psi(A), \psi(A, B)\}$ *so the UPs in the single pair are added and the sum-marginalization of* A *is performed. Finally the output of the algorithm is the PP* $\phi(B)$ *and set of UPs* $\{\psi(B)\}$*.*

12.3.4 Removal of a decision

The removal of a decision variable D from a set of PPs Φ_D and from a set of UPs Ψ_D is shown in Algorithm 26. This procedure does not imply the combination of any PP since any decision is d-separated from its predecessors (see Proposition 5, page 71) and any successor has already been removed (the removal order of the disjoint subsets of variables must respect the temporal constraints). Thus, any PP $\phi(D_k, \mathbf{X})$ must be directly transformed into $\phi(\mathbf{X})$ if D_k is a decision and $\mathbf{X}$ is a set of chance variables that belong to $\mathcal{I}_i$ with $i < k$. In practice, this means that each PP $\phi \in \Phi_D$ is restricted to any of the values $d \in \Omega_D$ (line 3). This restriction is denoted as $\phi^{R(D=d)}$.

Algorithm 26 RemoveDecision

input : D (decision variable), Φ_D (set of PPs relevant for removing D), Ψ_D (set of UPs relevant for removing D)

output : Φ'_D (set of PPs resulting from removing D), ψ'_D (UP resulting from removing D)

```
1: $\Phi'_D \leftarrow \emptyset$
2: for each $\phi \in \Phi_D$ do
3:     $\Phi'_D \leftarrow \Phi'_D \cup \{\phi^{R(D=d)}\}$
4: end for
5: $\psi'_D \leftarrow max\text{-}marginalize(D, \Psi_D)$
6: return $(\Phi'_D, \psi'_D)$
```

Algorithm 27 shows the procedure for finding the best order for summing all UPs containing a decision D. Notice that the pairwise candidate set does not contain singletons and the sum-marginalization is performed once all UPs have been summed.

Algorithm 27 max-marginalize

```
input :   $D$ (decision variable), $\Psi_D$ (set of UPs relevant for removing $D$)
output :   $\psi'_D$ (UPs resulting from removing $D$)
 1: $B' \leftarrow \emptyset$
 2: while $|\Psi_D| > 1$ do
 3:     $B \leftarrow addPairwiseCombinations(\Psi_D, B')$
 4:     $q \leftarrow selectBest(B')$
 5:     $\psi_{ij} \leftarrow \psi_i + \psi_j$
 6:     $B' \leftarrow \{q' \in B' | q' \cap q = \emptyset\}$                ▷ Update
 7:     $\Psi_D \leftarrow (\Psi_D \backslash q) \cup \{\psi_{ij}\}$
 8: end while
 9: Let $\psi^D$ be the single potential in $\Psi_D$
10: $\psi'_D \leftarrow \max_D \psi^D$
11: $\widehat{\delta}_D \leftarrow \arg\max_D \psi^D$
12: return $\psi'_D$
```

Example 50 *In order to illustrate Algorithms 26 and 27, let us consider that we aim to remove decision D with* $\Omega_D = \{d_0, d_1, d_2\}$ *from* $\Phi_D = \{\phi(A|D)\}$ *and* $\Psi_D = \{\psi(A,D), \psi(D,B,A), \psi(D)\}$. *In lines 2 and 3 of Algorithm 26, decision is removed from the PPs. That is,* $\Phi'_D = \{\phi(A|D)^{R(D=d_0)}\} = \{\phi(A)\}$. *Then,* $max\text{-}marginalize(D, \Psi_D)$ *is invoked. The combination candidate set* B' *is* $\{\{\psi(A,D), \psi(D,B,A)\}, \{\psi(A,D), \psi(D)\}, \{\psi(D,B,A), \psi(D)\}\}$. *Suppose that the pair* $\{\psi(A,D), \psi(D)\}$ *is selected, then both UPs are added giving as a result a new UP* $\psi(A,D)$. *In the second iteration* B' *is* $\{\{\psi(A,D), \psi(D,B,A)\}\}$ *so the addition* $\psi(D,B,A) = \psi(A,D) + \psi(D,B,A)$ *is done. Finally, in lines 9 and 10 D is max-marginalized out and its optimal policy is recorded.*

12.3.5 Combination heuristics

During the removal of a set of chance variables, at each iteration a pair of PPs is selected to be combined (Algorithm 24, line 5). Since computing the cost of future combinations and marginalizations could be extremely expensive, the decision must be taken using a heuristic. Some heuristics used with VE for selecting the next variable to remove can be adapted for choosing a pair in the SPI algorithm. Let $p = \{\phi_i, \phi_j\}$ be a candidate pair to be combined, let $\phi_{ij} = \phi_i \cdot \phi_j$ be the resulting potential of the combination. Then, the heuristics *minimum size* [99], and *minimum weight* [68] are defined as:

$$min\_size(p) = |dom(\phi_i) \cup dom(\phi_j)| = |dom(\phi_{ij})| \tag{12.6}$$

$$min\_weight(p) = \prod_{X \in dom(\phi_{ij})} |\Omega_X| \tag{12.7}$$

The previous heuristics choose the next pair to combine using only information from the PPs involved. However, they do not consider if the pair chosen will imply a costly combination with the utilities. As explained in Section 12.3.3, utilities are only combined at the moment a variable can be removed. Let $\mathbf{W}$ be the set of variables that can be removed after combining potentials in the pair p, let $\Psi_{\mathbf{W}}$ be the set of UPs containing any variable in $\mathbf{W}$ and let $\psi = \sum_{\psi_k \in \Psi_{\mathbf{W}}} \psi_k$. Then the heuristic *minimum utility* can be defined as follows:

$$min\_utility(p) = \prod_{X \in dom(\phi_{ij})} |\Omega_X| \cdot \prod_{Y \in dom(\psi) \setminus dom(\phi_{ij})} |\Omega_Y| \tag{12.8}$$

Any of the heuristics previously mentioned can also be used for selecting a pair of utility potentials at steps 7 and 4 of Algorithms 25 and 27 respectively. These heuristics will be considered in the experimental analysis.

12.3.6 Probabilistic Barren

A variable is *probabilistic barren* if it is barren when only the set of probability potentials of the ID is considered. In other words, such variable only belongs to one PP and, at least to one UP. Let Y be probabilistic barren and let $\phi(Y|\mathbf{X}_I)$ be a PP, then:

$$\sum_Y \phi(Y|\mathbf{X}_I) = 1_{\mathbf{X}_I}$$

where $1_{\mathbf{X}_I}$ is a unity potential, that is a potential defined on $\mathbf{X}_I$ assigning the value 1 to each configuration of $\Omega_{\mathbf{X}_I}$. When a probabilistic barren variable is removed, the sum-marginalization and division can be avoided (Algorithm 25, line 16). By contrast, the sum-marginalization from the UPs cannot be avoided. Unlike barren nodes (see Section 4.4.2), probabilistic barren variables cannot be removed during the minimalization phase as they have impact on the decisions. In conclusion, the efficiency of the computation can be improved if singletons are allowed and a detection a priori of probabilistic barren variables is performed. This point is empirically demonstrated in the experimental Section 12.7.

12.3.7 Example

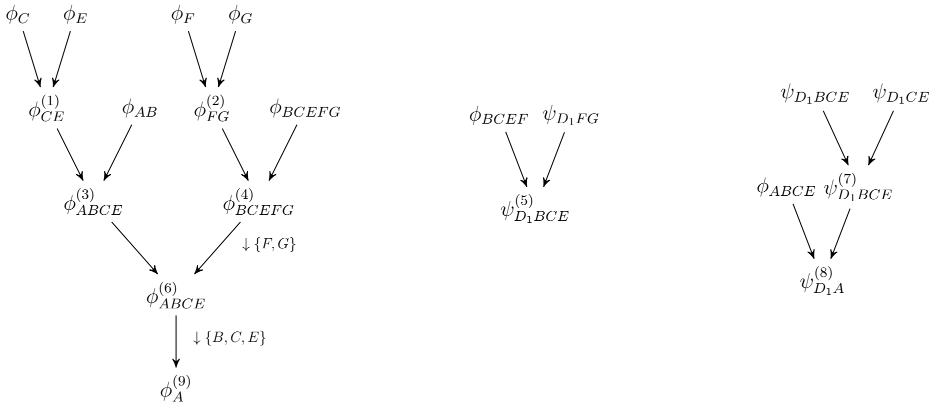
Let us consider the ID in Figure 12.1 in order to illustrate the behaviour of the SPI algorithm as described in this dissertation. For sake of simplicity, $\phi(X_1, \ldots, X_n)$ will be denoted $\phi_{X_1,\ldots,X_n}$. First, SPI proceeds to remove variables in the chance set $\mathcal{I}_1 = \{B, C, E, F, G\}$ using Algorithm 24. The initial combination candidate set is:

$$\{\phi_C, \phi_E\}, \{\phi_C, \phi_F\}, \{\phi_C, \phi_G\}, \{\phi_C, \phi_{BCEFG}\}, \{\phi_C, \phi_{AB}\}, \{\phi_E, \phi_F\},$$
$$\{\phi_E, \phi_G\}, \{\phi_E, \phi_{BCEFG}\}, \{\phi_E, \phi_{AB}\}, \{\phi_F, \phi_G\}, \{\phi_F, \phi_{BCEFG}\},$$
$$\{\phi_F, \phi_{AB}\}, \{\phi_G, \phi_{BCEFG}\}, \{\phi_G, \phi_{AB}\}, \{\phi_{BCEFG}, \phi_{AB}\}$$

If the *minimum size* heuristic is used for selecting the next pair of potentials, there are 6 pairs minimizing this score. Let us suppose that the pair $\{\phi_C, \phi_E\}$ is chosen, then the resulting potential is ϕ_{CE}. At this point there is not any variable that can be removed, since variables C and E are contained in another potential (e.g., ϕ_{BCEFG}). Then, the set B is updated by removing pairs containing ϕ_C or ϕ_E and by adding new pairwise combinations with ϕ_{CE}:

$$\{\phi_{CE},\phi_F\},\{\phi_{CE},\phi_G\},\{\phi_{CE},\phi_{BCEFG}\},\{\phi_{CE},\phi_{AB}\},\{\phi_F,\phi_G\},$$
$$\{\phi_F,\phi_{BCEFG}\},\{\phi_F,\phi_{AB}\},\{\phi_G,\phi_{BCEFG}\},\{\phi_G,\phi_{AB}\},\{\phi_{BCEFG},\phi_{AB}\}$$

The process will keep on choosing pairs to combine until all variables have been removed. The whole process is shown in Figure 12.2 using *factor graphs* [5]. Nodes without any parent correspond to initial potentials while child nodes to the resulting ones of a combination. The numbers above potentials indicate the combination order and arcs labels indicate the variables that are sum-marginalized.



(a) (b) (c)

Figure 12.2: Combination order of the potentials obtained using SPI for removing the chance set $\mathcal{I}_1 = \{B, C, E, F, G\}$ during the evaluation of the ID shown in Figure 12.1.

In the 4^{th} iteration, after generating the potential ϕ_{BCEFG}, variables F and G can be removed. Then, the Algorithm 25 is executed in order to combine utility potentials and max-marginalize these variables: the combination candidate set of utility potentials is $B = \{\{\psi_{D_1FG}\}\}$ and the resulting potentials are ϕ_{BCE} and ψ_{D_1BCE}. Similarly, in the 5^{th} iteration, variables B, C and E can be removed. Now, the combination candidate set contains a pair and a singleton, that is $B = \{\{\psi_{D_1CE}, \psi_{D_1BCE}\}, \{\psi_{D_1BCE}\}\}$. The element selected from B is the pair $\{\psi_{D_1CE}, \psi_{D_1BCE}\}$. The variables B, C and E can be removed after adding both utility potentials in the pair, thus it is not necessary to perform any additional iteration. The resulting potentials are ϕ_A and ψ_{D_1A} which are also, in this example, the resulting potentials in Algorithm 24. SPI will now proceed to remove decision D_1 using Algorithm 26 and chance variable A using Algorithm 24.

12.4 Correctness and complexity of SPI

In this section we prove the correctness of the SPI algorithm by proving that it is a correct reordering of the operations used by VE (the correctness of VE can be found in [64]). Evaluating an ID involves performing several sum-marginalizations and max-marginalizations of the variables in the ID (see Equations (4.24), (4.25) and (4.26)). The order of these two operations are not always interchangeable and therefore variables must be removed according to a strong elimination order. This condition is satisfied by the SPI algorithm (as described in Algorithm 20): *RemoveChanceSet* sum-marginalizes out $\mathcal{I}_n$, *RemoveDecision* max-marginalizes out D_n, *RemoveChanceSet* sum-marginalizes out $\mathcal{I}_{n-1}$, etc.

As explained in Section 12.2.1, the *RemoveChanceSet* algorithm should give as a result a factorization of probability potentials in the following form.

$$\sum_{\mathbf{X}} \prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}}} \phi_{\mathbf{X}}$$

Let us suppose that our method aims to remove a set of variables $\mathbf{V} \subseteq \mathbf{X}$ (Algorithm 25, line 16). Previously all PPs in $\Phi'_{\mathbf{X}} \subseteq \Phi_{\mathbf{X}}$ have been combined. In other words, the input argument ϕ of Algorithm 25 is equal to $\prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}}} \phi_{\mathbf{X}}$. Notice that it is guaranteed that any PP containing any variable in $\mathbf{V}$ has been combined (Algorithm 24, line 13). Using the distributive law we get:

$$\begin{aligned}
&\sum_{\mathbf{X}} \prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}}} \phi_{\mathbf{X}} = \\
&= \sum_{\mathbf{X}} \left(\prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}} \setminus \Phi'_{\mathbf{X}}} \phi_{\mathbf{X}} \prod_{\phi_i \in \Phi'_{\mathbf{X}}} \phi_i \right) = \\
&= \sum_{\mathbf{X} \setminus \mathbf{V}} \left(\prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}} \setminus \Phi'_{\mathbf{X}}} \phi_{\mathbf{X}} \sum_{\mathbf{V}} \prod_{\phi_i \in \Phi'_{\mathbf{X}}} \phi_i \right) = \\
&= \sum_{\mathbf{X} \setminus \mathbf{V}} \left(\prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}} \setminus \Phi'_{\mathbf{X}}} \phi_{\mathbf{X}} \sum_{\mathbf{V}} \phi \right) = \\
&= \sum_{\mathbf{X} \setminus \mathbf{V}} \left(\prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}} \setminus \Phi'_{\mathbf{X}}} \phi_{\mathbf{X}} \phi'_{\mathbf{V}} \right)
\end{aligned}$$

We see that the result of eliminating $\mathbf{V}$ implies removing each potential in $\Phi'_{\mathbf{X}}$ and replacing them by ϕ'_V. This is done in line 20 of Algorithm 24. Therefore, operations with probabilities are correct. Similarly, *RemoveChanceSet* algorithm should give also as a result a factorization of UPs in the following form.

$$\alpha \sum_{\mathbf{X}} \prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}}} \phi_{\mathbf{X}} \left(\sum_{\psi_{\mathbf{X}} \in \Psi_{\mathbf{X}}} \psi_{\mathbf{X}} \right) \tag{12.9}$$

where $\alpha = \sum_{\mathbf{X}} \prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}}} \phi_{\mathbf{X}}$. Let us now consider computations with UPs needed for removing $\mathbf{V}$. Previously all the addition of all UPs in $\Psi'_{\mathbf{X}} \subseteq \Psi_{\mathbf{X}}$ has been done. Notice that it is guaranteed that any utility potential containing any variable in $\mathbf{V}$ has been combined (Algorithm 25, line 15). Using the distributive law we obtain:

$$
\begin{aligned}
&\alpha \sum_{\mathbf{X}} \prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}}} \phi_{\mathbf{X}} \left(\sum_{\psi_{\mathbf{X}} \in \Psi_{\mathbf{X}}} \psi_{\mathbf{X}} \right) = \\
&= \alpha \sum_{\mathbf{X}} \left(\prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}} \setminus \Phi'_{\mathbf{X}}} \phi_{\mathbf{X}} \prod_{\phi_i \in \Phi'_{\mathbf{X}}} \phi_i \left(\sum_{\psi_{\mathbf{X}} \in \Psi_{\mathbf{X}} \setminus \Psi'_{\mathbf{X}}} \psi_{\mathbf{X}} + \sum_{\psi_i \in \Psi'_{\mathbf{X}}} \psi_i \right) \right) = \\
&= \alpha \sum_{\mathbf{X}} \left(\prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}} \setminus \Phi'_{\mathbf{X}}} \phi_{\mathbf{X}} \prod_{\phi_i \in \Phi'_{\mathbf{X}}} \phi_i \sum_{\psi_{\mathbf{X}} \in \Psi_{\mathbf{X}} \setminus \Psi'_{\mathbf{X}}} \psi_{\mathbf{X}} + \prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}} \setminus \Phi'_{\mathbf{X}}} \phi_{\mathbf{X}} \prod_{\phi_i \in \Phi'_{\mathbf{X}}} \phi_i \sum_{\psi_i \in \Psi'_{\mathbf{X}}} \psi_i \right) = \\
&= \alpha \sum_{\mathbf{X} \setminus \mathbf{V}} \prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}} \setminus \Phi'_{\mathbf{X}}} \phi_{\mathbf{X}} \sum_{\mathbf{V}} \prod_{\phi_i \in \Phi'_{\mathbf{X}}} \phi_i \left(\sum_{\psi_{\mathbf{X}} \in \Psi_{\mathbf{X}} \setminus \Psi'_{\mathbf{X}}} \psi_{\mathbf{X}} + \frac{\sum_{\mathbf{V}} \left(\prod_{\phi_i \in \Phi'_{\mathbf{X}}} \phi_i \sum_{\psi_i \in \Psi'_{\mathbf{X}}} \psi_i \right)}{\sum_{\mathbf{V}} \prod_{\phi_i \in \Phi'_{\mathbf{X}}} \phi_i} \right) = \\
&= \frac{\sum_{\mathbf{X} \setminus \mathbf{V}} \prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}} \setminus \Phi'_{\mathbf{X}}} \phi_{\mathbf{X}} \phi'_{\mathbf{V}} \left(\sum_{\psi_{\mathbf{X}} \in \Psi_{\mathbf{X}} \setminus \Psi'_{\mathbf{X}}} \psi_{\mathbf{X}} + \psi'_{\mathbf{V}} \right)}{\sum_{\mathbf{X} \setminus \mathbf{V}} \prod_{\phi_{\mathbf{X}} \in \Phi_{\mathbf{X}} \setminus \Phi'_{\mathbf{X}}} \phi_{\mathbf{X}} \phi'_{\mathbf{V}}}
\end{aligned}
$$

From previous expression, we see that the result of eliminating $\mathbf{V}$ also implies removing each potential in $\Psi'_{\mathbf{X}}$ and replacing them by ψ'_V. This is done in line 23 of Algorithm 25. Therefore, operations with UPs are also correct. When *removeDecision* is invoked for removing D_n out of Ψ_D, the result should be $\psi'_D = \max_D \sum_{\psi_D \in \Psi_D} \psi_D$. This procedure is correct since in Algorithm 27 max-marginalization operation is not made until the addition of all UPs is performed.

Similarly to VE, the complexity of the SPI algorithm is also linear to the size of the largest potential generated during the evaluation. The largest potential will be obtained right before the sum-marginalization or max-marginalization of a variable. That is, the result of the combination $\phi \cdot \psi_{ij}$ in line 16 of Algorithm 25 or the potential ψ^D in line 9 of Algorithm 27. Thus, the complexity of the SPI algorithm for evaluating an ID with n variables (chance or decision) is $\mathcal{O}(nN'_{max})$ where N'_{max} is the largest potential ever created during the evaluation. However, the size of a potential is exponential to the number of variables in its domain. Thus, the computational cost of the VE algorithm depends on the sizes of the intermediate potentials generated and on the order of operations with potentials.

12.5 SPI Lazy Evaluation

LE is based on message passing between cliques in a strong junction tree (see Section 4.5.3). Basically, the computation of these messages consists of removing variables not present in the parent separator from the sets of potentials in a clique. The original approach [79] uses VE for removing the variables. Thus, we will refer to this method as VE-Lazy Evaluation (VE-LE). Here we propose SPI Lazy Evaluation (SPI-LE), which is a variant of Lazy Evaluation that uses SPI instead of VE in order to compute the messages. This idea was already considered in a previous paper [78] where the SPI algorithm was used in Lazy Propagation in BNs. The process for building the strong junction tree and *Collect Message* algorithm are the same. The general scheme of the *Absorption* algorithm is slightly different (see Algorithm 28).

Algorithm 28 Absorption-SPI

```
$(\Phi^*_{S_j}, \Psi^*_{S_j}) \leftarrow (\Phi_{C_j} \cup \bigcup_{S' \in ch(C_j)} \Phi^*_{S'}, \Psi_{C_j} \cup \bigcup_{S' \in ch(C_j)} \Psi^*_{S'})$   ▷ Relevant potentials
$\mathbf{X} \leftarrow \{X | X \in C_j, X \notin S_j\}$   ▷ Variables to remove
Partition $\mathbf{X}$ into disjoint sets of chance variables or sets of single decisions. Let
  $\{\mathbf{X}_1 \prec \mathbf{X}_2 \prec \cdots \prec \mathbf{X}_n\}$ a partial order respecting the temporal constraints.
for $k \leftarrow n$ to $1$ do
    $(\Phi_{\mathbf{X}_k},\ \Psi_{\mathbf{X}_k}) \leftarrow (\{\phi \in \Phi^*_S | \mathbf{X}_k \cap dom(\phi) \neq \emptyset\},\ \{\psi \in \Psi^*_S | \mathbf{X}_k \cap dom(\psi) \neq \emptyset\})$
    $(\Phi^*_{S_j},\ \Psi^*_{S_j}) \leftarrow (\Phi^*_{S_j} \backslash \Phi_{\mathbf{X}_k},\ \Psi^*_{S_j} \backslash \Psi_{\mathbf{X}_k})$
    if $\mathbf{X}_k \subseteq \mathcal{U}_C$ then
        $(\Phi'_{\mathbf{X}_k},\ \Psi'_{\mathbf{X}_k}) \leftarrow RemoveChanceSet(\mathbf{X}_k, \Phi_{\mathbf{X}_k}, \Psi_{\mathbf{X}_k})$   ▷ Algorithm 24
        $(\Phi^*_{S_j},\ \Psi^*_{S_j}) \leftarrow (\Phi^*_{S_j} \cup \Phi'_{\mathbf{X}_k},\ \Psi^*_{S_j} \cup \Psi'_{\mathbf{X}_k})$
    else
        Let $D_k$ the single variable in $\mathbf{X}_k$
        $(\Phi'_{\mathbf{X}_k},\ \psi'_{\mathbf{X}_k}) \leftarrow RemoveDecision(D_k, \Phi_{\mathbf{X}_k}, \Psi_{\mathbf{X}_k})$   ▷ Algorithm 26
        $(\Phi^*_{S_j},\ \Psi^*_{S_j}) \leftarrow (\Phi^*_{S_j} \cup \Phi'_{\mathbf{X}_k},\ \Psi^*_{S_j} \cup \{\psi'_{\mathbf{X}_k}\})$
    end if
end for
Associate $\Phi^*_{S_j}$ and $\Psi^*_{S_j}$ to the parent separator $S_j$.
```

The main difference is that the set of variables to remove is partitioned into disjoint subsets of chance variables or single sets of decisions because variables in an ID should be removed according to an order that respects the temporal constraints. Notice that the removal of a subset of variables $\mathbf{X}_k$ is invoked only on the set of potentials containing any variable in $\mathbf{X}_k$. Another difference is the way variables are removed: in SPI-LE, procedures explained in Section 12.3.3 and 12.3.4 are used instead of VE.

12.6 Optimization of variable elimination

When evaluating an ID using the VE algorithm, variables are removed in reverse order of information precedence (see Section 4.5.1). The removal of a variable X_i implies combining all PPs and UPs with X_i in the domain (Algorithm 2 line 2). Here we propose using a greedy algorithm for optimizing the combination of the potentials involved in the removal of a variable. The procedure for combining a set of PPs is shown in Algorithm 29.

Algorithm 29 combineProbabilities

```
input :  $\Phi_{X_i}$ (set of PPs relevant for removing $X_i$)
output :  $\phi_{X_i}$ (PP resulting from combining all the potentials in $\Phi_{X_i}$)
$B \leftarrow \emptyset$
while $|\Phi_{X_i}| > 1$ do
    $B \leftarrow addPairwiseCombinations(\Phi_{X_i}, B)$
    $p \leftarrow selectBest(B)$                          ▷ $p$ is a pair $\{\phi_i, \phi_j\}$
    [ $\phi_{ij} \leftarrow \phi_i \cdot \phi_j$]
    $B \leftarrow \{p \in B' | p' \cap p = \emptyset\}$                ▷ Update
    $\Phi_{X_i} \leftarrow (\Phi_{X_i} \backslash p) \cup \{\phi_{ij}\}$
end while
Let $\phi_{X_i}$ be the single potential in $\Phi_{X_i}$
return $\phi_{X_i}$
```

This algorithm is quite similar to the procedure used by Shenoy [106] for building the binary join trees. However, in our approach no tree-like structure is created, potentials are directly combined. A similar approach must be considered for adding all UPs (see Algorithm 30).

Algorithm 30 addUtilities

```
input :  $\Psi_{X_i}$ (set of UPs relevant for removing $X_i$)
output :  $\psi_{X_i}$ (UP resulting from combining all the potentials in $\Psi_{X_i}$)
$B' \leftarrow \emptyset$
while $|\Psi_{X_i}| > 1$ do
    $B' \leftarrow addPairwiseCombinations(\Psi_{X_i}, B')$
    $q \leftarrow selectBest(B')$                         ▷ $q$ is a pair $\{\psi_i, \psi_j\}$
    $\psi_{ij} \leftarrow \psi_i + \psi_j$
    $B' \leftarrow \{q' \in B' | q' \cap q = \emptyset\}$                 ▷ Update
    $\Psi_{X_i} \leftarrow (\Psi_{X_i} \backslash q) \cup \{\psi_{ij}\}$
end while
Let $\psi_{X_i}$ be the single potential in $\Psi_{X_i}$
return $\psi_{X_i}$
```

Basically, in the two previous algorithms, at each iteration a pair of potentials is selected to be combined. For that, we use the combination candidate set as previously described in Section 12.3.2. For determining which is the best pair to combine, heuristics explained in Section 12.3.5 can be used. These algorithms stop when all the potentials have been combined.

12.7 Experimental work

12.7.1 Procedure and objectives

In general, the aim of the experimental work is to analyse the behaviour of all the algorithms considered in this chapter. We compare VE and SPI for directly evaluating an ID and for computing clique-to-clique messages in LE as well. The objectives of this experimentation are:

(a) Analyse if the SPI algorithm offers better results if probabilistic barren nodes are exploited and singletons are allowed. The combination heuristics explained in Section 12.3.5 are also compared.

(b) Compare the improved version of VE (Section 12.6) with the original one.

(c) Compare the algorithms VE and SPI.

A set of 18 IDs from the literature are used: NHL and Jaundice are two real world IDs used for medical purposes [76, 97]; the Appendicitis ID is a synthetic diagram modelling a medical decision problem [69]; the oil wildcatter's problem with one and two utilities [96, 40]; an ID representing the Car Buyer problem [94]; an ID used to evaluate the population viability of wildlife species [81]; the Chest Clinic ID [55] obtained from the Asia BN; an ID representing the decision problem in the poker game [64]; two different IDs used at agriculture for treating mildew [64]; an ID to model a simplified version of the dice game called *Thinkbox* [57];an ID for solving the maze problem [113]; finally, three synthetic IDs are used: the motivation example shown in Figure 12.1 and two IDs proposed by Jensen et al.[62]. The details of these IDs are shown in Table 12.1, which contains the number of nodes of each kind, the size of the largest partition $\mathcal{I}_i$ of chance nodes and the total table size (number of entries in a table containing all the variables).

ID	$\lvert\mathcal{U}_C\rvert$	$\lvert\mathcal{U}_D\rvert$	$\lvert\mathcal{U}_V\rvert$	$\max\lvert\mathcal{I}_i\rvert$	average potential size
Appendicitis	4	1	1	2	3.6
Car Buyer	3	3	1	1	64.5
ChestClinic	8	2	2	5	5.2
Competitive Asymm.	10	9	1	10	35.182
Jaundice	21	2	1	10	41.5
Jensen et al. 1	4	2	2	3	5
Jensen et al. 2	12	4	4	8	4.875
Maze	14	2	1	6	3100.2
Mildew 1	6	1	2	6	28.375
Mildew 4	7	2	2	4	32.222
Motivation ID	6	1	2	5	27.5
NHL	17	3	1	11	468.111
Oil	2	2	2	1	7.25
Oil Split Costs	2	2	3	1	6.2
Poker	7	1	1	7	94
Thinkbox	5	2	4	2	20.444
Threat of Entry	3	9	1	3	46
Wildlife	9	1	1	9	5

Table 12.1: Features of the IDs used in the experimentation. More details about these models are given in Appendix B.2.

To compare VE and SPI for computing the clique-to-clique messages, a strong junction tree is built from each ID using the *minimum size* heuristic [99] for triangulating the graph. Table 12.2 shows, for each tree, the number of cliques, the minimum and maximum clique sizes $|C|$ and clique weights $w(C)$.

The SPI algorithm and the improved version of VE may have non-trivial additional computational costs for selecting the next potentials to combine. In order to check that these new algorithms do not have a large overhead, all the comparisons are made in terms of computation time. Moreover, the portion of time corresponding to this overhead is shown in all the graphics. To avoid the influence of outliers, each ID is evaluated 100 times with each evaluation scheme.

		$\vert C\vert$		$w(C)$	
ID	Cliques	min	max	min	max
Appendicitis	1	4	4	16	16
Car Buyer	1	6	6	384	384
ChestClinic	5	3	6	8	64
Competitive Asymm.	3	4	6	96	1152
Jaundice	9	4	12	16	$1.555{\cdot}10^5$
Jensen et al. 1	3	3	4	8	16
Jensen et al. 2	9	3	5	8	32
Maze	3	5	12	$1.984{\cdot}10^4$	$4.032{\cdot}10^5$
Mildew 1	2	3	3	64	112
Mildew 4	4	4	6	256	9408
Motivation ID	2	3	6	30	432
NHL	8	5	12	32	$5.530{\cdot}10^5$
Oil	1	4	4	36	36
Oil Split Costs	1	4	4	36	36
Poker	5	3	3	32	324
Thinkbox	2	4	6	8	384
Threat of Entry	3	4	9	48	1728
Wildlife	7	3	4	8	16

Table 12.2: Features of the strong junction trees used for the experimental work obtained with *minimum size* heuristic.

12.7.2 Singletons and probabilistic barren

Here, the objective (a) is considered, that is we analyse if the efficiency of the computation can be improved if singletons are allowed and a detection a priori of probabilistic barren is performed. For that purpose each ID is evaluated using different schemes and algorithms. For the basic version of the SPI algorithm, four evaluation schemes are considered: SPI, SPI_B, SPI_S and SPI_{BS} where the subscript B means that probabilistic barren nodes are detected and the subscript S means that singletons are allowed. Similarly, each ID is also evaluated using the SPI-LE algorithm considering the schemes $SPI\text{-}LE$, $SPI\text{-}LE_B$, SPI-

LE_S and $SPI\text{-}LE_{BS}$. Figure 12.3 shows the average computation time needed for evaluating each ID using the schemes of the basic version of the SPI algorithm. The combination heuristics considered are those explained in Section 12.3.5. For the majority of the IDs, the algorithm without any improvement (SPI) offers the worst performance. By contrast, the best results are obtained if both of the improvements proposed are applied (SPI_{BS}). This scheme is the fastest for evaluating 11, 14, and 10 networks when using the heuristics min_size, min_weight and $min\_utility$ respectively.

If we analyse the effect of adding singletons to the combination candidate set, we can observe that for some large IDs such as NHL the computation time can increase if these improvements are considered (SPI_S and SPI_{BS} with min_weight heuristic). The reason for that is that the search space is too large and the algorithm will give preference to selecting singletons even if the operations with the utilities are costly. This problem disappears if the heuristic considers the cost of operations with the utility ($min\_utility$). The detection of probabilistic barren nodes (SPI_B) does not have any drawback: it is a simple procedure that will never increase the number of operations and in many cases will reduce it.

Figure 12.3 also includes the overhead introduced by the SPI algorithm (bars in black). That is, the time required for selecting the next pair to combine (operations with the combination candidate set) and for updating the sets of potentials and variables. It can be observed that for most of the IDs, this overhead is small or insignificant: most of the time corresponds with the time required for computing with potentials. However, when evaluating the Wildlife ID the overhead is high. In this ID, all the chance nodes are in a single and large partition $\mathcal{I}$. As the overhead increases exponentially in the number of potentials, the overhead will be high. Even though there are other IDs with partitions of a similar size, they contain larger potentials. As a consequence, the overhead is smaller compared to the time required for computing with potentials.

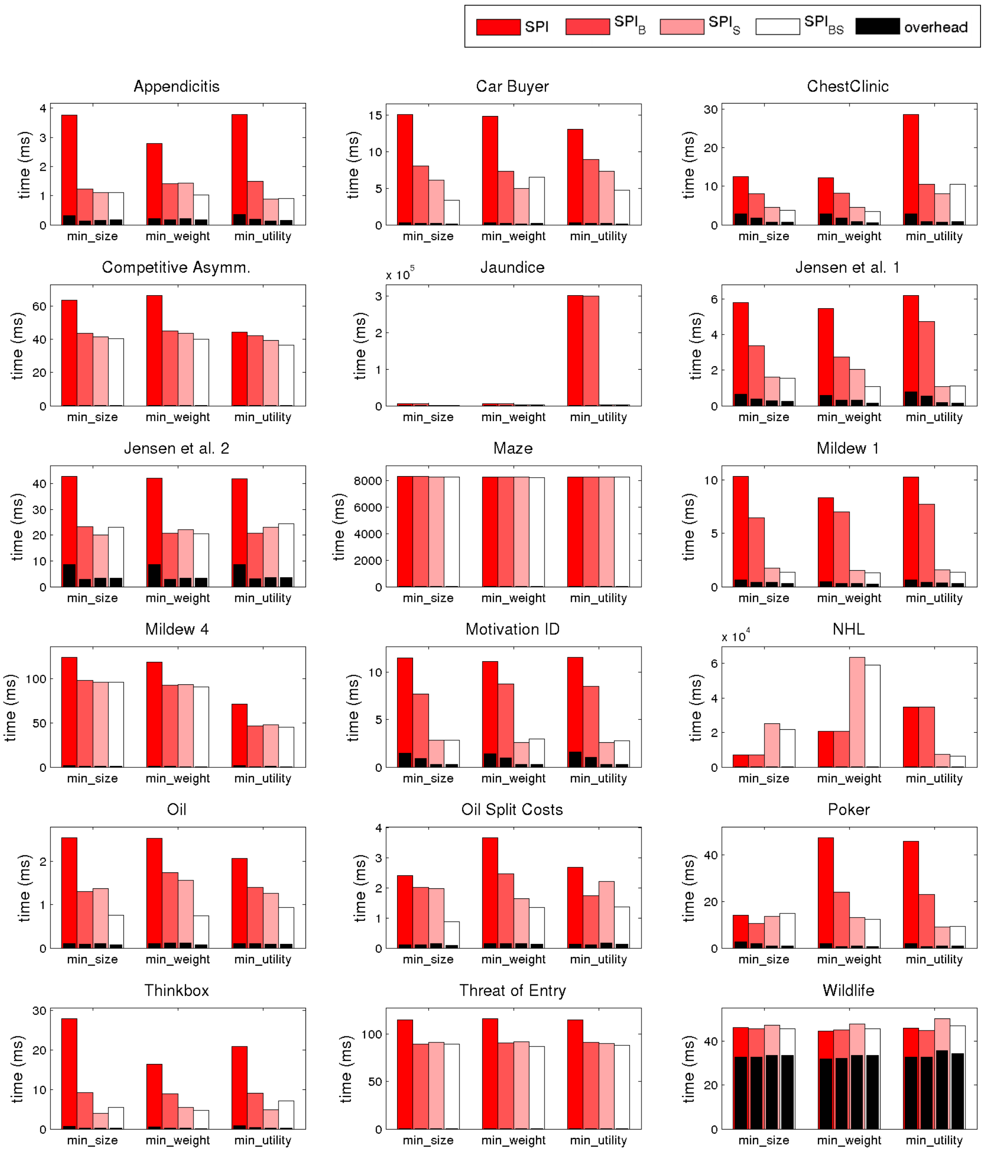


Figure 12.3: Comparison of the computation time using the basic version of SPI with different combination heuristics and considering the improvements of singletons and probabilistic barren (SPI_B, SPI_S and SPI_{BS}) and without them (SPI).

Table 12.3 shows the total time for evaluating all the IDs. It can be observed that the lowest cumulative time is obtained with SPI_{BS} using the *minimum utility* heuristic. Considering also that, in most of the IDs, the scheme SPI_{BS} offers the best performance and that the heuristic min_weight avoids the problems produced by considering the singletons, we state that the best results are obtained with SPI_{BS} using the *minimum utility* heuristic.

	min_size	min_weight	$min\_utility$
SPI	21540	35100	345100
SPI_B	21350	34980	343800
SPI_S	35400	73960	17650
SPI_{BS}	31730	69250	**16420**

Table 12.3: Cumulative time (ms) for evaluating all the IDs using the basic version of the SPI algorithm with different combination heuristics and considering the improvements of singletons and probabilistic barren (SPI_B, SPI_S and SPI_{BS}) and without them (SPI).

Similarly, Figure 12.4 shows the average time required for evaluating each ID using the four schemes of LE with SPI for computing the clique-to-clique messages (SPI-LE, SPI-LE_B, SPI-LE_S and SPI-LE_{BS}). This evaluation time includes the time required for building the strong junction tree and for propagating the messages.

It can be observed that, for most of the considered IDs, SPI-LE_{BS} is the most efficient scheme. Moreover, it can also be observed that the problem about the growth of the evaluation time in large IDs disappears: the search space for selecting a pair is now smaller (a large part of this search is now made during the building of the strong junction tree). In fact, there are less differences between schemes and heuristics: the combinatorial problem is divided into several subproblems. As a consequence there is less room for improvement but the overhead is smaller. In fact, the large overhead introduced by the algorithm for evaluating the Wildlife ID is now insignificant. SPI-LE_{BS} with *minimum weight* will be considered as the best scheme since it offers the best results for all the IDs used in the experimentation. In addition, the lowest cumulative time for evaluating all the IDs is obtained with this heuristic (see Table 12.4).

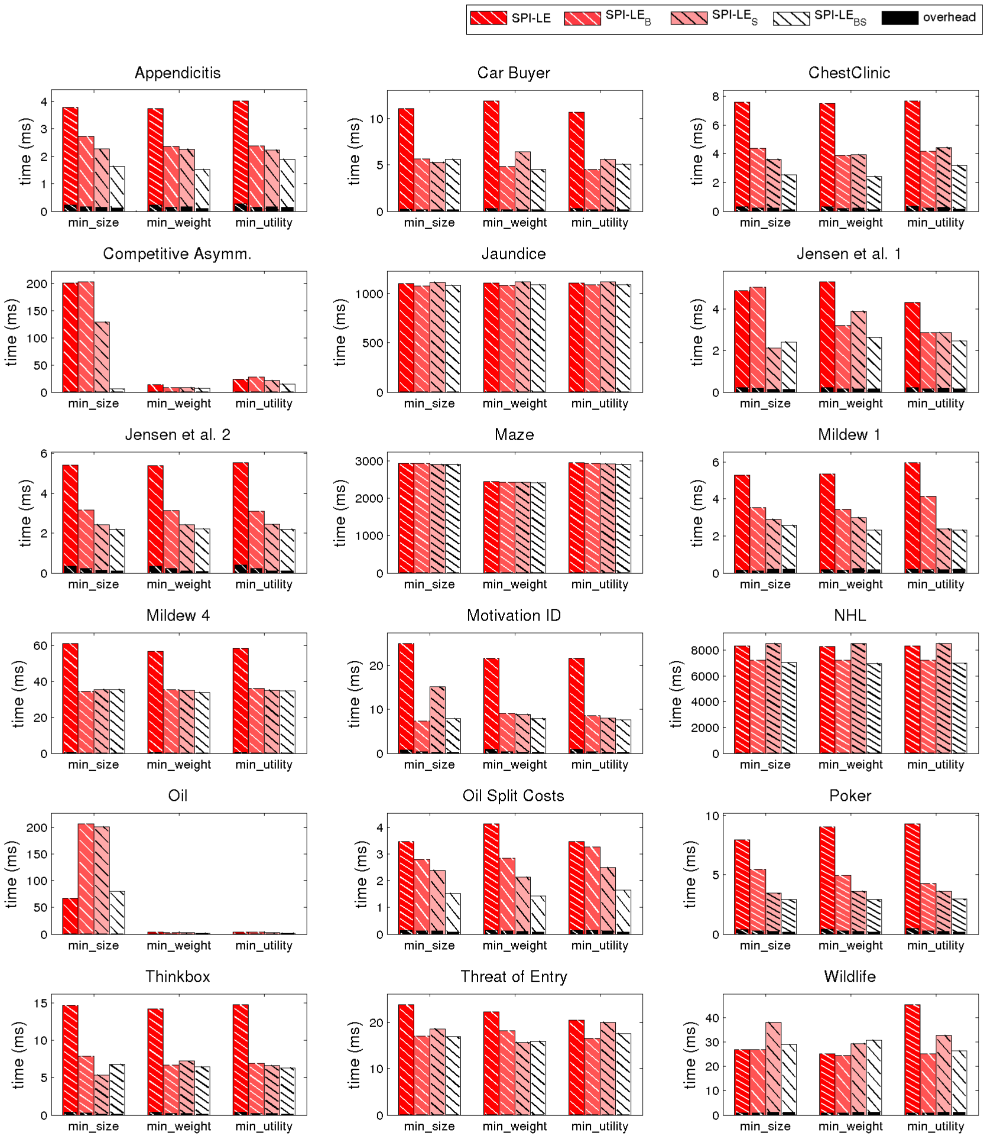


Figure 12.4: Comparison of the average computation time using the basic version of the SPI-LE algorithm with different combination heuristics and considering the improvements of singletons and probabilistic barren (SPI-LE_B, SPI-LE_S and SPI-LE_{BS}) and without them (SPI-LE).

	min_size	min_weight	$min\_utility$
$SPI\text{-}LE$	12800	12050	12610
$SPI\text{-}LE_B$	11790	10880	11410
$SPI\text{-}LE_S$	12980	12160	12690
$SPI\text{-}LE_{BS}$	11210	**10590**	11100

Table 12.4: Cumulative time for evaluating all the IDs using the basic version of the SPI-LE algorithm with different combination heuristics and considering the improvements of singletons and probabilistic barren ($SPI\text{-}LE_B$, $SPI\text{-}LE_S$ and $SPI\text{-}LE_{BS}$) and without them ($SPI\text{-}LE$).

12.7.3 Optimization of variable elimination

In Section 12.6 a variation of the algorithm VE is proposed. This version of the algorithm optimizes the combination of the potentials involved in the removal of a variable using a greedy algorithm. Thus, the performance of VE should be improved (objective (b)). In order to simplify the experimentation, the equivalent heuristic for selecting the variable to remove is used for selecting the pair of potentials to combine. In particular, heuristics *minimum size* and *minimum weight* are considered.

Figure 12.5 shows the computation required time by the basic version of VE and the optimized one (VE_{opt}) for evaluating each ID. For most of the IDs, the optimized version requires less time and, in those IDs where VE_{opt} offers the worst performance, the results are quite similar. These graphics also include the overhead introduced by the optimization but also the one corresponding to the time required for choosing the next variable to remove. It can be observed that the optimization does not introduce a large overhead.

VE_{opt} using *minimum size* or *minimum weight* heuristics are considered as the best configuration schemes since the total time for evaluating all the IDs are the lowest (see Table 12.5).

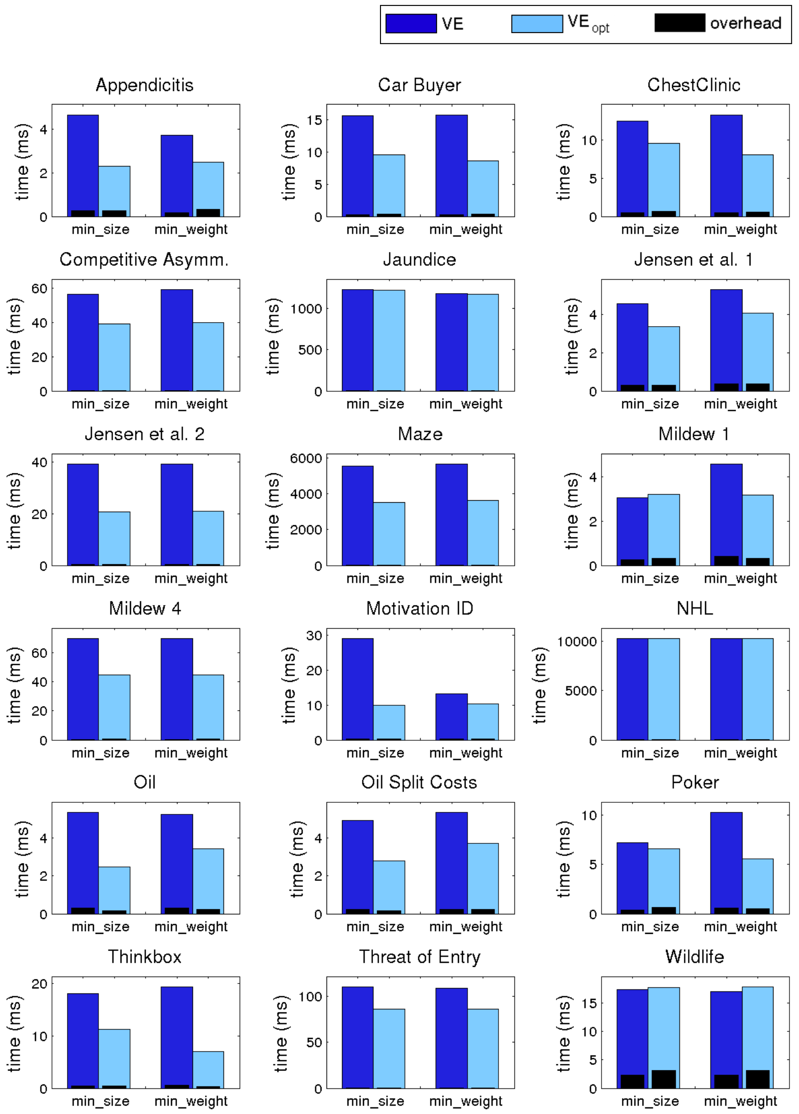


Figure 12.5: Comparison of the average computation time required by VE and the optimized version with different heuristics.

	min_size	min_weight
VE	1749	1751
VE_{opt}	**1532**	1534

Table 12.5: Cumulative time for evaluating all the IDs using VE and the optimized version with different heuristics.

Similarly, Figure 12.6 shows the comparison of LE using both methods for computing the clique-to-clique messages (VE-LE and VE_{opt}-LE). Again, the optimized version is faster for evaluating most of the IDs. VE_{opt}-LE with any of the heuristics are considered as the best configuration schemes for computing the clique-to-clique messages since the cumulative time is the lowest (see Table 12.6).

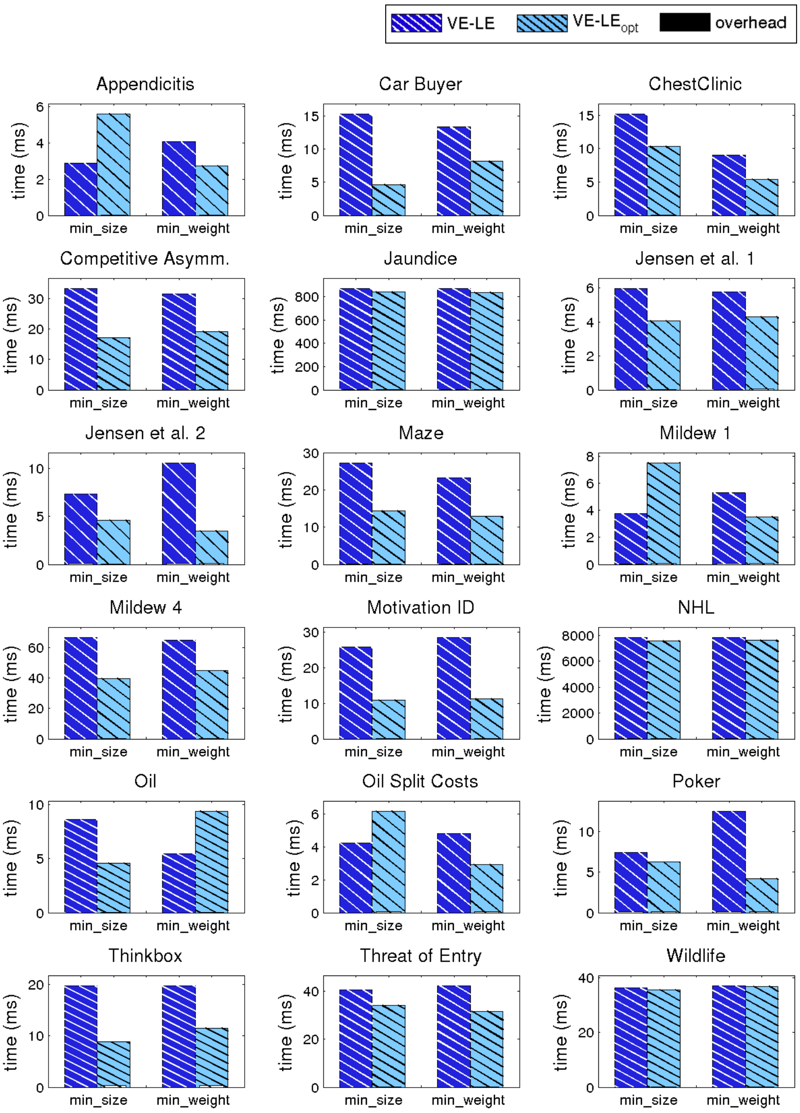


Figure 12.6: Comparison of the average computation time required by the basic $VE - LE$ and the optimized version with different heuristics.

	min_size	min_weight
VE-LE	9045	9048
VE-LE_{opt}	**8621**	8662

Table 12.6: Cumulative time for evaluating all the IDs using $VE - LE$ and the optimized version with different heuristics.

12.7.4 Comparison of SPI and VE

In previous subsections, it has been studied which is the best configuration for the algorithms SPI and VE. Figure 12.7 shows the average computation time comparing the best scheme of VE against SPI and the best scheme of VE-LE against SPI-LE for evaluating each ID (objective (c)).

First, it is compared VE_{opt} using the *minimum size* heuristic against SPI_{BS} with the *minimum utility* heuristic. The SPI algorithm offers the best results in 10 out 18 IDs. If we analyse those IDs were VE offers better performance than SPI, in 3 of them there are not great differences (ChestClinic, Jensen et al. 2 and Threat of Entry). Secondly both algorithms are also compared for computing clique-to-clique messages. That is, VE-LE_{opt} using *minimum size* heuristic against SPI_{BS} with the *minimum weight* heuristic. Now the SPI algorithm for computing the messages offers the best results for evaluating 16 out of 18 IDs.

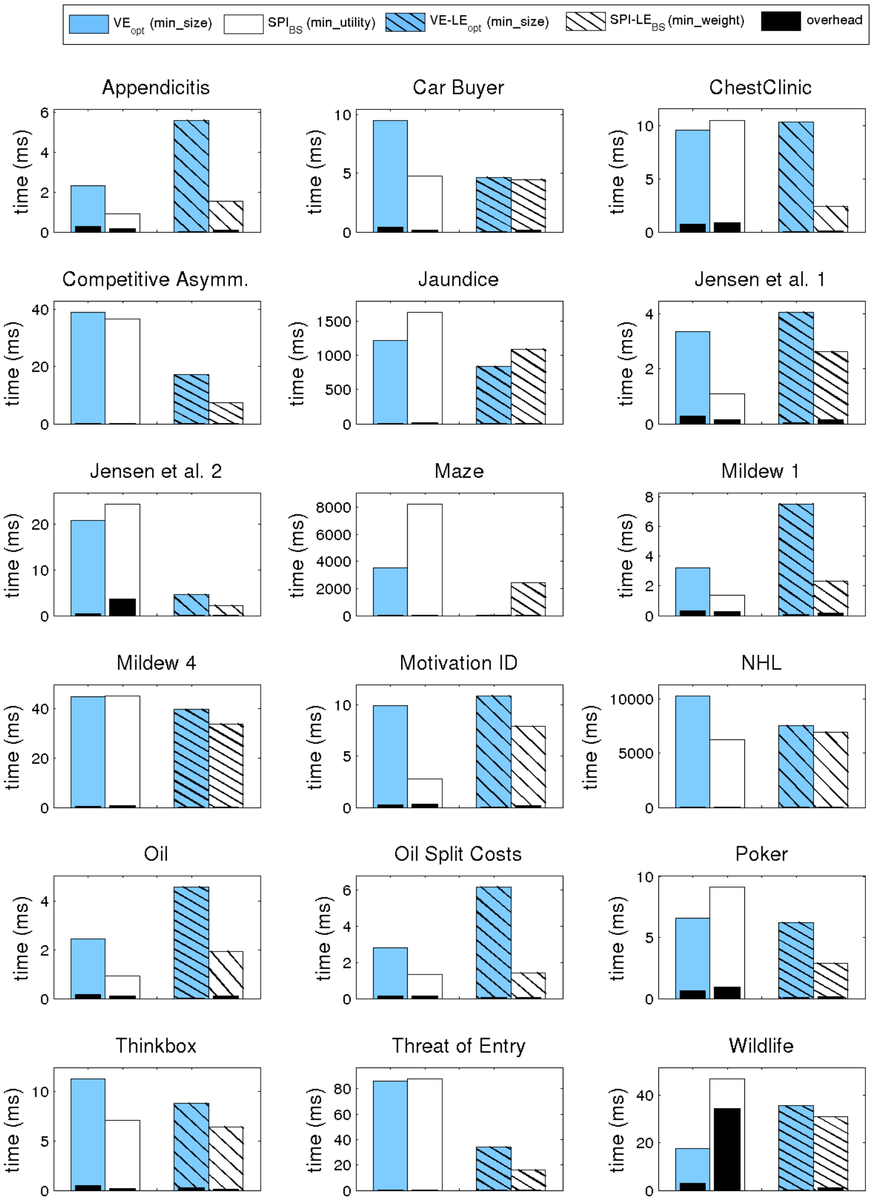


Figure 12.7: Average computation time comparing the best scheme of VE against SPI and the best scheme of VE-LE against SPI-LE for evaluating each ID.

12.7.5 Pre-analysis algorithm

In the results previously given, we have seen that SPI can outperform VE in many IDs. However, it cannot be assured that this will always be the case. The efficiency of these algorithms depends on several heuristics for determining the order of the operations involved in the evaluation. Therefore, we cannot assure which method (or heuristic) will offer the best results with a given ID until we evaluate it. However, we know that there is a correlation between the number of arithmetic operations and the efficiency of the methods. Table 12.7 shows the number of arithmetic operations and evaluation time for evaluating each ID with VE_{opt} and SPI_{BS} using the heuristics min_size and $min\_utility$ respectively. It can be observed that in 16 out of 18 a reduction in the number of operations means a reduction in the evaluation time. In the largest IDs (NHL, Jaundice and Maze), where it is more important to determine which is the best method, there is a high correlation. By contrast, in the Car Buyer and Mildew 1 IDs the reduction in the number of operations does not mean a reduction in the evaluation time. These two IDs are quite small and the time required by the arithmetic operations has a lower weight in the total evaluation time.

Taking into account this correlation, a pre-analysis algorithm for predicting which method is the most efficient one can be developed. This algorithm computes the number of arithmetic operations by evaluating an ID with each method in a *qualitative* way. This kind of evaluation is similar to the numerical (usual) evaluation but it does not perform the operations with potentials (domain of the resulting potentials are only computed). As the pre-analysis algorithm knows the domains of potentials involved in each operation with potentials, the number of arithmetic operations can be easily computed. Finally, the pre-analysis algorithm will base its decision on the of number arithmetic operations.

In the rightmost column of Table 12.7, the time for performing this pre-analysis. It can be observed that, for large IDs, the pre-analysis time is much smaller than the time required for evaluating the ID. By contrast, in smaller IDs, this pre-analysis could take more time than the evaluation.

	$\#\ operations$		$time\ (ms.)$		
	VE_{opt}	SPI_{BS}	VE_{opt}	SPI_{BS}	*time*
ID	min_size	$min\_utility$	min_size	$min\_utility$	*pre-analysis*
Motivation ID	**5939**	**1982**	**9.927**	**2.776**	2.4
Oil	**133**	**121**	**2.456**	**0.932**	3.319
Oil Split Costs	**145**	**133**	**2.794**	**1.354**	1.753
NHL	$\mathbf{3.570{\cdot}10^{6}}$	$\mathbf{2.384{\cdot}10^{6}}$	$\mathbf{1.031{\cdot}10^{4}}$	**6240.162**	75.438
Jaundice	$\mathbf{4.791{\cdot}10^{5}}$	$\mathbf{8.704{\cdot}10^{5}}$	**1227.203**	**1639.544**	220.016
Maze	$\mathbf{1.802{\cdot}10^{6}}$	$\mathbf{4.846{\cdot}10^{6}}$	**3515.281**	**8256.285**	48.482
Car Buyer	1375	1487	9.536	4.748	3.334
Mildew 1	395	511	3.211	1.361	1.884
Mildew 4	$\mathbf{3.272{\cdot}10^{4}}$	$\mathbf{4.432{\cdot}10^{4}}$	**44.636**	**45.154**	6.446
Poker	**1586**	**7034**	**6.573**	**9.163**	2.886
ChestClinic	**593**	**2465**	**9.601**	**10.52**	3.572
Appendicitis	**65**	**59**	**2.309**	**0.896**	1.721
Jensen et al. 1	**123**	**119**	**3.351**	**1.105**	3.95
Jensen et al. 2	**449**	**779**	**20.725**	**24.358**	6.358
Thinkbox	**1891**	**1793**	**11.31**	**7.079**	3.86
Threat of Entry	**3755**	**3971**	**86.247**	**87.95**	4.774
Wildlife	**155**	**163**	**17.657**	**46.948**	5.092
Competitive Asymm.	**3431**	**3135**	**39.094**	**36.536**	2.747

Table 12.7: Number of arithmetic operations and evaluation time for evaluating each ID with VE_{opt} and SPI_{BS} using the heuristics min_size and $min\_utility$ respectively. The time for evaluating each ID with both methods in a qualitative way is also given (pre-analysis time).

12.8 Conclusions

In this chapter, two algorithms for optimizing the operation order in the evaluation of IDs are described. First, the details of the adaptation of the SPI algorithm for evaluating IDs are given. This method was already used for making inference on BNs and it is more fine-grained than other methods in the literature such as VE. For this adaptation, differences between IDs and BNs have been taken into account: two kinds of potentials, temporal order between variables, etc. Secondly, an optimization of VE have also been proposed. This improved version consists of using a greedy algorithm for minimizing the cost of the combination of all the potentials involved in the removal of a variable. Both algorithms have been described for the direct evaluation of IDs and for the computation of clique-to-clique messages as well.

In the experimental work, these algorithms have been tested using a set of IDs from the literature. It has been demonstrated that, for many of the IDs considered, the SPI algorithm is more efficient if singletons are allowed and a priori detection of probabilistic barren is performed. For some large IDs the improvement of allowing singletons can produce an important growth in the evaluation time. However, this problem is solved if a combination heuristic that also considers the size of the utility potentials is used. For the computation of clique-to-clique messages, where the search space is smaller, this growth disappears. By contrast, the detection of probabilistic barren will never increase the evaluation time (it will remain the same or lower). Secondly, it has also been demonstrated that the optimized version of VE offers better results than the basic version. Finally, the best configuration schemes of both algorithms have been compared. For the direct evaluation of IDs and for the clique-to-clique message computation, SPI can outperform VE in many IDs.

The proposed SPI algorithm only considers the next pair of potentials to combine. Thus, a line of future research could be to study the behaviour of the algorithm using a higher neighbourhood degree. It could also be interesting to make a study that let us to characterize which features of an ID make SPI outperform VE. Concerning to the pre-analysis, alternative indicators to the number of operations could be studied.

Part IV

Conclusions

Chapter 13

Conclusions and Future Work

This last chapter summarizes all the conclusions that have been presented throughout this dissertation. The list of publications with the majority of the work presented here is included. This chapter concludes with an enumeration of the future research to be performed on the topic.

The whole dissertation has been devoted, as its title indicates, to new data structures and methods for evaluating IDs. In particular, we have addressed some drawbacks of IDs: high computational cost, inefficient evaluation of asymmetric decision problems and incapacity to express imprecision.

We have proposed the use of BTs for representing and managing the potentials involved in influence diagrams. This kind of tree allows representing general forms of independencies that are more fine-grained compared to those encoded using other representations. This enhanced capacity allows to reduce the computational cost of the evaluation due to the smaller size of the potentials. Moreover, binary trees allow computing approximate solutions when exact inference is not feasible. Heuristic methods for building (from tables) and pruning BTs have been proposed. Besides, we have explained how some existing evaluation algorithms can be adapted for working with BTs. The experimental evaluation showed that, in general, this approach demands lower memory resources than other representations such as NTs or tables. As a consequence, the ID evaluation is usually faster using BTs. However, for some IDs it is necessary to approximate the potentials in order to obtain any benefits from the use of BTs. Another conclusion is that using BTs for evaluating IDs offers better approximate solutions than using NTs: the same error level is achieved with a lower computation time.

In relation to one of the algorithms adapted for working with BTs, namely VE, we have proposed two new heuristics for determining the elimination order. Unlike traditional heuristics, these consider that potentials are represented as BTs instead as tables. In doing so, the computational cost (storage and time) is reduced even more.

For addressing the problem of the computational cost, we have also explored different alternatives that allow to represent the potentials as tables. In particular, we have proposed the adaptation of the SPI algorithm for evaluating IDs and an optimization of VE. These methods aim to optimize the order of the operations with potentials involved in the evaluation. In the experimental work, these algorithms have been tested using a set of IDs from the literature. It has been demonstrated that, the algorithms proposed can improve the efficiency of the evaluation. When comparing these two new algorithms, we have seen that SPI can outperform the optimized version of VE in many IDs.

The second drawback of IDs, namely the inefficient evaluation of IDs representing asymmetric decision problems, can also be solved using BTs. In our approach, potentials are represented as BTs but also the qualitative information about the problem (constraints, due to asymmetries). As the same data structure is used for both, potentials and asymmetries, they can be easily applied in order to reduce the number of scenarios to consider. We have empirically proved that this approach usually improves the efficiency of the evaluation of IDs representing asymmetric decision problems. However, for very small IDs, the efficiency might not be improved due to the overhead introduced by the application of constraints.

Another proposal for the representation of potentials in IDs are interval-valued potentials. These are basically a generalization of the precise potentials that assigns intervals (i.e., a lower and upper bound) instead of sharp values to each configuration. We have proposed replacing the precise potentials in an ID by interval-valued potentials. In doing so, we address the problem of incapacity of standard IDs to express the imprecision or vagueness in their potentials. Some evaluation algorithms have been generalized in order to cope with the interval-valued case. These are outer approximations of the exact solutions. Yet, the use of linear programming methods avoids to produce unnecessarily large outer approximations without increasing the computational complexity: this remains the same as with precise potentials for both the algorithms. The empirically analysis have shown that the new methods based on linear programming are clearly more accurate than other approaches for evaluating IDs with intervals.

13.1 List of publications

The different studies included in this dissertation have been presented in the following publications:

Publications in International Journals

R. Cabañas, A. Antonucci, A. Cano, and M Gómez-Olmedo. Evaluating interval-valued influence diagrams. *International Journal of Approximate Reasoning*, 80:393–411, 2017

R. Cabañas, A. Cano, M. Gómez-Olmedo, and A. L. Madsen. Improvements to variable elimination and symbolic probabilistic inference for evaluating influence diagrams. *International Journal of Approximate Reasoning*, 70:13–35, 2016

R. Cabañas, M. Gómez-Olmedo, and A. Cano. Using binary trees for the evaluation of influence diagrams. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 24(01):59–89, 2016

Publications in International Conferences and Workshops

R. Cabañas, A. Antonucci, A. Cano, and M. Gómez-Olmedo. Variable elimination for interval-valued influence diagrams. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty: 13th European Conference, ECSQARU 2015, Compiègne, France, July 15-17, 2015. Proceedings*, volume 9161 LNAI, pages 541–551. Springer, 2015

R. Cabañas, A. Cano, M. Gómez-Olmedo, and A. L. Madsen. On SPI-lazy evaluation of influence diagrams. In *Probabilistic Graphical Models: 7th European Workshop, PGM 2014, Utrecht, The Netherlands, September 17-19, 2014. Proceedings*, pages 97–112. Springer International Publishing, 2014

R. Cabañas, A. L. Madsen, M. Gómez-Olmedo, and A. Cano. *On SPI for evaluating Influence Diagrams*, pages 506–516. Springer International Publishing, Cham, 2014

R. Cabañas, M. Gómez-Olmedo, and A. Cano. Evaluating asymmetric decision problems with binary constraint trees. In *Symbolic and Quantiative Approaches to Resoning with Uncertainty: 12th European Conference, ECSQARU 2013, Utrecht, The Netherlands, July 8-10, 2013, Proceedings*, volume 7958 LNAI, pages 85–96. Springer, 2013

R. Cabañas, M. Gómez, and A. Cano. Approximate inference in influence diagrams using binary trees. In *Proceedings of the Sixth European Workshop on Probabilistic Graphical Models (PGM-12)*, 2012

Publications in National Conferences

R. Cabañas, A Cano, M Gómez-Olmedo, and A. L. Madsen. Approximate lazy evaluation of influence diagrams. In *Advances in Artificial Intelligence: 15th Conference of the Spanish Association for Artificial Intelligence, CAEPIA 2013, Madrid, September 17-20, 2013, Proceedings*, volume 8109, page 321. Springer, 2013

R. Cabañas, A. Cano, M Gómez-Olmedo, and A.L. Madsen. Heuristics for determining the elimination ordering in the influence diagram evaluation with binary trees. In *Twelfth Scandinavian Conference on Artificial Intelligence: SCAI 2013*, volume 257 of *Frontiers in Artificial Intelligence and Applications*, pages 65–74. IOS Press, 2013

13.2 Future work

This last section attempts to provide a general overview of the comments regarding future lines of research.

Concerning to the use of BTs, we can study alternative ways of applying the pruning operation during the evaluation. In the approach presented here, all the BTs are pruned before the evaluation. Instead, we might consider applying an online operation that only prunes the BTs involved in a specific operation if this is extremely costly. In doing so, we might minimize the error of the approximation as the pruning is only done for a few potentials. We could also consider using data structures representing a full model such as RPTs for evaluating IDs.

In relation to the evaluation of asymmetric decision problems, we shall study the behaviour of BTs with constraints using alternatives to the VE inference algorithm, like *Arc Reversal* [102], *Lazy propagation* [79], etc.

As presented in this dissertation, the SPI algorithm only considers next pair of potentials to combine. Thus, a line of future research could be studying the behaviour of the algorithm using a higher neighbourhood degree. It could also be interesting to make a study that let us to characterize which features of an ID make it for a possible candidate for choosing SPI over VE.

Although the interval-valued potentials have been shown as an efficient alternative for modelling imprecision in IDs, we could intend to extend this formalism to more general imprecise frameworks, e.g., credal sets represented by extreme points or generic linear constraints. This should affect the computational complexity of evaluation process, thus making necessary the development of specific approximate algorithms. One possible solution could be using BTs for representing imprecise potentials.

Appendices

Appendix A

Proof of Proposition 6

Herein we demonstrate the correctness of the alternative expression for computing the information gain given in Proposition 6 (page 113). That is, we show that these expressions are equivalent to the one given in the definition of the information gain (see Definition 33). The proof for the information gain computation using the Kullback Leibler divergence is given in Section A.1. The analogous one for the case of the Euclidean distance is given in Section A.2.

A.1 Information gain computation with Kullback-Leibler divergence

Let t be the candidate node to be expanded. Let $v_1, v_2, \ldots, v_n$ the values in $\phi^{R(\boldsymbol{A}^t)}$. Let t_l and t_r the new children of t. The values in $\phi^{R(\boldsymbol{A}^{t_l})}$ are $vl_1, vl_2, \ldots, vl_{n_l}$ and those consistent with $\phi^{R(\boldsymbol{A}^{t_r})}$ are $vr_1, vr_2, \ldots, vr_{n_r}$. Then, the Kullback Leibler divergence between each intermediate tree to the real potential can be written as follows.

$$D_{KL}(\phi, \mathcal{BT}_j) = \sum_{i=1}^{n} v_i \cdot \log \frac{v_i}{\frac{\sum_{i=1}^{n} v_i}{n}} = \left(\sum_{i=1}^{n} v_i\right) \cdot \log \frac{n}{\sum_{i=1}^{n} v_i} + \sum_{i=1}^{n} v_i \log v_i \tag{A.1}$$

$$
\begin{aligned}
&D_{KL}(\phi, \mathcal{BT}_{j+1}(t, X_i, \Omega^{t_l}_{X_i}, \Omega^{t_l}_{X_i})) = \\
&\qquad = \sum_{i=1}^{n_l} vl_i \cdot \log \frac{vl_i}{\frac{\sum_{i=1}^{n_l} vl_i}{n_l}} + \sum_{i=1}^{n_r} vr_i \cdot \log \frac{vr_i}{\frac{\sum_{i=1}^{n_l} vr_i}{n_r}} = \\
&\qquad = \left(\sum_{i=1}^{n_l} vl_i\right) \cdot \log \frac{n_l}{\sum_{i=1}^{n_l} vl_i} + \sum_{i=1}^{n_l} vl_i \log vl_i + \\
&\qquad + \left(\sum_{i=1}^{n_l} vr_i\right) \cdot \log \frac{n_r}{\sum_{i=1}^{n_l} vr_i} + \sum_{i=1}^{n_r} vr_i \log vl_i
\end{aligned}
\tag{A.2}
$$

Using (A.1) and (A.2), the information gain in Equation (5.1) can be rewritten as follows.

$$
\begin{aligned}
&I_{KL}(t, X_i, \Omega^{t_l}_{X_i}, \Omega^{t_r}_{X_i}) = D_{KB}(\psi, \mathcal{BT}_j) - D_{KL}(\psi, \mathcal{BT}_{j+1}(t, X_i, \Omega^{t_l}_{X_i}, \Omega^{t_l}_{X_i})) = \\
&\qquad = \left(\sum_{i=1}^{n} v_i\right) \cdot \log \frac{n}{\sum_{i=1}^{n} v_i} + \sum_{i=1}^{n} v_i \log v_i \\
&\qquad - \left(\sum_{i=1}^{n_l} vl_i\right) \cdot \log \frac{n_l}{\sum_{i=1}^{n_l} vl_i} + \sum_{i=1}^{n_l} vl_i \log vl_i \\
&\qquad - \left(\sum_{i=1}^{n_l} vr_i\right) \cdot \log \frac{n_r}{\sum_{i=1}^{n_l} vr_i} + \sum_{i=1}^{n_r} vr_i \log vl_i
\end{aligned}
\tag{A.3}
$$

It holds that $\frac{\Omega^{t_l}_{X_i}}{\Omega^{t}_{X_i}} = \frac{n_l}{n}$ and $\frac{\Omega^{t_r}_{X_i}}{\Omega^{t}_{X_i}} = \frac{n_r}{n}$. Then, we obtain:

$$\begin{aligned} I_{KL}(t, X_i, \Omega^{t_l}_{X_i}, \Omega^{t_r}_{X_i}) &= \left(\sum_{i=1}^{n} v_i\right) \cdot \log \frac{\left|\Omega^{t}_{X_i}\right|}{\sum_{i=1}^{n} v_i} + \\ &+ \left(\sum_{i=1}^{n_l} vl_i\right) \cdot \log \frac{\sum_{i=1}^{n_l} vl_i}{\left|\Omega^{t_l}_{X_i}\right|} + \\ &+ \left(\sum_{i=1}^{n_l} vr_i\right) \cdot \log \frac{\sum_{i=1}^{n_l} vr_i}{\left|\Omega^{t_r}_{X_i}\right|} \end{aligned} \tag{A.4}$$

Finally, using the notation $sum(\bullet)$, we obtain:

$$\begin{aligned} I_{KL}(t, X_i, \Omega^{t_l}_{X_i}, \Omega^{t_r}_{X_i}) = sum(\phi^{R(\boldsymbol{A}^{t})}) \cdot \log \frac{\left|\Omega^{t}_{X_i}\right|}{sum(\phi^{R(\boldsymbol{A}^{t})})} \\ + sum(\phi^{R(\boldsymbol{A}^{t_l})}) \cdot \log \frac{sum(\phi^{R(\boldsymbol{A}^{t_l})})}{\left|\Omega^{t_l}_{X_i}\right|} \\ + sum(\phi^{R(\boldsymbol{A}^{t_r})}) \cdot \log \frac{sum(\phi^{R(\boldsymbol{A}^{t_r})})}{\left|\Omega^{t_r}_{X_i}\right|} \end{aligned} \tag{A.5}$$

which is the expression given in Equation (5.4). ■

A.2 Information gain computation with Euclidean distance

Let t be the candidate node to be expanded. Let $v_1, v_2, \ldots, v_n$ the values in $\psi^{R(\boldsymbol{A}^{t})}$. Let t_l and t_r the new children of t. The values in $\psi^{R(\boldsymbol{A}^{t_l})}$ are $vl_1, vl_2, \ldots, vl_{n_l}$ and those consistent with $\psi^{R(\boldsymbol{A}^{t_r})}$ are $vr_1, vr_2, \ldots, vr_{n_r}$. Then, the information gain in Equation (5.1) can be rewritten as follows.

$$I(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) = D(\psi, \mathcal{BT}_j) - D(\psi, \mathcal{BT}_{j+1}(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_l})) =$$
$$= \sqrt{\sum_{i=1}^{n}(v_i - \frac{\sum_{i=1}^{n} v_i}{n})^2} - \sqrt{\sum_{i=1}^{n_l}(vl_i - \frac{\sum_{i=1}^{n_r} vl_i}{n_l})^2 + \sum_{i=1}^{n_r}(vr_i - \frac{\sum_{i=1}^{n_r} vr_i}{n_r})^2} \tag{A.6}$$

The variance σ^2 of $v_1, v_2, \ldots, v_n$ is:

$$\sigma^2 = \frac{\sum_{i=1}^{n}(v_i - \frac{\sum_{i=1}^{n} v_i}{n})^2}{n} \Leftrightarrow \sigma^2 \cdot n = \sum_{i=1}^{n}(v_i - \frac{\sum_{i=1}^{n} v_i}{n})^2 \tag{A.7}$$

Similarly, the variance σ_l^2 of $vl_1, vl_2, \ldots, vl_{n_l}$ and the variance σ_r^2 of $vr_1, vr_2, \ldots, vr_{n_r}$ are:

$$\sigma_l^2 = \frac{\sum_{i=1}^{n_l}(v_{li} - \frac{\sum_{i=1}^{n_l} vl_i}{n_l})^2}{n_l} \Leftrightarrow \sigma_l^2 \cdot n_l = \sum_{i=1}^{n_l}(vl_i - \frac{\sum_{i=1}^{n_l} vl_i}{n_l})^2 \tag{A.8}$$

$$\sigma_r^2 = \frac{\sum_{i=1}^{n_r}(v_{ri} - \frac{\sum_{i=1}^{n_r} vr_i}{n_r})^2}{n_r} \Leftrightarrow \sigma_r^2 \cdot n_r = \sum_{i=1}^{n_r}(vr_i - \frac{\sum_{i=1}^{n_r} vr_i}{n_r})^2 \tag{A.9}$$

Using Equations (A.7), (A.8), and (A.9), the information gain in Equation (A.6) can be written as:

$$I(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) = \sqrt{\sigma^2 \cdot n} - \sqrt{\sigma_l^2 \cdot n_l + \sigma_r^2 \cdot n_r} \tag{A.10}$$

Since the variance of a variable is the mean of the square variable minus the square of the mean, then:

$$I(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) = \sqrt{\left(\frac{\sum_{i=1}^{n} v_i^2}{n} - \left(\frac{\sum_{i=1}^{n} v_i}{n}\right)^2\right) \cdot n}$$
$$- \sqrt{\left(\frac{\sum_{i=1}^{n_l} vl_i^2}{n_l} - \left(\frac{\sum_{i=1}^{n_l} vl_i}{n_l}\right)^2\right) \cdot n_l + \left(\frac{\sum_{i=1}^{n_r} vr_i^2}{n_r} - \left(\frac{\sum_{i=1}^{n_r} vr_i}{n_r}\right)^2\right) \cdot n_r} \tag{A.11}$$

Finally, simplifying the following expression is obtained:

$$I(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) = \sqrt{\sum_{i=1}^{n} v_i^2 - \frac{\left(\sum_{i=1}^{n} v_i\right)^2}{n}} - \sqrt{\sum_{i=1}^{n_l} vl_i^2 - \frac{\left(\sum_{i=1}^{n_l} vl_i\right)^2}{n_l} + \sum_{i=1}^{n_r} vr_i^2 - \frac{\left(\sum_{i=1}^{n_r} vr_i\right)^2}{n_r}} \tag{A.12}$$

which is the expression given in Equation (5.5). ■

Appendix B

Additional Information about the Experimental Work

B.1 Code and system details

All the experimental work described in this dissertation has been performed using the Elvira software[1], which is an open-source tool for modelling PGMs. All the algorithms here described have been coded in Java. A Github repository including the source code and IDs (in Elvira format) used in the experimentation has been created [2]. All the experiments were run in a Intel Core i7 with 2.0 GHz and 8GB of memory. Additionally, the algorithms explained in Chapter 10 involved the solution of linear programming tasks. For that, lp_solve[3] 5.5 was used. This is a GNU library coded in C for solving linear (integer) programming problems based on the revised simplex method and the Branch-and-bound method for the integers.

[1] `http://leo.ugr.es/elvira/`

[2] `https://github.com/rcabanasdepaz/bayelvira`

[3] `http://lpsolve.sourceforge.net`

B.2 Influence diagrams details

Herein we give a brief description of all the IDs used in the experiments of this dissertation. In particular, for each ID, we show its graph and the details of each variable (i.e., type and domain).

Appendicitis

This is an example of a simple ID modelling a medical decision problem [69].

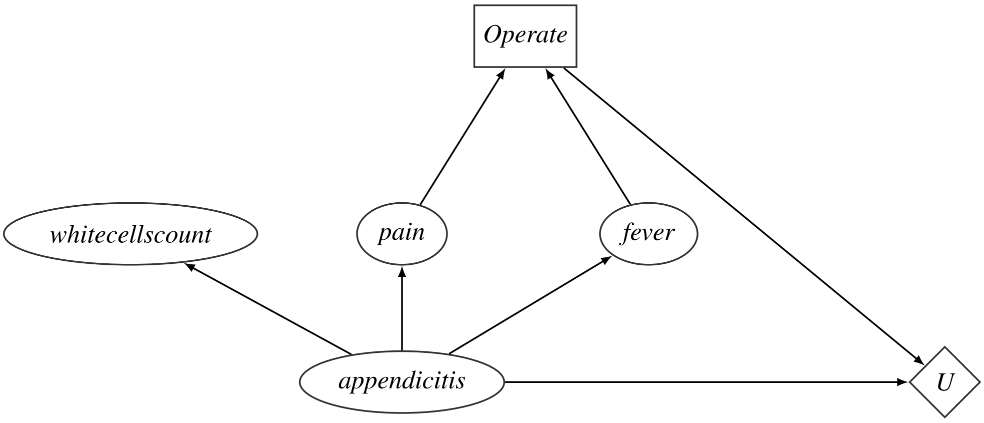


Figure B.1: Graph of the Appendicitis ID.

name	type	states
whitecellscount	chance	*normal, high*
appendicitis	chance	*false, true*
fever	chance	*false, true*
pain	chance	*false, true*
Operate	decision	*now, wait*

Table B.1: Details of each variable in the Appendicitis ID.

Car Buyer

This ID models the used car buyer problem, which was proposed by Qi et al. [94] as an example of an ID representing an asymmetric decision problem.

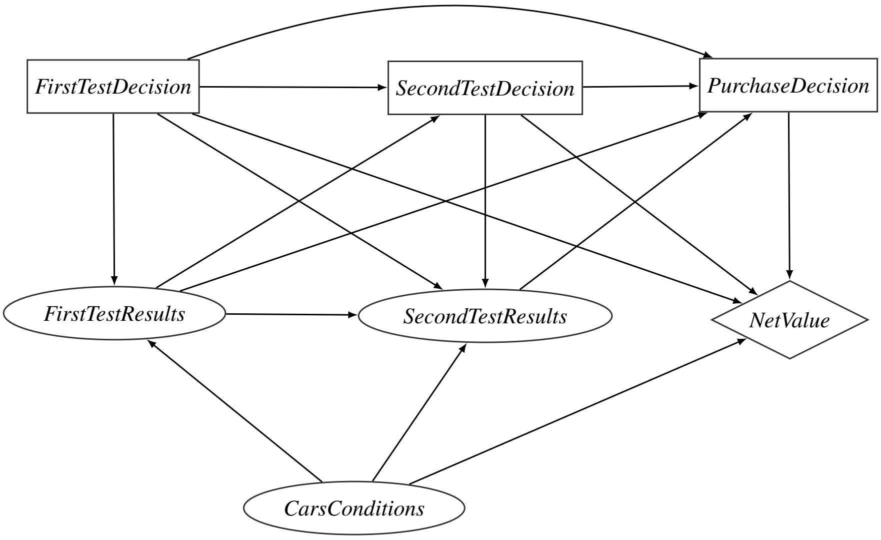


Figure B.2: Graph of the Car Buyer ID.

name	type	states
CarsConditions	chance	*Peach, Lemon*
SecondTestResults	chance	*NoResults, Defects0, Defects1*
FirstTestResults	chance	*NoResults, Defects0, Defects1, Defects2*
PurchaseDecision	decision	*No, Yes*
SecondTestDecision	decision	*NoTest, Differential*
FirstTestDecision	decision	*NoTest, Steering, Transmission, FuelElect*
A	chance	*present, absent*

Table B.2: Details of each variable in the Car Buyer ID.

ChestClinic

This is an ID [55] obtained from the well-known Asia BN, originally due to Lauritzen and Spiegelhalter [73].

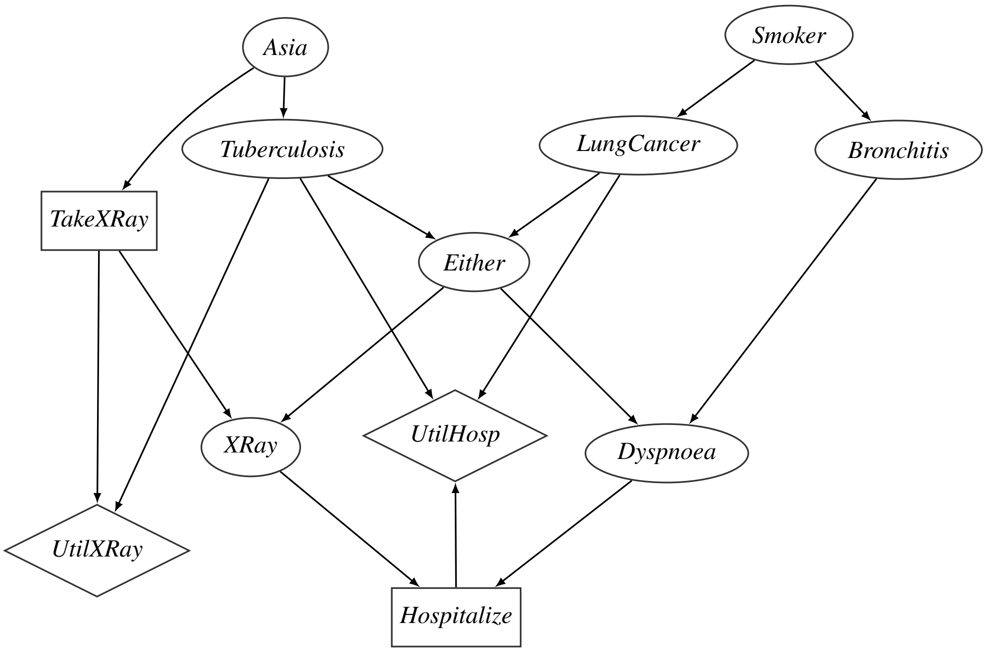


Figure B.3: Graph of the Chest Clinic ID.

name	type	states
Asia	chance	*yes, no*
Smoker	chance	*yes, no*
Tuberculosis	chance	*yes, no*
LungCancer	chance	*yes, no*
Bronchitis	chance	*yes, no*
Either	chance	*yes, no*
Dyspnoea	chance	*yes, no*
TakeXRay	decision	*yes, no*
XRay	chance	*positive, negative*
Hospitalize	decision	*yes, no*

Table B.3: Details of each variable in the Chest Clinic ID.

Competitive Asymmetries

This ID is one of the networks developed for teaching MBA students competitive analysis, at New York University Stern School of Business [57].

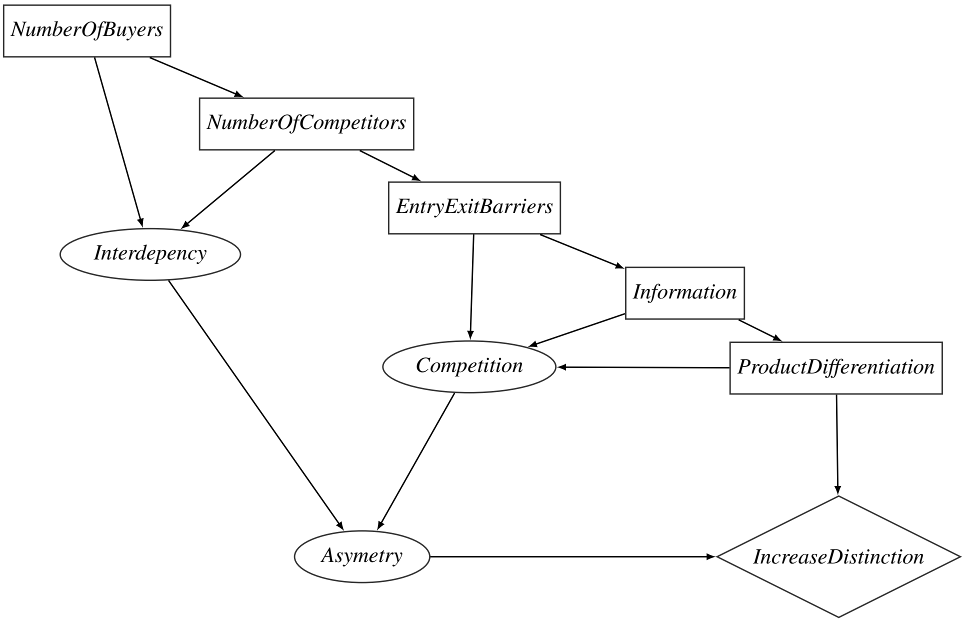


Figure B.4: Graph of the Competitive Asymmetries ID.

name	type	states
Interdepency	chance	*Domination, Symbiosis, Dependancy, FreePlayers*
NumberOfBuyers	decision	*One, Few, Many*
Asymetry	chance	*High, Moderate, None*
Information	decision	*Proprietary, Restricted, PerfectAvaibility*
ProductDifferentiation	decision	*Extensive, Moderate, Low, None*
EntryExitBarriers	decision	*Legal, High, Significant, None*
Competition	chance	*Intense, Limited*
NumberOfCompetitors	decision	*One, Two, Few, Many*

Table B.4: Details of each variable in the Competitive Asymmetries ID.

Dating

This ID represents the dating decision problem [71].

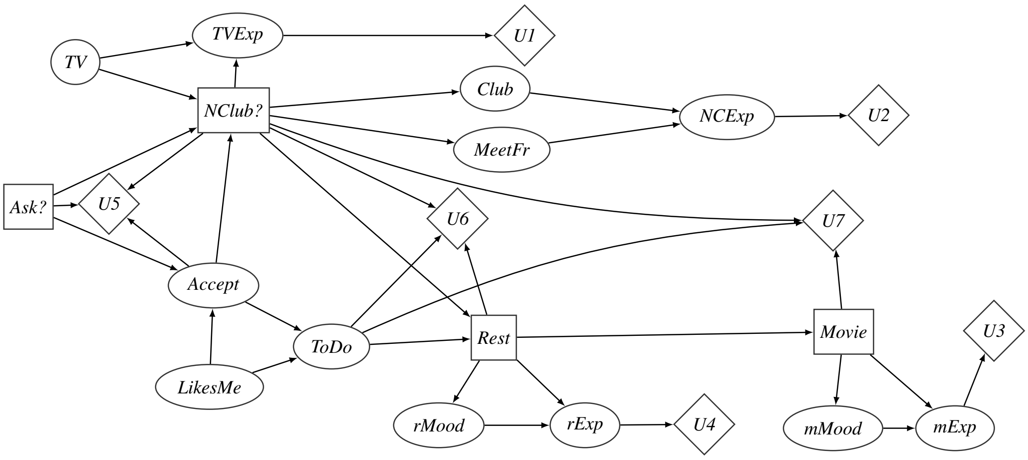


Figure B.5: Graph of the Dating ID.

name	type	states
Accept	chance	*yes, no, noresp*
ToDo	chance	*movie, restaurant, nopref*
LikesMe	chance	*yes, no*
TV	chance	*good, bad*
TVExp	chance	*positive, negative, unknown*
Club	chance	*positive, negative, unknown*
MeetFr	chance	*positive, negative, unknown*
NCExp	chance	*positive, negative, unknown*
mMood	chance	*good, bad, unknown*
mExp	chance	*positive, negative, unknown*
rMood	chance	*good, bad, unknown*
rExp	chance	*positive, negative, unknown*
Ask?	decision	*yes, no*
NClub?	decision	*yes, no, nodec*
Movie	decision	*action, romantic, nodec*
Rest	decision	*expensive, cheap, nodec*

Table B.5: Details of each variable in the Dating ID.

Diabetes

This ID models the problem of diagnosing diabetes [65].

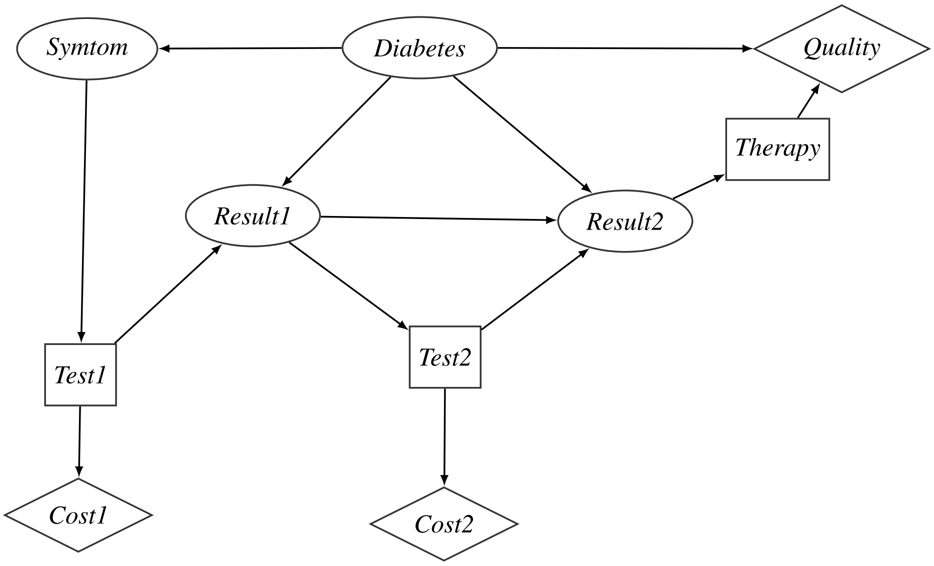


Figure B.6: Graph of the Diabetes ID.

name	type	states
Symtom	chance	*negative, positive*
Diabetes	chance	*absent, present*
Result1	chance	*bloodpositive, bloodnegative, urinepositive, urinenegative, nores*
Result2	chance	*bloodpositive, bloodnegative, urinepositive, urinenegative, nores*
Test1	decision	*notest, urine, blood*
Test2	decision	*notest, urine, blood*
Therapy	decision	*no, yes*

Table B.6: Details of each variable in the Diabetes ID.

Jaundice

This is a real-world ID for neonatal jaundice management [97].

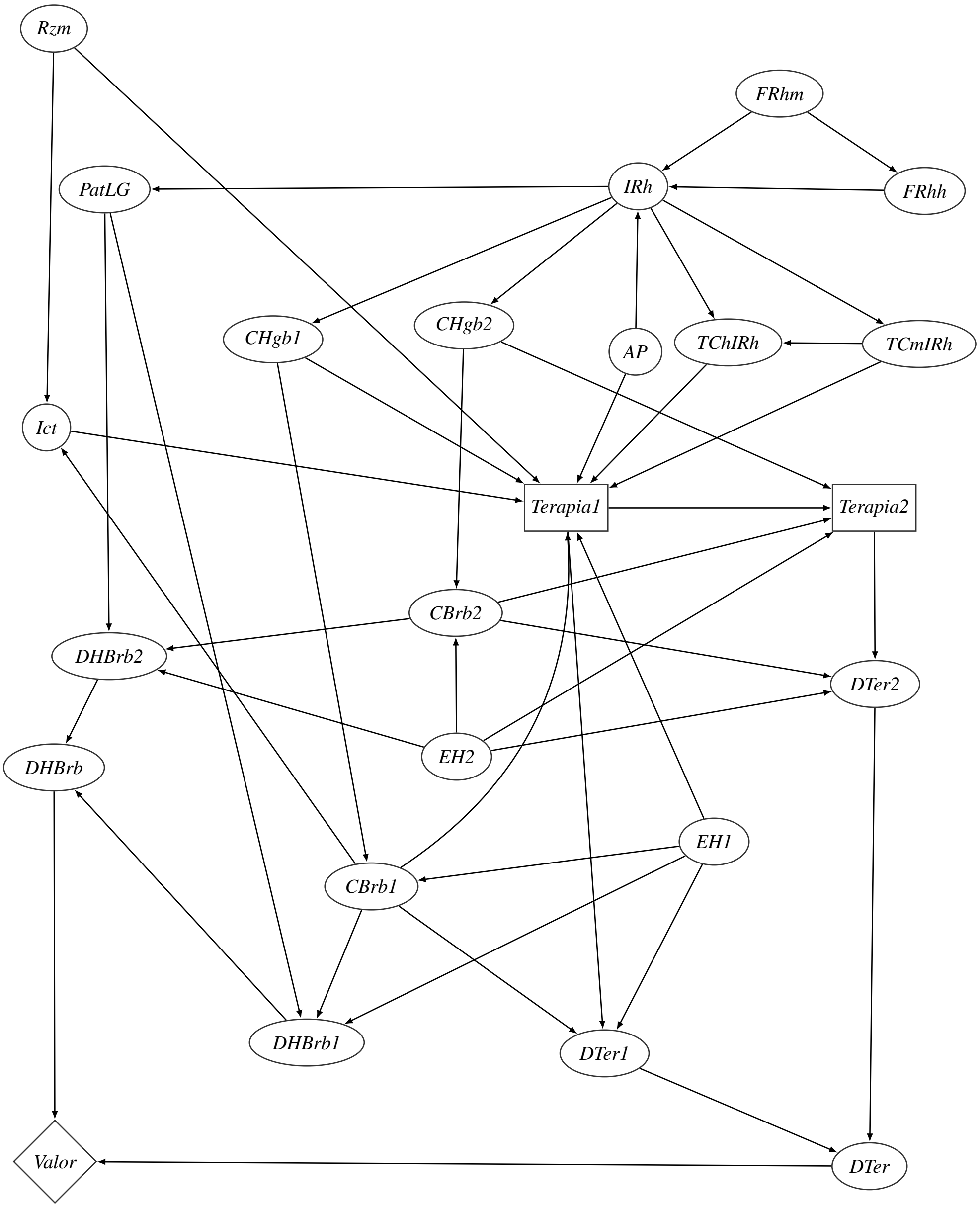


Figure B.7: Graph of the Jaundice ID.

name	type	states
DHBrb	chance	*NULO, BAJO, MEDIO, ALTO, MUYALTO, EXTREMO*
DHBrb2	chance	*NULO, BAJO, MEDIO, ALTO, MUYALTO, EXTREMO*
DHBrb1	chance	*NULO, BAJO, MEDIO, ALTO, MUYALTO, EXTREMO*
DTer	chance	*NULO, BAJO, MEDIO, ALTO, MUYALTO*
DTer2	chance	*NULO, BAJO, MEDIO, ALTO, MUYALTO*
DTer1	chance	*NULO, BAJO, MEDIO, ALTO, MUYALTO*
PatLG	chance	*Leve, Grave, MuyGrave*
IRh	chance	*Ausente, Presente*
CHgb2	chance	*Baja, Media, Alta*
CBrb2	chance	*Normal, Alta*
EH2	chance	*menosde24horas, masde24horas*
CHgb1	chance	*Baja, Media, Alta*
CBrb1	chance	*Normal, Alta*
EH1	chance	*menosde12horas, masde12horas*
Rzm	chance	*Negra, Caucasica, Asiatica, Gitana*
AP	chance	*Primeriza, Multipara*
Ict	chance	*Normal, Amarillo, Pies, Calabaza*
TCmIRh	chance	*Negativo, Positivo*
TChIRh	chance	*Negativo, Positivo*
FRhm	chance	*NEGATIVO, POSITIVO*
FRhh	chance	*NEGATIVO, POSITIVO*
Terapia2	decision	*TerNo, ObsAlt, Obs6, Fot6, Fot12, Fot24, Fot6ExaFot6, Fot6ExaFot12, ObsTer*
Terapia1	decision	*TerNo, Fot6, Fot12, Fot24, ObsTer*

Table B.7: Details of each variable in the Jaundice ID.

Jensen et al. 1

This is a synthetic ID used by Jensen et al. [64, page 226] to illustrate some properties of the IDs. Similarly, in this dissertation, this ID was used for illustrating the d-separation in ID (page 70).

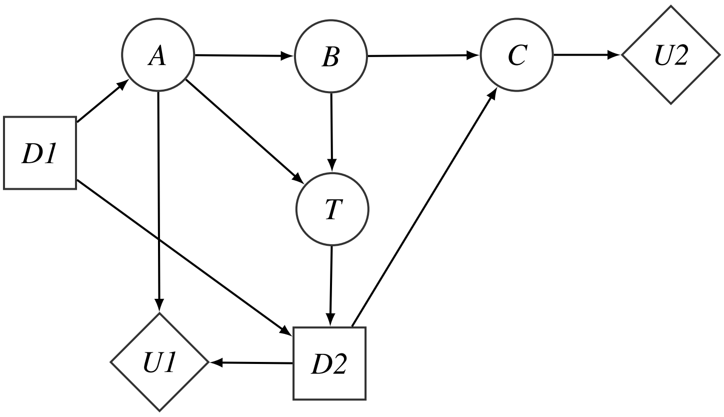


Figure B.8: Graph of the Jensen et al. 1 ID.

name	type	states
A	chance	*present, absent*
B	chance	*present, absent*
C	chance	*present, absent*
T	chance	*present, absent*
D1	decision	*yes, no*
D2	decision	*yes, no*

Table B.8: Details of each variable in the Jensen et al. 1 ID.

Jensen et al. 2

This is a synthetic ID used by Jensen et al. [64, page 141] to illustrate some properties of the IDs. Similarly, in this dissertation, this ID was used for illustrating the minimalization of an ID (page 72).

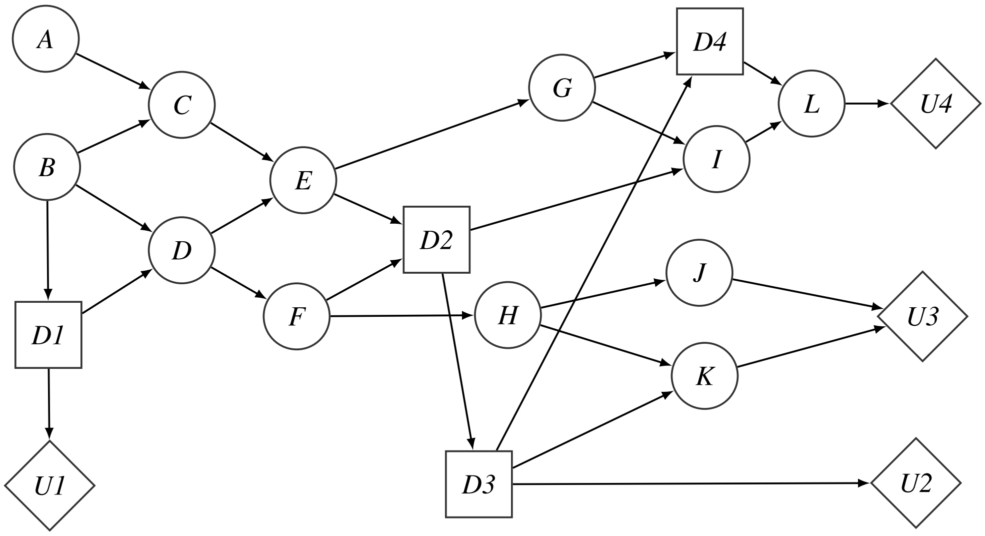


Figure B.9: Graph of the Jensen et al. 2 ID.

name	type	states	name	type	states
D1	decision	*yes, no*	*E*	chance	*present, absent*
D2	decision	*yes, no*	*F*	chance	*present, absent*
D3	decision	*yes, no*	*G*	chance	*present, absent*
D4	decision	*yes, no*	*H*	chance	*present, absent*
A	chance	*present, absent*	*I*	chance	*present, absent*
B	chance	*present, absent*	*J*	chance	*present, absent*
C	chance	*present, absent*	*K*	chance	*present, absent*
D	chance	*present, absent*	*L*	chance	*present, absent*

Table B.9: Details of each variable in the Jensen et al. 2 ID.

Maze

This is an ID for solving the problem of finding a path in a maze [113].

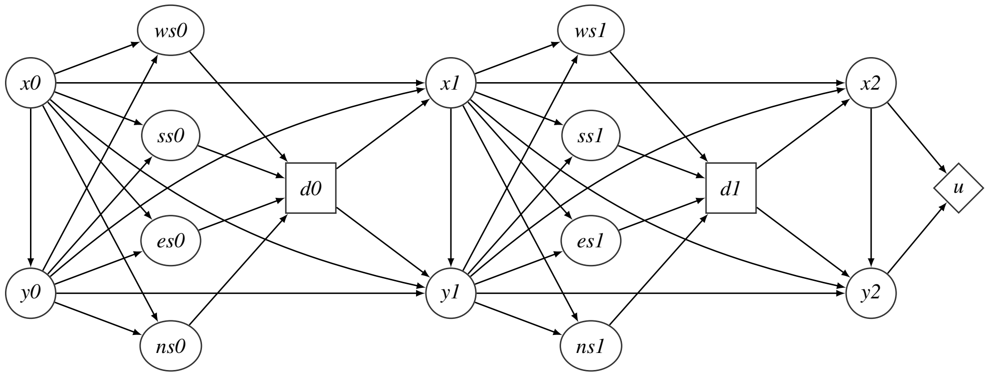


Figure B.10: Graph of the Maze ID.

name	type	states
x2	chance	*x0, x1, x2, x3, x4, x5, x6, x7, x8*
y2	chance	*y0, y1, y2, y3, y4, y5, y6*
d1	decision	*North, East, South, West, Notmove*
ns1	chance	*0, 1*
es1	chance	*0, 1*
ss1	chance	*0, 1*
ws1	chance	*0, 1*
x1	chance	*x0, x1, x2, x3, x4, x5, x6, x7, x8*
y1	chance	*y0, y1, y2, y3, y4, y5, y6*
d0	decision	*North, East, South, West, Notmove*
ns0	chance	*0, 1*
es0	chance	*0, 1*
ss0	chance	*0, 1*
ws0	chance	*0, 1*
x0	chance	*x0, x1, x2, x3, x4, x5, x6, x7, x8*
y0	chance	*y0, y1, y2, y3, y4, y5, y6*

Table B.10: Details of each variable in the Maze ID.

Mildew 1

This ID [63] a decision problem in agriculture where a farmer has to decide on a treatment with fungicides for a wheat field.

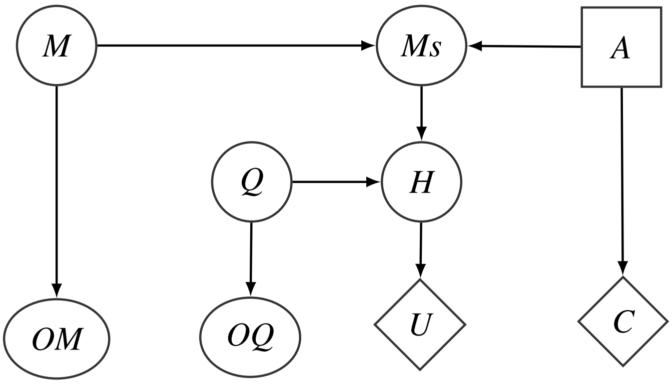


Figure B.11: Graph of the Mildew 1 ID.

name	type	states
Q	chance	*f, a, g, v*
OQ	chance	*f, a, g, v*
H	chance	*r, b, p, f, a, g, v*
M	chance	*no, l, m, s*
Ms	chance	*no, l, m, s*
A	decision	*no, l, m, h*
OM	chance	*no, l, m, s*

Table B.11: Details of each variable in the Mildew 1 ID.

Mildew 4

This is an extended version of the previous ID [63].

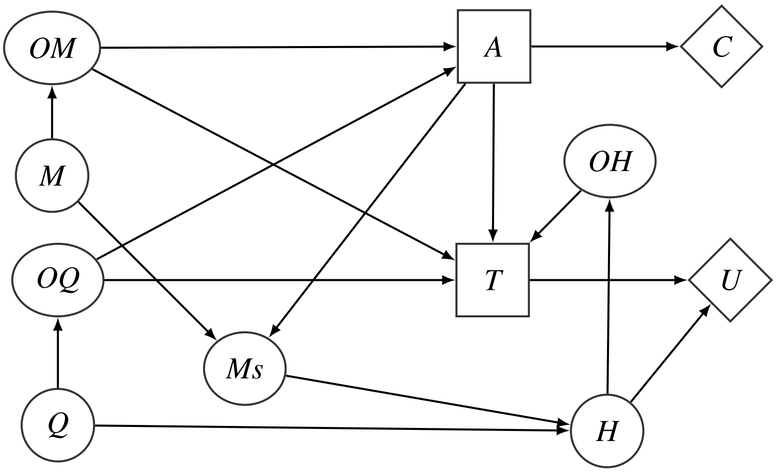


Figure B.12: Graph of the Mildew 4 ID.

name	type	states
OH	chance	*r, b, p, f, a, g, v*
T	decision	*now, wait1week, wait2weeks*
OM	chance	*no, l, m, s*
A	decision	*no, l, m, h*
Ms	chance	*no, l, m, s*
M	chance	*no, l, m, s*
H	chance	*r, b, p, f, a, g, v*
OQ	chance	*f, a, g, v*
Q	chance	*f, a, g, v*

Table B.12: Details of each variable in the Mildew 4 ID.

Motivation ID

This is a synthetic ID was used in this dissertation for motivating the improved evaluation methods proposed in Chapter 12. It was originally proposed at [20].

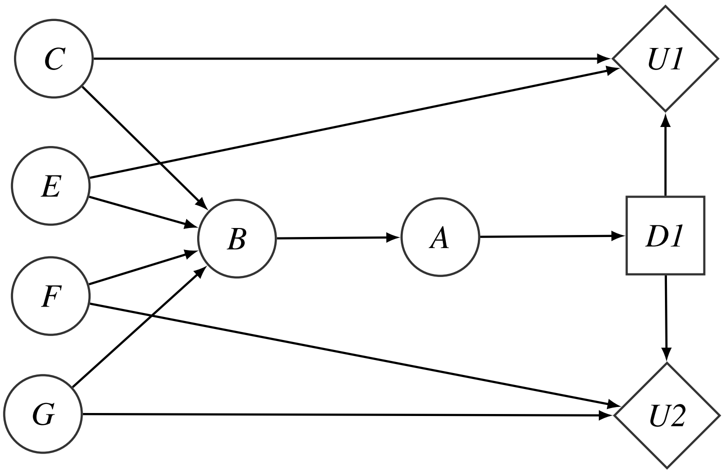


Figure B.13: Graph of the Motivation ID.

name	type	states
A	chance	*a1, a2, a3, a4, a5*
B	chance	*present, absent*
C	chance	*present, absent*
E	chance	*high, medium, low*
F	chance	*f1, f3, f3, f4*
G	chance	*high, medium, low*
D1	decision	*maybe, yes, no*

Table B.13: Details of each variable in the Motivation ID.

NHL

This a real world ID used in medicine for the management of primary gastric non-Hodgkin lymphoma [76].

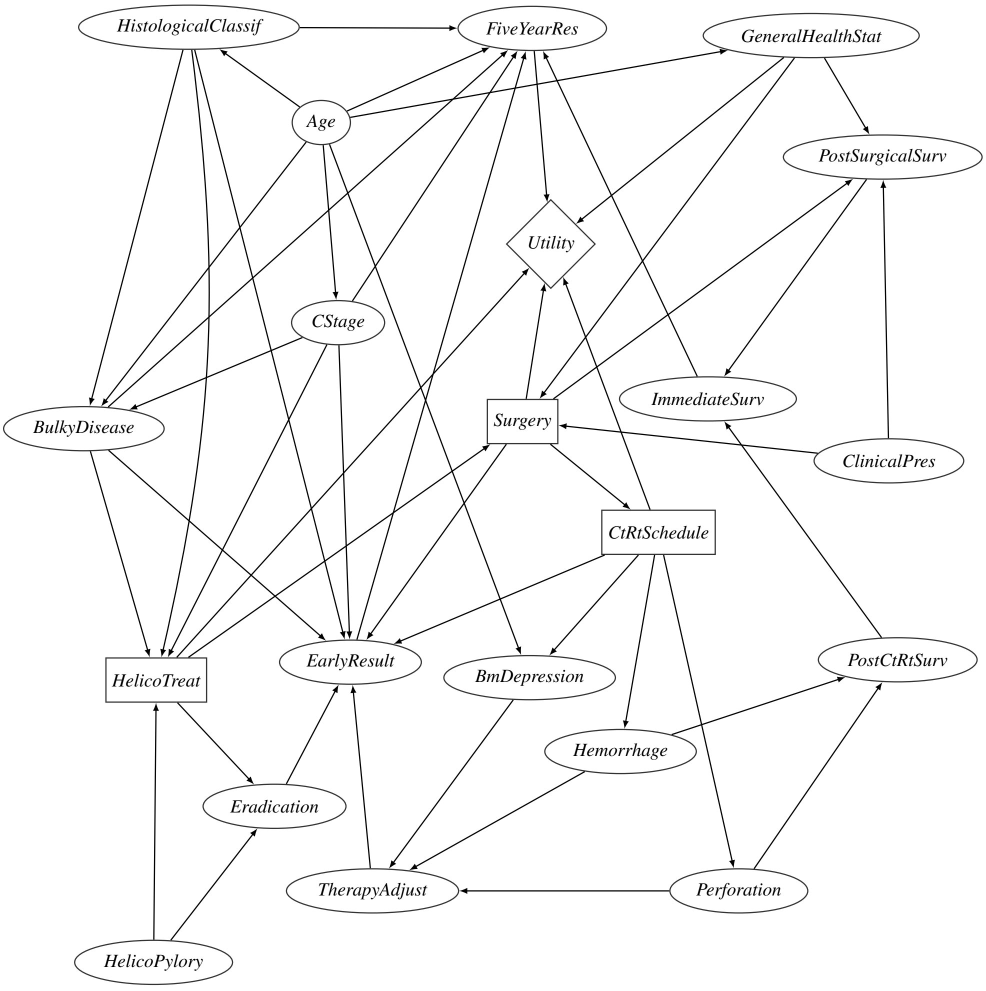


Figure B.14: Graph of the NHL ID.

name	type	states
FiveYearRes	chance	*ALIVE, DEATH*
EarlyResult	chance	*CR, PR, NC, PD*
ImmediateSurv	chance	*NO, YES*
PostSurgicalSurv	chance	*NO, YES*
PostCtRtSurv	chance	*NO, YES*
TherapyAdjust	chance	*NO, YES*
Hemorrhage	chance	*NO, YES*
Perforation	chance	*NO, YES*
BmDepression	chance	*NO, YES*
Eradication	chance	*NO, YES*
HelicoPylory	chance	*ABSENT, PRESENT*
HistologicalClassif	chance	*LOWGRADE, HIGHGRADE*
GeneralHealthStat	chance	*POOR, AVERAGE, GOOD*
ClinicalPres	chance	*NONE, HEMORRHAGE, PERFORATION, OBSTRUCTION*
CStage	chance	*I, II1, II2, III, IV*
BulkyDisease	chance	*YES, NO*
Age	chance	*v1019, v2029, v3039, v4044, v4549, v5054, v5559, v6064, v6569, v7079, v8089, GE90*
CtRtSchedule	decision	*NONE, RadioTherapy, ChemoTherapy, ChemoTherapyNEXTRadioTherapy*
Surgery	decision	*NONE, CURATIVE, PALLIATIVE*
HelicoTreat	decision	*NO, YES*

Table B.14: Details of each variable in the NHL ID.

Oil

This ID represents the oil wildcatter's decision problem [96, 105]. In this dissertation, this ID was used to illustrate the basic concepts about IDs (page 55). Yet, here the full names of the variables and states are shown (e.g., 'empty' instead of 'e').

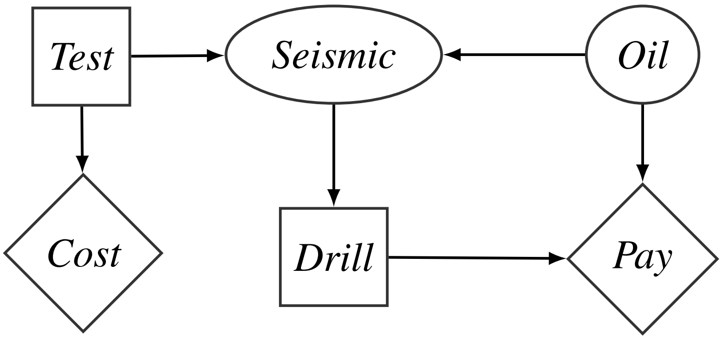


Figure B.15: Graph of the Oil ID.

name	type	states
Oil	chance	*empty, wet, soak*
Seismic	chance	*closed, open, diff*
Drill	decision	*drill, notDrill*
Test	decision	*test, notTest*

Table B.15: Details of each variable in the Oil ID.

Oil Split Costs

This ID is equivalent to the previous one, but with an additional utility node representing the cost of drilling. In the original diagram, this cost is included in the utility potential associated to the node Pay.

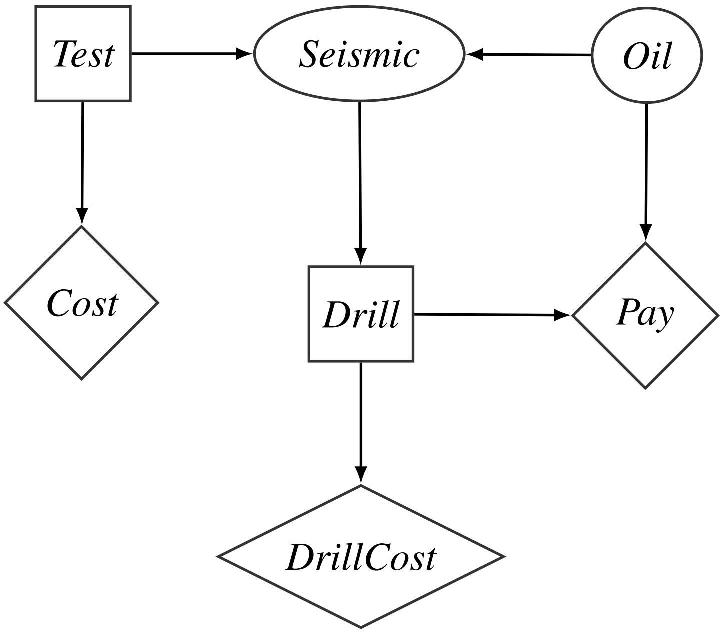


Figure B.16: Graph of the Oil Split Costs ID.

name	type	states
Oil	chance	*empty, wet, soak*
Seismic	chance	*closed, open, diff*
Drill	decision	*drill, notDrill*
Test	decision	*test, notTest*

Table B.16: Details of each variable in the Oil Split Costs ID.

Poker

This is an ID representing the decision problem in the poker game [64, page 111].

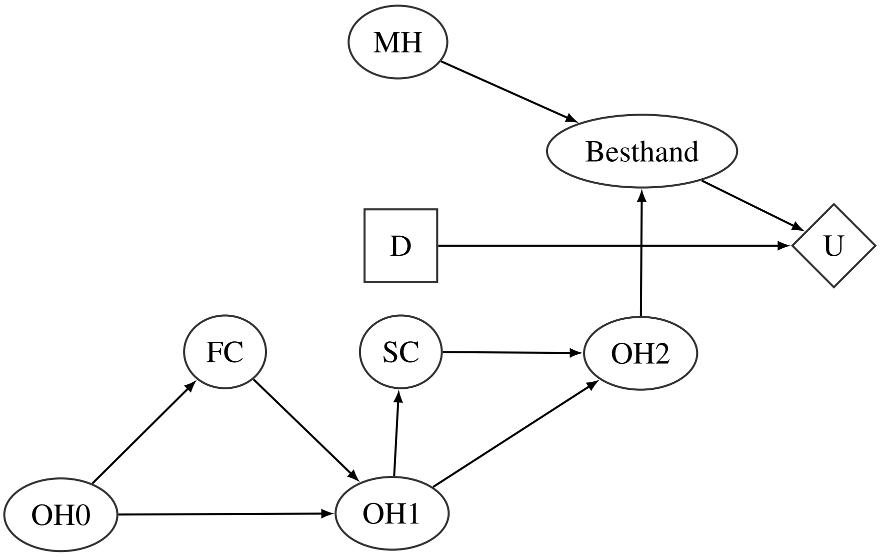


Figure B.17: Graph of the Poker ID.

name	type	states
Besthand	chance	*myhand, opponent*
OH2	chance	*no, 1a, 2v, 2a, fl, st, 3v, stfl*
MH	chance	*no, 1a, 2v, 2a, fl, st, 3v, stfl*
OH1	chance	*no, 1a, 2cons, 2s, 2v, fl, st, 3v, stfl*
SC	chance	*0Changed, 1Changed, 2Changed*
OH0	chance	*no, 1a, 2cons, 2s, 2v, fl, st, 3v, stfl*
FC	chance	*ochanged, 1changed, 2changed, 3changed*
D	decision	*yes, no*

Table B.17: Details of each variable in the Poker ID.

Poker Extended

This is an extended version of the previous ID representing the decision problem in the poker game [64, page 137].

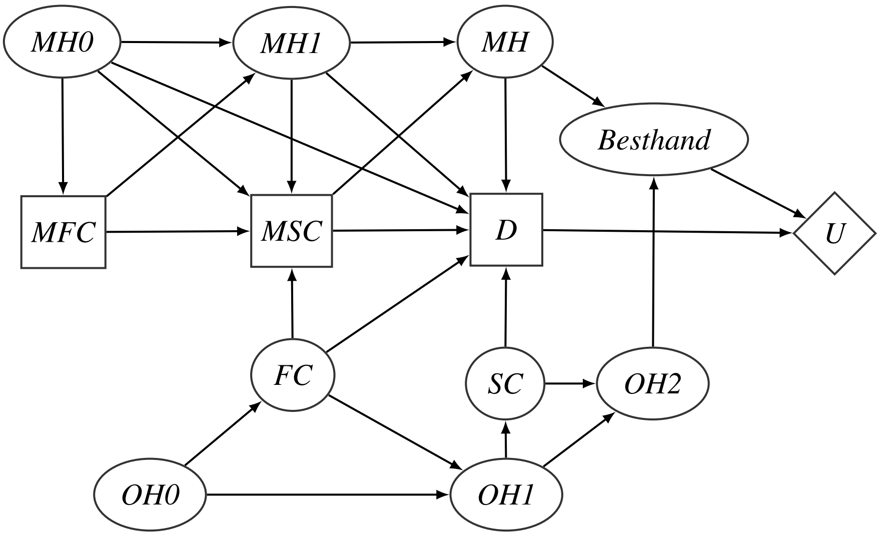


Figure B.18: Graph of the Poker Extended ID.

name	type	states
MSC	decision	*0changed, 1changed, 2changed*
MFC	decision	*0changed, 1changed, 2changed, 3changed*
MH0	chance	*no, 1a, 2cons, 2s, 2v, fl, st, 3v, stfl*
MH1	chance	*no, 1a, 2cons, 2s, 2v, fl, st, 3v, stfl*
MH	chance	*no, 1a, 2v, 2a, fl, st, 3v, stfl*
D	decision	*fold, call*
Besthand	chance	*myhand, opponent*
OH2	chance	*no, 1a, 2v, 2a, fl, st, 3v, stfl*
OH1	chance	*no, 1a, 2cons, 2s, 2v, fl, st, 3v, stfl*
SC	chance	*0Changed, 1Changed, 2Changed*
OH0	chance	*no, 1a, 2cons, 2s, 2v, fl, st, 3v, stfl*
FC	chance	*0changed, 1changed, 2changed, 3changed*

Table B.18: Details of each variable in the Poker Extended ID.

Reactor

This is an ID modelling the Reactor decision problem as described by Bielza and Shenoy [3, 4]. In Chapter 6, this ID was used to introduce asymmetric decision problems. Note that the long names for the variables and states have been used.

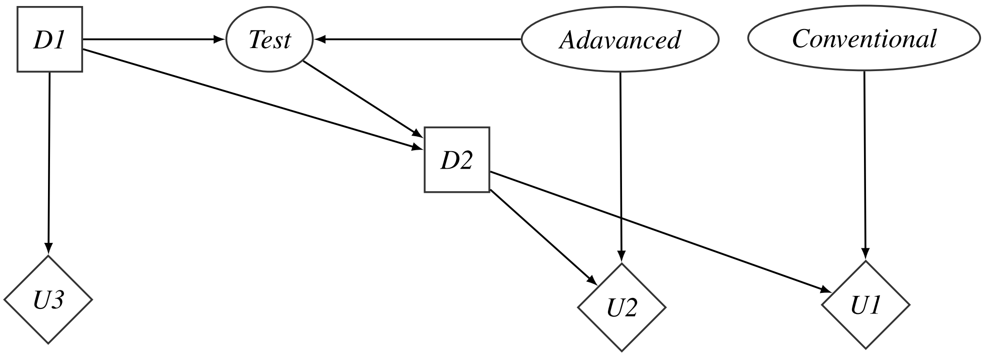


Figure B.19: Graph of the Reactor ID.

name	type	states
Adavanced	chance	*aSuccess, aLimited, aMajor*
Test	chance	*excellent, good, bad, noresult*
Conventional	chance	*cSuccess, cFailure*
D1	decision	*test, notest*
D2	decision	*advanced, conventional, none*

Table B.19: Details of each variable in the Reactor ID.

Thinkbox

This ID models a simplified version of the dice game called "Tænkeboks" [57].

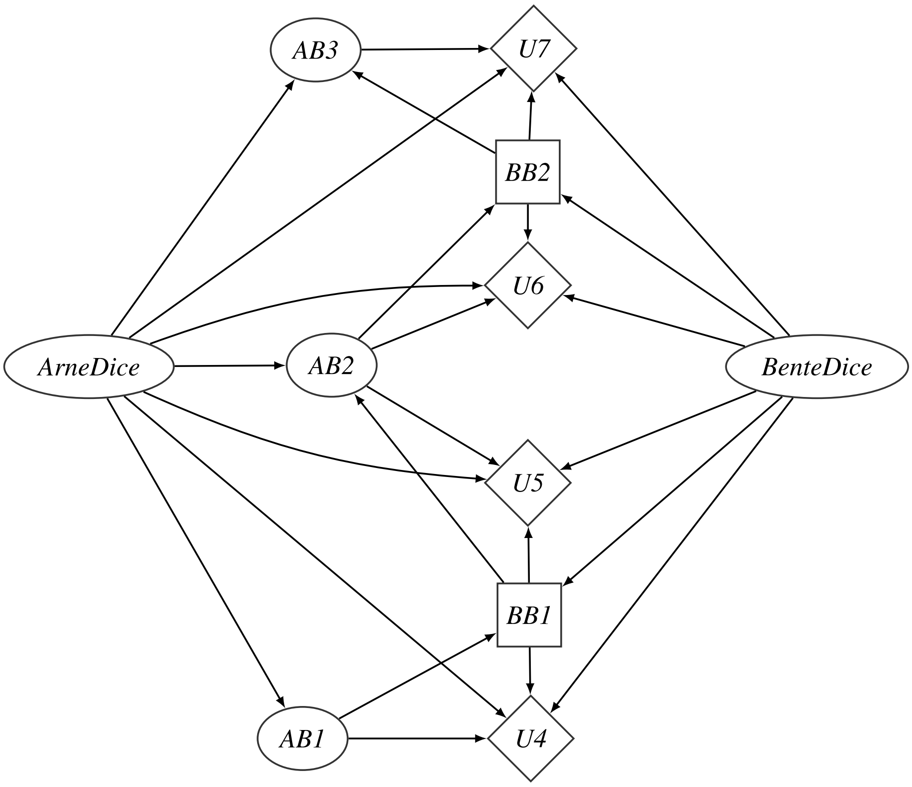


Figure B.20: Graph of the Thinkbox ID.

name	type	states
ArneDice	chance	*v1, v2*
BenteDice	chance	*v1, v2*
AB1	chance	*v1, v2, v2*1, v2*2*
BB1	decision	*v2, v2*1, v2*2, call*
AB2	chance	*v2*1, v2*2, call*
BB2	decision	*v2*2, call*
AB3	chance	*call*

Table B.20: Details of each variable in the Thinkbox ID.

Threat of Entry

This ID is one of the networks developed for teaching MBA students competitive analysis, at New York University Stern School of Business [57].

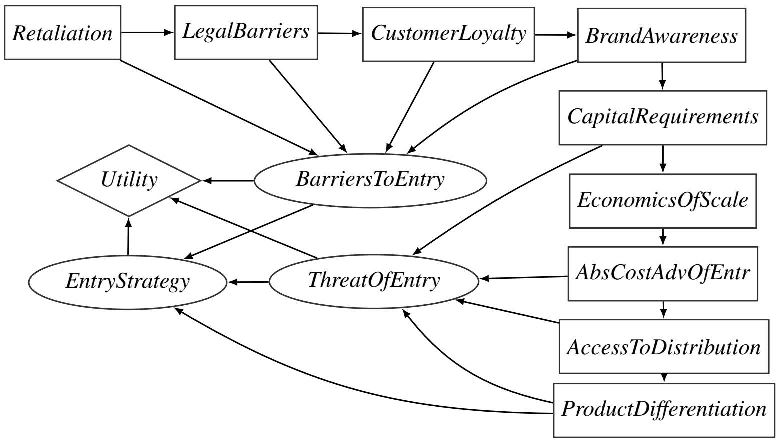


Figure B.21: Graph of the Threat of Entry ID.

name	type	states
EntryStrategy	chance	*Niche, Differentiation, PriceCutting, AbandonEntry*
Retaliation	decision	*HighlyExpected, NotExpected*
LegalBarriers	decision	*GovRegulation, QuasiMonopoly, Few*
AccessToDistribution	decision	*FreeAccess, FixedCostsInCarryingNewProducts, DistributorsPreferenceToEstablishedFirms*
CustomerLoyalty	decision	*Strong, Weak*
BrandAwareness	decision	*Strong, Weak*
BarriersToEntry	chance	*StrongBarriers, WeakBarriers*
ThreatOfEntry	chance	*HightThreat, LowThreat*
ProductDifferentiation	decision	*VeryHigh, Moderate, Low*
AbsCostAdvOfEntr	decision	*Yes, No*
EconomicsOfScale	decision	*Necessary, Unnecessary*
CapitalRequirements	decision	*Low, High*

Table B.21: Details of each variable in the Threat of Entry ID.

Wildlife

This is an ID used to evaluate the population viability of wildlife species [81].

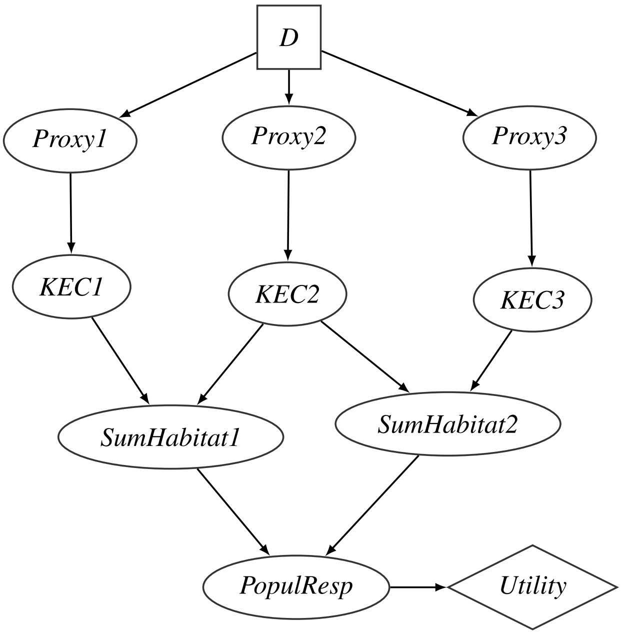


Figure B.22: Graph of the Wildlife ID.

name	type	states
D	decision	*yes, no*
Proxy1	chance	*high, low*
Proxy2	chance	*high, low*
Proxy3	chance	*high, low*
KEC1	chance	*high, low*
KEC2	chance	*high, low*
KEC3	chance	*high, low*
SumHabitat1	chance	*high, low*
SumHabitat2	chance	*high, low*
PopulResp	chance	*high, low*

Table B.22: Details of each variable in the Wildlife ID.

B.3 Constraint rules

For the empirical validation of the proposed method for solving asymmetric decision problems (Section 10.4), a set of six IDs was used. Herein we detail the set of constraint rules of each these IDs.

Car Buyer

$FirstTestDecision \in \{NoTest\} \Leftrightarrow FirstTestResults \in \{NoResults\}$
$FirstTestResults \in \{Defects2\} \Rightarrow FirstTestDecision \in \{FuelElect\}$
$SecondTestDecision \in \{NoTest\} \Leftrightarrow SecondTestResults \in \{NoResults\}$

Dating

$Ask? \in \{no\} \Leftrightarrow Accept \in \{noresp\}$
$Accept \in \{noresp, no\} \Leftrightarrow ToDo \in \{nopref\}$
$Rest \in \{nodec\} \Leftrightarrow rMood \in \{unknown\}$
$Rest \in \{nodec\} \vee rMood \in \{unknown\} \Leftrightarrow rExp \in \{unknown\}$
$Movie \in \{nodec\} \Leftrightarrow mMood \in \{unknown\}$
$Movie \in \{nodec\} \vee mMood \in \{unknown\} \Leftrightarrow mExp \in \{unknown\}$
$NClub? \in \{nodec, yes\} \Leftrightarrow TVExp \in \{unknown\}$
$NClub? \in \{nodec, no\} \Leftrightarrow Club \in \{unknown\}$
$NClub? \in \{nodec, no\} \Leftrightarrow MeetFr \in \{unknown\}$
$Club \in \{unknown\} \vee MeetFr \in \{unknown\} \Leftrightarrow NCExp \in \{unknown\}$

Diabetes

$Test1 \in \{notest\} \Leftrightarrow Result1 \in \{nores\}$
$Test1 \in \{urine\} \Leftrightarrow Result1 \in \{urinepositive, urinenegative\}$
$Test1 \in \{blood\} \Leftrightarrow Result1 \in \{bloodpositive, bloodnegative\}$
$Test2 \in \{notest\} \Leftrightarrow Result2 \in \{nores\}$
$Test2 \in \{urine\} \Leftrightarrow Result2 \in \{urinepositive, urinenegative\}$
$Test2 \in \{blood\} \Leftrightarrow Result2 \in \{bloodpositive, bloodnegative\}$

Maze

$d0 \in \{North\} \Rightarrow d1 \notin \{South\}$
$d1 \in \{North\} \Rightarrow d0 \notin \{South\}$
$d0 \in \{South\} \Rightarrow d1 \notin \{North\}$
$d1 \in \{South\} \Rightarrow d0 \notin \{North\}$
$d0 \in \{East\} \Rightarrow d1 \notin \{West\}$
$d1 \in \{East\} \Rightarrow d0 \notin \{West\}$
$d0 \in \{West\} \Rightarrow d1 \notin \{East\}$
$d1 \in \{West\} \Rightarrow d0 \notin \{East\}$
$x0 \in \{x0\} \Rightarrow x1 \in \{x0, x1\}$
$x0 \in \{x1\} \Rightarrow x1 \in \{x1, x0, x2\}$
$x0 \in \{x2\} \Rightarrow x1 \in \{x2, x1, x3\}$
$x0 \in \{x3\} \Rightarrow x1 \in \{x3, x2, x4\}$
$x0 \in \{x4\} \Rightarrow x1 \in \{x4, x3, x5\}$
$x0 \in \{x5\} \Rightarrow x1 \in \{x5, x4, x6\}$
$x0 \in \{x6\} \Rightarrow x1 \in \{x6, x5, x7\}$
$x0 \in \{x7\} \Rightarrow x1 \in \{x7, x6, x8\}$
$x0 \in \{x8\} \Rightarrow x1 \in \{x8, x7\}$
$x1 \in \{x0\} \Rightarrow x2 \in \{x0, x1\}$
$x1 \in \{x1\} \Rightarrow x2 \in \{x1, x0, x2\}$
$x1 \in \{x2\} \Rightarrow x2 \in \{x2, x1, x3\}$
$x1 \in \{x3\} \Rightarrow x2 \in \{x3, x2, x4\}$
$x1 \in \{x4\} \Rightarrow x2 \in \{x4, x3, x5\}$
$x1 \in \{x5\} \Rightarrow x2 \in \{x5, x4, x6\}$
$x1 \in \{x6\} \Rightarrow x2 \in \{x6, x5, x7\}$
$x1 \in \{x7\} \Rightarrow x2 \in \{x7, x6, x8\}$
$x1 \in \{x8\} \Rightarrow x2 \in \{x8, x7\}$
$y0 \in \{y0\} \Rightarrow y1 \in \{y0, y1\}$
$y0 \in \{y1\} \Rightarrow y1 \in \{y1, y0, y2\}$
$y0 \in \{y2\} \Rightarrow y1 \in \{y2, y1, y3\}$
$y0 \in \{y3\} \Rightarrow y1 \in \{y3, y2, y4\}$
$y0 \in \{y4\} \Rightarrow y1 \in \{y4, y3, y5\}$
$y0 \in \{y5\} \Rightarrow y1 \in \{y5, y4, y6\}$
$y0 \in \{y6\} \Rightarrow y1 \in \{y6, y5\}$

$y1 \in \{y0\} \Rightarrow y2 \in \{y0, y1\}$
$y1 \in \{y1\} \Rightarrow y2 \in \{y1, y0, y2\}$
$y1 \in \{y2\} \Rightarrow y2 \in \{y2, y1, y3\}$
$y1 \in \{y3\} \Rightarrow y2 \in \{y3, y2, y4\}$
$y1 \in \{y4\} \Rightarrow y2 \in \{y4, y3, y5\}$
$y1 \in \{y5\} \Rightarrow y2 \in \{y5, y4, y6\}$
$y1 \in \{y6\} \Rightarrow y2 \in \{y6, y5\}$

NHL

$HelicoTreat \in \{NO\} \Rightarrow Surgery \in \{NONE\}$
$HelicoTreat \in \{NO\} \Rightarrow CtRtSchedule \in \{NONE\}$

Reactor [4]

$D1 \in \{test\} \Rightarrow Test \in \{excelent, good, bad\}$
$D1 \in \{notest\} \Rightarrow Test \in \{noresult\}$
$Test \in \{bad\} \Rightarrow D2 \in \{conventional, none\}$
$D2 \in \{conventional, none\} \vee Test \in \{b\} \Rightarrow Advanced \in \{\}$
$D2 \in \{advanced, none\} \Rightarrow Conventional \in \{\}$

[4] The constraint rules for the Reactor ID are depicted using the long names for variables and states given in page 312.

Index

Bibliography

[1] M. A. Artaso Landa. An empirical comparison of Influence Diagrams algorithms. Master's thesis, UNED, Escuela Técnica Superior de Ingeniería Informática, 2014.

[2] S. Benferhat and S. Smaoui. Hybrid possibilistic networks. *International Journal of Approximate Reasoning*, 44(3):224 – 243, 2007.

[3] C. Bielza and P. P. Shenoy. A comparison of graphical techniques for asymmetric decision problems: Supplement to management science paper. Technical report, Working Paper, 1998.

[4] C. Bielza and P. P. Shenoy. A comparison of graphical techniques for asymmetric decision problems. *Management Science*, 45(11):1552–1569, 1999.

[5] Mark Bloemeke and Marco Valtorta. A hybrid algorithm to compute marginal and joint beliefs in Bayesian networks and its complexity. In *Proceedings of the 14th conference on Uncertainty in AI*, pages 16–23. Morgan Kaufmann Publishers Inc., 1998.

[6] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 226–234. ACM, 1993.

[7] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proceedings of the 12th International Conference on Uncertainty in Artificial Intelligence*, pages 115–123. Morgan Kaufmann Publishers Inc., 1996.

[8] J. S. Breese and K. W. Fertig. Decision making with interval influence diagrams. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, pages 467–480. Elsevier Science Inc., 1990.

[9] C. J. Butz, J. Chen, K. Konkel, and P. Lingras. A comparative study of variable elimination and arc reversal in Bayesian network inference. In *FLAIRS Conference*, 2009.

[10] C. J. Butz and M. J. Sanscartier. On the role of contextual weak independence in probabilistic inference. In *Advances in Artificial Intelligence*, pages 185–194. Springer, 2002.

[11] R. Cabañas, A. Antonucci, A. Cano, and M. Gómez-Olmedo. Variable elimination for interval-valued influence diagrams. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty: 13th European Conference, ECSQARU 2015, Compiègne, France, July 15-17, 2015. Proceedings*, volume 9161 LNAI, pages 541–551. Springer, 2015.

[12] R. Cabañas, A. Antonucci, A. Cano, and M Gómez-Olmedo. Evaluating interval-valued influence diagrams. *International Journal of Approximate Reasoning*, 80:393–411, 2017.

[13] R. Cabañas, A Cano, M Gómez-Olmedo, and A. L. Madsen. Approximate lazy evaluation of influence diagrams. In *Advances in Artificial Intelligence: 15th Conference of the Spanish Association for Artificial Intelligence, CAEPIA 2013, Madrid, September 17-20, 2013, Proceedings*, volume 8109, page 321. Springer, 2013.

[14] R. Cabañas, A. Cano, M. Gómez-Olmedo, and A. L. Madsen. On SPI-lazy evaluation of influence diagrams. In *Probabilistic Graphical Models: 7th European Workshop, PGM 2014, Utrecht, The Netherlands, September 17-19, 2014. Proceedings*, pages 97–112. Springer International Publishing, 2014.

[15] R. Cabañas, A. Cano, M. Gómez-Olmedo, and A. L. Madsen. Improvements to variable elimination and symbolic probabilistic inference for evaluating influence diagrams. *International Journal of Approximate Reasoning*, 70:13–35, 2016.

[16] R. Cabañas, A. Cano, M Gómez-Olmedo, and A.L. Madsen. Heuristics for determining the elimination ordering in the influence diagram evaluation with binary trees. In *Twelfth Scandinavian Conference on Artificial Intelligence: SCAI 2013*, volume 257 of *Frontiers in Artificial Intelligence and Applications*, pages 65–74. IOS Press, 2013.

[17] R. Cabañas, M. Gómez, and A. Cano. Approximate inference in influence diagrams using binary trees. In *Proceedings of the Sixth European Workshop on Probabilistic Graphical Models (PGM-12)*, 2012.

[18] R. Cabañas, M. Gómez-Olmedo, and A. Cano. Evaluating asymmetric decision problems with binary constraint trees. In *Symbolic and Quantiative Approaches to Resoning with Uncertainty: 12th European Conference, ECSQARU 2013, Utrecht, The Netherlands, July 8-10, 2013, Proceedings*, volume 7958 LNAI, pages 85–96. Springer, 2013.

[19] R. Cabañas, M. Gómez-Olmedo, and A. Cano. Using binary trees for the evaluation of influence diagrams. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 24(01):59–89, 2016.

[20] R. Cabañas, A. L. Madsen, M. Gómez-Olmedo, and A. Cano. *On SPI for evaluating Influence Diagrams*, pages 506–516. Springer International Publishing, Cham, 2014.

[21] H. J. Call and W. A. Miller. A comparison of approaches and implementations for automating decision analysis. *Reliability Engineering & System Safety*, 30(1):115–162, 1990.

[22] A. Cano, M. Gómez, and S. Moral. A forward–backward Monte Carlo method for solving influence diagrams. *International Journal of Approximate Reasoning*, 42(1):119–135, 2006.

[23] A. Cano, M. Gómez-Olmedo, S. Moral, and C. B. Pérez-Ariza. Recursive probability trees for bayesian networks. In *Current Topics in Artificial Intelligence*, pages 242–251. Springer, 2009.

[24] A. Cano, M. Gómez-Olmedo, S. Moral, C. B. Pérez-Ariza, and A. Salmerón. Learning recursive probability trees from probabilistic potentials. *International Journal of Approximate Reasoning*, 53(9):1367–1387, 2012.

[25] A. Cano, M. Gómez-Olmedo, S. Moral, C. B. Pérez-Ariza, and A. Salmerón. Inference in bayesian networks with recursive probability trees: data structure definition and operations. *International Journal of Intelligent Systems*, 28(7):623–647, 2013.

[26] A. Cano, M. Gómez-Olmedo, and S. Moral. Approximate inference in Bayesian networks using binary probability trees. *International Journal of Approximate Reasoning*, 52(1):49–62, 2011.

[27] A. Cano and S. Moral. Heuristic algorithms for the triangulation of graphs. *Advances in Intelligent Computing—IPMU'94*, pages 98–107, 1995.

[28] A. Cano, S. Moral, and A. Salmerón. Penniless propagation in join trees. *International Journal of Intelligent Systems*, 15(11):1027–1059, 2000.

[29] J. F. Carriger and M. G. Barron. Minimizing risks from spilled oil to ecosystem services using influence diagrams: The deepwater horizon spill response. *Environmental science & technology*, 45(18):7631–7639, 2011.

[30] A. Charnes and W. W. Cooper. Programming with linear fractional functionals. *Naval Research logistics quarterly*, 9(3-4):181–186, 1962.

[31] J. M. Charnes and P. P. Shenoy. Multistage Monte Carlo method for solving influence diagrams using local computation. *Management Science*, pages 405–418, 2004.

[32] E. Charniak. Bayesian networks without tears. *AI magazine*, 12(4):50, 1991.

[33] B. Cheng and D. M. Titterington. Neural networks: A review from a statistical perspective. *Statistical science*, pages 2–30, 1994.

[34] B. R. Cobb. Hybrid influence diagrams for threat identification. In *Proceedings of the 14th Army Conference on Applied*

Statistics, Lexington, VA. Available online at: http://www. vmi. edu/uploadedfiles/faculty_webs/ecbu/cobbbr/edited_ books/acas2009. pdf. Citeseer, 2008.

[35] G. Corani, A. Antonucci, and M. Zaffalon. Bayesian networks with imprecise probabilities: Theory and application to classification. In *Data Mining: Foundations and Intelligent Paradigms*, pages 49–93. Springer, 2012.

[36] Z. Covaliu and R.M. Oliver. Representation and solution of decision problems using sequential decision diagrams. *Management Science*, 41(12):1860–1881, 1995.

[37] F. G. Cozman. Credal networks. *Artificial Intelligence*, 120:199–233, 2000.

[38] P. Dagum and M. Luby. An optimal approximation algorithm for Bayesian inference. *Artificial Intelligence*, 93(1):1–27, 1997.

[39] B. D'Ambrosio and S. Burgess. Some experiments with real-time decision algorithms. In *Proceedings of the 12th international conference on Uncertainty in AI*, pages 194–202. Morgan Kaufmann Publishers Inc., 1996.

[40] P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic networks and expert systems: Exact computational methods for Bayesian networks*. Springer, 2007.

[41] L. M. de Campos, J. F. Huete, and S. Moral. Probability intervals: a tool for uncertain reasoning. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 2(02):167–196, 1994.

[42] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Learning in graphical models*, pages 75–104. Springer, 1998.

[43] R. Demirer and P.P. Shenoy. Sequential valuation networks for asymmetric decision problems. *European Journal of Operational Research*, 169(1):286–309, 2006.

[44] F. J. Díez and M. Luque. Representing decision problems with decision analysis networks. Technical report, UNED, Madrid, Spain, 2010.

[45] Hugin Expert. Hugin api–reference manual, version 7.0. *Hugin Expert A/S*, 2008.

[46] B.C. Ezell, S.P. Bennett, D. Von Winterfeldt, J. Sokolowski, and A.J. Collins. Probabilistic risk analysis and terrorism risk. *Risk Analysis*, 30(4):575–589, 2010.

[47] E. Fagiuoli and M. Zaffalon. Decisions under uncertainty with credal influence diagrams. Technical Report 51-98, IDSIA, 1998. (unpublished).

[48] E. Fagiuoli and M. Zaffalon. A note about redundancy in influence diagrams. *International Journal of Approximate Reasoning*, 19(3):351–365, 1998.

[49] E. A. Feinberg and A. Shwartz. *Handbook of Markov decision processes: methods and applications*, volume 40. Springer Science & Business Media, 2012.

[50] K. W. Fertig and J. S. Breese. Interval influence diagrams. In *Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence*, pages 149–162. North-Holland Publishing Co., 1990.

[51] K. W. Fertig and J. S. Breese. Probability intervals over influence diagrams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):280–286, 1993.

[52] D. Geiger and D. Heckerman. Advances in probabilistic reasoning. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 118–126. Morgan Kaufmann Publishers Inc., 1991.

[53] D. Geiger, T. Verma, and J. Pearl. Identifying independence in Bayesian networks. *Networks*, 20(5):507–534, 1990.

[54] M. Gómez-Olmedo and A. Cano. Applying numerical trees to evaluate asymmetric decision problems. In T.D. Nielsen and N. Zhang, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 2711 of *Lecture Notes in Computer Science*, pages 196–207. Springer Berlin Heidelberg, 2003.

[55] C. Goutis. A graphical method for solving a decision analysis problem. *Systems, Man and Cybernetics, IEEE Transactions on*, 25(8):1181–1193, 1995.

[56] R. A. Howard and J. E. Matheson. Influence diagram retrospective. *Decision Analysis*, 2(3):144–147, 2005.

[57] HUGIN A/S. Hugin GUI help. `http://download.hugin.com/webdocs/manuals/Htmlhelp/`. Accessed: 20th February 2017.

[58] N. Huntley and M. C. M. Troffaes. Normal form backward induction for decision trees with coherent lower previsions. *Annals of Operations Research*, 195(1):111–134, 2012.

[59] M. Jaeger. Probabilistic decision graphs—combining verification and AI techniques for probabilistic inference. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 12(supp01):19–42, 2004.

[60] G. Jeantet and O. Spanjaard. Computing rank dependent utility in graphical models for sequential decision problems. *Artificial Intelligence*, 175(7):1366–1389, 2011.

[61] C. S. Jensen, U. Kjærulff, and A. Kong. Blocking Gibbs sampling in very large probabilistic expert systems. *International Journal of Human-Computer Studies*, 42(6):647–666, 1995.

[62] F. Jensen, F. V. Jensen, and S. L. Dittmer. From Influence Diagrams to junction trees. In *Proceedings of the 10th international conference on Uncertainty in AI*, pages 367–373. Morgan Kaufmann Publishers Inc., 1994.

[63] F. V. Jensen. *Introduction to Bayesian Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1996.

[64] F. V. Jensen and T. D. Nielsen. *Bayesian networks and decision graphs*. Springer Verlag, 2007.

[65] F. V. Jensen, T. D. Nielsen, and P. P. Shenoy. Sequential influence diagrams: A unified asymmetry framework. *International Journal of Approximate Reasoning*, 42(1-2):101–118, 2006.

[66] Y. Jin and B. Sendhoff. Pareto-based multiobjective machine learning: An overview and case studies. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(3):397–415, 2008.

[67] D. Kikuti, F. G. Cozman, and C. P. de Campos. Partially ordered preferences in decision trees: computing strategies with imprecision in probabilities. In *IJCAI workshop on advances in preference handling*, pages 118–123, 2005.

[68] U. B. Kjærulff. Triangulation of graphs–algorithms giving small total state space. Technical report, Aalborg University, 1990.

[69] U. B. Kjaerulff and A. . Madsen. Bayesian networks and influence diagrams. *Springer Science+ Business Media*, 200:114, 2008.

[70] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[71] C. L. König. Representing asymmetric decision problems with decision analysis networks. *Master Thesis Advanced Artificial Intelligence*, 2012.

[72] S. L. Lauritzen and D. Nilsson. Representing and solving decision problems with limited information. *Management Science*, pages 1235–1251, 2001.

[73] S. L. Lauritzen and D. J Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 157–224, 1988.

[74] Z. Li and B. D'Ambrosio. Efficient inference in Bayes networks as a combinatorial optimization problem. *International Journal of Approximate Reasoning*, 11(1):55–81, 1994.

[75] J. Lintusaari. *PCSI-labeled Directed Acyclic Graphs*. PhD thesis, University of Helsinki, 2014.

[76] P. J. F. Lucas, H. Boot, and B. G. Taal. Computer-based decision support in the management of primary gastric non-hodgkin lymphoma. *Methods of Information in Medicine-Methodik der Information in der Medizin*, 37(3):206–219, 1998.

[77] M. Luque, F. J. Dıez, and C. Disdier. Influence diagrams for medical decision problems: Some limitations and proposed solutions. *Proceedings of the Intelligent Data Analysis in Medicine and Pharmacology*, pages 85–86, 2005.

[78] A. L. Madsen. Variations over the message computation algorithm of lazy propagation. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36(3):636–648, 2005.

[79] A. L. Madsen and F. V. Jensen. Lazy evaluation of symmetric Bayesian decision problems. In *Proceedings of the 15th Conference on Uncertainty in AI*, pages 382–390. Morgan Kaufmann Publishers Inc., 1999.

[80] A. L. Madsen and F. V. Jensen. Lazy propagation: a junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence*, 113(1-2):203–245, 1999.

[81] B.G. Marcot, R.S. Holthausen, M.G. Raphael, M.M. Rowland, and M.J. Wisdom. Using Bayesian belief networks to evaluate fish and wildlife population viability under land management alternatives from an environmental impact statement. *Forest ecology and management*, 153(1):29–42, 2001.

[82] J. Merrick and G. S. Parnell. A comparative analysis of pra and intelligent adversary methods for counterterrorism risk management. *Risk Analysis*, 31(9):1488–1510, 2011.

[83] L. J. Neumann and O. Morgenstern. *Theory of games and economic behavior*. Princeton University Press Princeton, NJ, 1947.

[84] T. D. Nielsen and F. V. Jensen. Representing and solving asymmetric Bayesian decision problems. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 416–425. Morgan Kaufmann Publishers Inc., 2000.

[85] T. D. Nielsen and F. V. Jensen. Sensitivity analysis in influence diagrams. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 33(2):223–234, 2003.

[86] D. Nilsson and S. L. Lauritzen. Evaluating influence diagrams using LIMIDs. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 436–445. Morgan Kaufmann Publishers Inc., 2000.

[87] S. M. Olmsted. Representing and solving decision problems. *Dissertation Abstracts International Part B: Science and Engineering*, 45(3), 1984.

[88] D. K. Owens, R. D. Shachter, and R. F. Nease. Representation and analysis of medical decision problems with influence diagrams. *Medical Decision Making*, 17(3):241–262, 1997.

[89] J. Pearl. *Bayesian networks: A model of self-activated memory for evidential reasoning*. University of California (Los Angeles). Computer Science Department, 1985.

[90] J. Pearl. Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence*, 32(2):245–257, 1987.

[91] J. Pearl and S. Russell. *Bayesian networks*. Computer Science Department, University of California, 1998.

[92] J. Pensar, H. Nyman, J. Lintusaari, and J. Corander. The role of local partial independence in learning of Bayesian networks. *Int. J. Approx. Reasoning*, 69(C):91–105, February 2016.

[93] A. Piatti, A. Antonucci, and M. Zaffalon. Building knowledge-based systems by credal networks: a tutorial. *Nova Science*, page 0, 2010.

[94] R. Qi, L. Zhang, and D. Poole. Solving asymmetric decision problems with influence diagrams. In *Proceedings of the Tenth international conference on Uncertainty in artificial intelligence*, pages 491–497. Morgan Kaufmann Publishers Inc., 1994.

[95] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[96] H. Raiffa. *Decision analysis: Introductory lectures on choices under uncertainty.* Addison-Wesley, 1968.

[97] S. Ríos-Insua, M. Gómez, C. Bielza, and J. A. Fernández del Pozo. Implementation of IctNeo: a decision support system for jaundice management. In *Operations Research Proceedings 1999*, pages 554–559. Springer, 2000.

[98] N. Robertson and P. D. Seymour. Graph minors. IV. tree-width and well-quasi-ordering. *Journal of Combinatorial Theory, Series B*, 48(2):227–254, 1990.

[99] D. J. Rose. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. *Graph theory and computing*, 183:217, 1972.

[100] N. Wermuth S. L. Lauritzen. Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, 17(1):31–57, 1989.

[101] A. Salmerón, A. Cano, and S. Moral. Importance sampling in Bayesian networks using probability trees. *Computational Statistics & Data Analysis*, 34(4):387–413, 2000.

[102] R. D. Shachter. Evaluating influence diagrams. *Operations research*, pages 871–882, 1986.

[103] R. D. Shachter. Bayes-ball: Rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 480–487. Morgan Kaufmann Publishers Inc., 1998.

[104] R. D. Shachter, B. D'Ambrosio, and B. Del Favero. Symbolic probabilistic inference in belief networks. In *AAAI*, volume 90, pages 126–131, 1990.

[105] P. P. Shenoy. Valuation-based systems for Bayesian decision analysis. *Operations research*, 40(3):463–484, 1992.

[106] P. P. Shenoy. Binary join trees for computing marginals in the shenoy-shafer architecture. *International Journal of Approximate Reasoning*, 17(2):239–263, 1997.

[107] P. P. Shenoy. Valuation network representation and solution of asymmetric decision problems. *European Journal of Operational Research*, 121(3):579–608, 2000.

[108] J. E. Smith, S. Holtzman, and J. E. Matheson. Structuring conditional relationships in influence diagrams. *Operations Research*, 41(2):280–297, 1993.

[109] M. C. M. Troffaes. Decision making under uncertainty using imprecise probabilities. *International Journal of Approximate Reasoning*, 45(1):17–29, 2007.

[110] S. K. M. Wong and C. J. Butz. Contextual weak independence in Bayesian networks. In *Proceedings of the 15th conference on Uncertainty in AI*, pages 670–679. Morgan Kaufmann Publishers Inc., 1999.

[111] H. Xu and P. Smets. Reasoning in evidential networks with conditional belief functions. *International Journal of Approximate Reasoning*, 14(2–3):155–185, 1996.

[112] R. R. Yager and L. A. Zadeh. *An introduction to fuzzy logic applications in intelligent systems*, volume 165. Springer Science & Business Media, 2012.

[113] C. Yuan, X. Wu, and E. A. Hansen. Solving multistage influence diagrams using branch-and-bound search. *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence, UAI 2010*, pages 691–700, 2010.

[114] Lihua Z., Weiyi L., and Lizhen W. Influence diagram model with interval-valued utilities. In *Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, pages 601–605. IEEE Computer Society, 2009.

[115] M. Zaffalon. The naive credal classifier. *Journal of Statistical Planning and Inference*, 105(1):5–21, 2002.

[116] N. L. Zhang and D. Poole. Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328, 1996.

[117] E. Zitzler, D. Brockhoff, and L. Thiele. The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In *Evolutionary Multi-Criterion Optimization*, pages 862–876. Springer, 2007.