UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

MÁSTER EN INVESTIGACIÓN EN INTELIGENCIA ARTIFICIAL

Trabajo de fin de máster

OPTIMIZACIÓN DE ÁRBOLES DE ESTRATEGIA UNICRITERIO Y DE COSTE-EFECTIVIDAD

Ángel Miguel Gil González

Dirigido por:

Prof. Dr. / Manuel Arias Calleja.

Prof. Dr. / Manuel Luque Gallego.

Prof. Dr. / Francisco Javier Díez Vegas

Curso: 2019-2020: 2ª Convocatoria

Agradecimientos

Quisiera agradecer a Manuel Arias y Manuel Luque todas las reuniones, sugerencias, explicaciones, bibliografías recomendadas,.. En definitiva por toda su paciencia y todos los conocimientos transmitidos.

Si algo valoro de este trabajo fin máster es lo productivo que ha sido y la cantidad de conocimiento aprendido que sin duda van a ser muy útiles para mi vida profesional.

También agradecer a Francisco Javier Díez el haberme dado la oportunidad de trabajar en un trabajo tan interesante y de haber formado por un periodo de tiempo de su equipo de investigadores.

Aunque hubo momentos duros, que pensaba que no podría conseguirlo, ahora solo me quedo con todo lo bueno de este trabajo y del máster en general, donde la experiencia ha sido muy gratificante.

Por último agradecer a mi familia la paciencia por todo el tiempo que no les he dedicado.

Gracias a todos

Abstract

Introduction: A cost-effectiveness analysis (CEA) helps us select the most effective intervention for the financial budget we have. In these analyzes through CEP, we generated several lambda intervals, each with the optimal intervention, its cost and expected effectiveness. Being able to represent these lambda intervals, through a graphic model, will help us see all the available information and will speed us up to make a decision. If we can optimize this tree, it will be easier to study and understand it.

Objective: To find the algorithm that returns the best optimization of the generated tree after applying the deterministic CEA analysis.

Methods: Applying pruning techniques on the tree nodes (variable exchange, elimination of redundancies, lambda displacement) in a search algorithm, we will be able to get partial results of the search optimization, until we reach the most optimal tree optimization result created with CEP results.

Results: Once optimized with the most appropriate pruning techniques and algorithms, we will have a graphic model where the different options can be observed in a more effective and clear way than when we use a textual description.

Conclusion: Through the open source software tool "OpenMarkov" the possibility of graphically displaying the result of a deterministic CEA analysis has been implemented. With this new tool we can graphically evaluate the results that were previously shown in a table, and that did not allow us to appreciate in the same detail, each of the concepts that are observed in each branch of the tree.

Contents

List of Figures

Índice de tablas

Chapter 1

Presentación

1.1 Motivación

Las relaciones y patrones ocultos dentro de una base de datos representan el conocimiento.

Este conocimiento puede ser extraído a través de técnicas como redes neuronales, cálculos evolutivos, redes bayesianas, arboles de decisión,.. y de esta forma podremos disponer de un modelo gráfico que nos ayudará a tomar la decisión más óptima ante un problema planteado.

Los modelos generados a través de técnicas de la inteligencia artificial (IA) son representaciones abstractas de la realidad.

Para llegar a este modelado de la información es necesario un planteamiento del problema y una selección de datos acorde con el problema planteado.

También es necesario el disponer de agentes inteligentes que puedan analizar la información para detectar los patrones y las relaciones ocultos en la base de datos.

Los árboles de decisión se propusieron en la década de 1960 [Raiffa and Schlaifer, 1961, Raiffa, 1968] como una herramienta para resolver problemas de decisión. Cada árbol de decisión tiene tres tipos de nodos: azar, decisión y valor.

Los modelos gráficos probabilísticos (PGM) [Koller and Friedman, 2009] son una de las principales técnicas utilizadas en inteligencia artificial. Cada modelo establece una relación entre un gráfico G y una distribución de probabilidad P, de manera que algunas propiedades de G implican independencia condicional en P, dependiendo del tipo de PGM. Los primeros tipos de PGM surgieron en la década de 1980 en California, EE.UU .: los diagramas de influencia (ID) fueron propuestos en el Stanford Research Institute por [Howard and Matheson, 1984] y las redes bayesianas en la Universidad de California en Los Ángeles (UCLA) por Judea [Pearl, 1988]. Ambos tipos de modelos se basan en gráficos dirigidos acíclicos. La principal diferencia entre ellos es que los ID contienen tres tipos de nodos, al igual que los árboles de decisión, mientras que las redes bayesianas solo tienen nodos de azar. Por esta razón, el primero puede usarse para modelar y resolver problemas de decisión, mientras que el segundo solo puede usarse para razonamiento probabilístico; por ejemplo, diagnóstico.

OpenMarkov, dispone de diferentes técnicas ya integradas para proporcionar diferentes tipos de modelos gráficos probabilistas (PGM).

Con OpenMarkov podemos diseñar árboles de decisión o generarlos a partir de un ID o DAN, son redes $.pgmx$, resultado de estudios o análisis de bases de datos.

Cada ID se puede expandir en un árbol de decisiones, cuya complejidad generalmente aumenta con el número de nodos en el ID. Por lo tanto, la principal ventaja de los ID es la compacidad. La evaluación de un árbol de decisión devuelve una tabla de políticas para cada nodo de decisión, que crece exponencialmente con el número de variables conocidas al tomar esa decisión. Por ejemplo, la tabla para la última decisión en el ID Mediastinet, construida en la UNED [Luque et al., 2016], contiene más de 15.000 columnas, la mayoría de las cuales son irrelevantes, porque corresponden a escenarios imposibles o subóptimos. Este problema se resolvió mediante un algoritmo para ID que, en lugar de devolver una tabla, devuelve un árbol de estrategia. (Un árbol de estrategia es similar a un árbol de decisión, pero solo tiene dos tipos de nodos, azar y decisión, y no hay parámetros numéricos.) En el caso de Mediastinet, este algoritmo devuelve un árbol de estrategia que contiene solo 5 nodos, lo cual es claramente mucho más compacto que las tablas de decisión.

Otra limitación de la identificación es su incapacidad para modelar asimetrías. Existe una asimetría, por ejemplo, cuando existe una restricción en una decisión o cuando la información disponible para una decisión depende de elecciones previas. El tercer tipo de asimetría, más relevante, se debe a decisiones no ordenadas. Por ejemplo, en medicina es típico tener varias pruebas disponibles y no siempre está claro cuál debe hacerse primero y cuál (s) después, posiblemente dependiendo de los resultados de pruebas anteriores. Para superar esta limitación, varios investigadores del Centro de Investigación en Sistemas Inteligentes de Soporte a la Decisión (CISIAD), de la UNED, han propuesto un nuevo tipo de PGM, denominado redes de análisis de decisiones (DANs) [Díez et al., 2018], especialmente diseñado para modelar y resolución de problemas de decisión asimétrica. Los algoritmos propuestos para evaluar las DAN devuelven directamente árboles de estrategias, pero estos árboles suelen tener muchas redundancias. Por este motivo, sería deseable compactarlos, con el fin de reducir el número de ramas y hojas.

Por otro lado, el análisis de costo-efectividad es una técnica de decisión, utilizada principalmente en la economía de la salud [Drummond et al., 2015, Hunink et al., 2004], para determinar si el costo de una intervención supera el beneficio para la salud que aporta. En el caso de resultados inciertos, el problema se puede modelar con árboles de decisión, pero cuando se usa el algoritmo estándar de costo-efectividad, el tamaño del árbol crece de manera súper exponencial con el número de decisiones [Arias and Díez, 2014]. Un nuevo algoritmo propuesto en la UNED [Arias and Díez, 2011] permite construir árboles de decisión mucho más compactos, y su extensión a IDs [Arias and Díez, 2015] y DANs conduce a representaciones mucho más compactas y evaluaciones eficiente [Arias et al., 2017b]. Desafortunadamente, las estrategias correspondientes contienen incluso más redundancias que en el caso de los modelos

de decisión de unicriterio.

Pudiendo trabajar con el coste y efectividad asociados a un problema, tratamiento, intervención,.. Tendremos la posibilidad, a través de un análisis coste-efectividad (CEA), de estar aplicando la decisión más correcta en cada momento.

El resultado del análisis CEA suele ser no determinista, ya que partiendo de los mismos valores de estudio los resultados son muy variados, no es posible llegar a una única solución.

Por este motivo se aplicará un análisis CEA determinista [Arias and Díez, 2011] y a través de las particiones de coste-efectividad (CEP),veremos para cada intervalo de λ la intervención óptima, su coste y efectividad.

En el TFM aquí propuesto se plantea generar un árbol de decisión con nodo raíz λ, donde cada una de sus ramas es la intervención correspondiente al intervalo de λ generado en CEA determinista a través de CEP.

Con los árboles de decisión podemos ver el camino a seguir para cada resultado que se genera, por este motivo, tendremos un modelo gráfico que nos mostrara todas las opciones disponibles en el estudio realizado.

El principal problema de los árboles de decisión puede ser su tamaño, ya que va creciendo de forma exponencial según las variables que tengamos.

Un diagrama de influencia (ID) [Pauker and Wong, 2005] , es más directo al mostrar de forma global todas las probabilidades y dependencias entre las variables, pero no muestran al detalle toda la información que contiene cada intervención, para llegar a la solución, como lo haría un árbol de decisión.

Una vez que tenemos el árbol constituido por un nodo raíz λ y con cada una de sus ramas representando la intervención más óptima para cada intervalo de λ, el procedimiento será:

- Realizar un estudio con varias estrategias de búsqueda no informada e informada para ver cual de ellos se adapta mejor al proceso de optimización que necesitamos.
- Ver que técnicas podemos aplicar. Cada una de estas técnicas será una acción que podremos aplicar al árbol, por lo que vamos a generar una lista de acciones que se irán aplicando bajo el control del algoritmo que diseñemos.
- Las acciones consiste en aplicar alguna de las siguientes técnicas de optimización:
  - Intercambio de λ con los otros nodos del árbol. Esto provoca un desplazamiento de λ a lo largo del árbol agrupando las ramas y nodos en aquellos casos que sea posible.
    - λ y los nodos hijos tipo DECISION, realizara el agrupamiento en una única rama con un único nodo DECISION, que pasara a ser padre de λ.
    - λ y tipo CHANCE los hijos. En este caso se analiza si los dos nodos de tipo CHANCE son iguales y tienen una rama en común. Si se cumplen estas características podemos desplazar λ y que se convierta en hijo del nodo CHANCE.

Este nodo estará constituido por dos ramas, una de ellas será la rama común a a los dos nodos CHANCE y la otra formada por una nueva rama cuyo nodo raíz será λy cada una de sus ramas, las ramas diferentes entre los dos nodos CHANCES iniciales.

- Intercambio de variables.
  - Podemos tener intercambio entre variables del mismo tipo DECISION o CHANCE.
  - Intercambio entre variable Tipo CHANCE y DECISION. Si tenemos un nodo CHANCE y sus hijos son dos nodos tipo DECISION iguales y con el mismo estado, podremos realizar un intercambio entre CHANCE y DECISION y de esta manera reducir un nodo del árbol.
- Eliminación de redundancias. Cuando tenemos un nodo CHANCE con dos ramas iguales, podemos eliminar el nodo CHANCE y una de las ramas.

- Para cada iteración del algoritmo, si es posible aplicar una acción de la lista de acciones generadas, se irán generando diferentes estados como consecuencia de aplicar la acción correspondiente.

- El árbol resultante tiene que ser equivalente al original, es decir las decisiones deben ser las mismas. Se pretende que el árbol sea más pequeño, pero cada cambio tiene que ser equivalente al anterior.

- El proceso de optimización terminara cuando alcancemos una profundidad indicada en el espacio de búsqueda o encontremos la solución.

Con el análisis CEA determinista obtenemos una tabla de resultados donde podemos observar para cada columna, la estrategia óptima, coste que supone aplicar dicha estrategia , efectividad y los intervalos de λ donde esta estrategia es óptima.

Esta información es difícil de analizar, por tanto se planteo en estudios previos, la posibilidad de convertir toda la información generada a través de CEP en un árbol de decisión cuyo nodo raíz fuese λ y cada una de sus ramas las estrategias óptimas para dicho intervalo.

Aunque se consigue representar toda la información de CEP en un árbol, en ocasiones continua siendo difícil de interpretar, por la cantidad de ramas y nodos que contienen.

Por este motivo se planteo generar para OpenMarkov una herramienta que pueda simplificar los árboles de decisión.

En este Trabajo Fin de Máster se propone un proceso de optimización para los árboles de decisión, que se pueda aplicar a los árboles generados con CEA determinista y de esta manera facilitar la comprensión de toda la información mostrada.

1.2 Objetivos

El objetivo es disponer de un modelo gráfico, lo más sencillo posible (es decir, compactar) las estrategias obtenidas de diferentes modelos de decisión.

- ID de unicriterio y DAN
- Coste-efectividad
- OpenMarkov

1.3 Organización del documento

Siguiendo las recomendaciones del profesorado de este máster en inteligencia artificial, hemos intentado redactar una tesis relativamente breve que luego pueda ser enviada como artículo a una revista científica, como la Inteligencia Artificial en Medicina.

Por ello, en este primer capítulo presentamos una introducción a la tesis, la cual es más extensa que la propia de un artículo de revista: ya hemos expuesto explícitamente la motivación y objetivos de nuestro trabajo, en la siguiente sección ofrecemos un trasfondo detallado sobre teoría de grafos, búsqueda uniformada y heurística y modelos de decisión, incluyendo los fundamentos del análisis de costo-efectividad, y finalmente incluimos en la Sección [sec: Aspectos éticos] algunas consideraciones éticas.

En el segundo capítulo nos centraremos más en la optimización de árboles de estrategias para problemas de decisión y expondremos los experimentos y resultados realizados.

1.4 Conceptos básicos de teoría de grafos

Un grafo es un conjunto de vértices (nodos) unidos a través de un conjunto de aristas (arcos).

$G = (V, E)$

V Conjunto no vació de vértices (nodos).

E Conjunto de aristas (arcos), que muestran las relaciones entre los vértices.

- Hijos de un nodo.Sea G = (V, E) un grafo dirigido y v_ {1} un nodo en V. Un nodo v_ {2} en V es un hijo de v_ {1} si existe un arco (v_ {1}, v_ {2}) en E. El conjunto de hijos de un nodo v se denota como hijos (v).

- Padres de un nodo. Sea G = (V, E) un gráfico dirigido y v_ {1} y v_ {2} dos nodos en V. Decimos que v_ {1} es padre de de v_ {2} si v_ {2} es hijo de v_ {1}. El conjunto de padres de un nodo v se denota como padre (v).

- Arco saliente. Sea G = (V, E) una gráfica dirigida y v_ {1} sea un nodo en V. Cada arco e = (v_ {1}, v_ {2}) en E , donde v_ {2} es un nodo en V, se denomina arco saliente de v_ {1}.

- Ruta dirigida. Nodos inicial y final. Sea G = (V, E) una gráfica dirigida. Una ruta dirigida en G es una secuencia de n arcos (e_ {1}, e_ {2}, ..., e_ {n}) donde:
  - n> 0, y para todo entero i tal que 0 <i <n, el segundo nodo en el arco e_ {i-1} es igual al primer nodo en el arco e_ {i}.
  - El primer nodo de e_ {1} se denomina nodo inicial de la ruta dirigida.
  - El segundo nodo de e_ {n} se denomina nodo final de la ruta dirigida.
- Ciclo en un gráfico dirigido. Un ciclo en un gráfico dirigido es una ruta en la que el nodo inicial y el nodo final son iguales.
- Grafo dirigido acíclico. Se dice que un grafo dirigido es acíclico si no tiene ciclos.
- Ruta dirigida. Sea G = (V, E) una gráfica dirigida. Sean n_ {1} y n_ {2} dos nodos en V. Una ruta dirigida desde n_ {1} a n_ {2} es una ruta dirigida cuyo nodo inicial es n_ {1} y cuyo nodo final es n_ {2} .ción}
- Nodo alcanzable. Sea G un grafo dirigido y n_ {1} y n_ {2} dos nodos en G. Decimos que n_ {2} es accesible desde n_ {1} si existe una ruta desde n_ {1} a n_ {2}.
- Root. Sea G = (V, E) una gráfica dirigida. Una raíz es un nodo n_ {1} tal que por cada nodo n_ {2} en V \ setminus {n} tenemos que n_ {2} es accesible desde n_ {1}.
- Grafo dirigido enraizado. Sea G = (V, E) un grafo dirigido. Decimos que G tiene raíz si tiene exactamente una raíz.
- Árbol. Un gráfico G es un árbol si es un gráfico dirigido con raíz y es acíclico.
- Descendientes de un nodo. Sea v un nodo en un árbol t. Definimos los descendientes de un nodo v, denotado como descendiente (v), como el conjunto de nodos que son accesibles desde v en t.

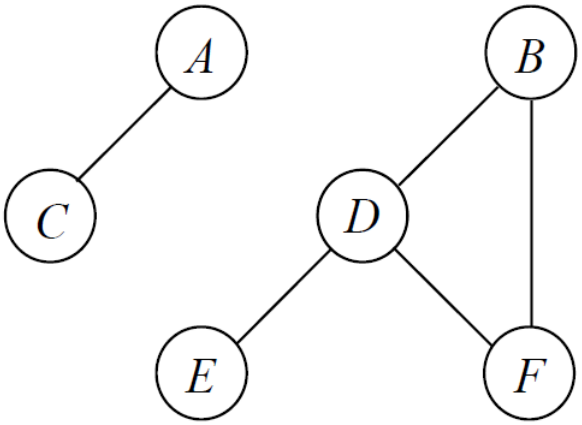

Figure 1.1: Ejemplo grafos

Cuando las aristas tienen un sentido definido estamos hablando de grafos dirigidos, en caso contrario tendremos grafos no dirigidos.

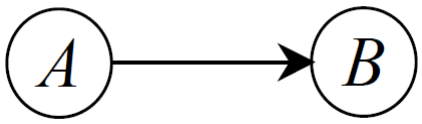

Figure 1.2: Grafo dirigido.

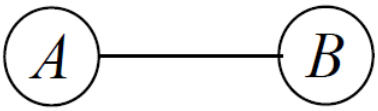

Figure 1.3: Grafo No dirigido.

Uno de los grafos más importantes son los árboles de decisión.

Un problema complejo lo podemos representar mediante una estructura de árbol. Esta estructura se base en la teoría de grafos, en la que cada una de sus ramas representa un grafo dirigido a una solución.

Las características principales de un árbol de decisión son:

- Los podemos considerar como un grafo dirigido acíclico, es decir, ningún arco empieza y termina en el mismo nodo.

- También son considerados como poliárbol, puesto que son grafos con todos sus arcos dirigidos.

- Su estructura permite analizar alternativas, eventos y los resultados.

- Están constituidos por nodos y ramas.

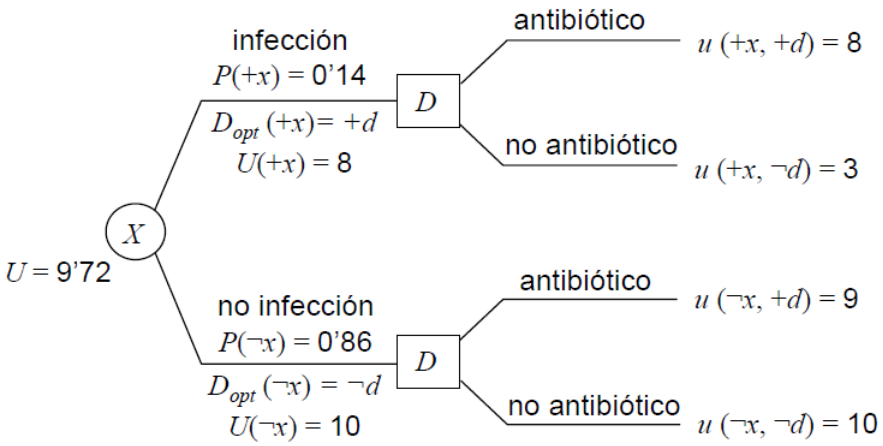

Figure 1.4: Ejemplo árbol decisión.

1.5 Conceptos básicos de política y estrategia en modelos gráficos probabilistas

1.5.1 Modelos gráficos probabilistas

Los diagramas de influencia (ID)

Son una parte importante en el análisis de decisiones, ya que muestran en un grafo las diferentes opciones a tomar ante un problema. Son grafos dirigidos y a cíclicos que representan la incertidumbre, las diferentes acciones a tomar y las preferencias. Los ID buscan encontrar la mejor solución para obtener la mayor ganancia.

Los ID pueden verse representados por un árbol de decisión.

Tienen tres tipos de nodos o variables:

- Azar. Variables aleatorias.
- Decisión. Posibles valores o alternativas que se pueden tomar.
- Utilidad. Objetivo a maximizar, es decir, la utilidad esperada

Redes de análisis de decisiones(DAN)

Es un nuevo tipo de modelo gráfico probabilista muy útil para resolver problemas de decisión. Un ID lo podemos convertir en un DAN, por lo que este modelo gráfico, igual que sucedía con los diagramas de influencia puede verse representado por un árbol de decisión.[Díez et al., 2018]

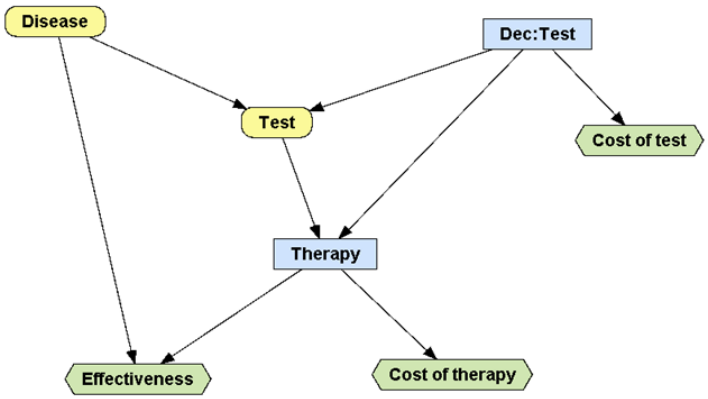

Figure 1.5: Ejemplo diagrama influencia (ID).

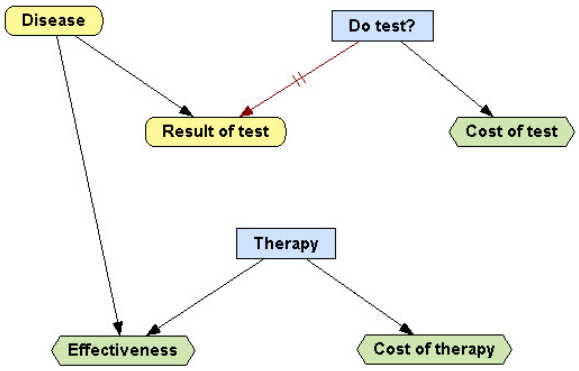

Figure 1.6: Ejemplo red análisis decisiones (DAN).

Los ID necesitan que todas las posibles decisiones que representan lleven un orden. En el caso de los DAN están decisiones pueden estar desordenadas o parcialmente ordenadas.

Bajo este criterio los DAN pueden ayudar, no solo a elegir que terapia aplicar ante cierta enfermedad, sino que si hay varias nos dirá cual de ellas es conveniente aplicar primero y cuales después en caso de ser necesario.

Con los ID solo podemos optar a saber que terapia aplicar en caso de tener varias opciones.

1.5.2 Política y estrategia en modelos gráficos probabilistas

La política consiste en una serie de criterios para poder alcanzar un objetivo.

La política en un ID consisten en una distribución de la probabilidad aplicada a una decisión teniendo en cuenta sus predecesores.[Luque and Díez, 2010, Luque, 2009]

Una estrategia para una ID es un conjunto de políticas, una para cada decisión.

En los árboles de decisión la política es la decisión alcanzada en cada rama, el nodo hoja.

La estrategia sería para cada rama los diferentes nodos no terminales que nos llevan a la decisión (hoja).

En los arboles de estrategia como veremos a continuación, la estrategia esta formada por

diferentes políticas (nodos de decisión y azar) que nos llevan a la decisión final, política en las hojas.

1.6 Árboles de decisión y de estrategia

Las políticas y estrategias se podrían representar en forma de tabla, pero la estructura en árbol puede ayudar a una presentación más visual. Vamos a repasar las ideas básicas existentes en la literatura para este tipo de representación.

1.6.1 Árboles de decisión

Machine learning genera modelos donde a través del uso de información aprende a identificar patrones e ir evolucionando para poder tomar decisiones y mostrar la solución más conveniente en cada caso.[Mitchell, 1997]

Los árboles de decisión se generan a partir de estas técnicas de machine learning, donde a través de un aprendizaje supervisado realiza entrenamientos que generan reglas tipo $if - else$ que resultan muy útiles pera poder interpretar la información.

Los sistemas de aprendizaje se pueden clasificar según:[Quinlan, 1986]

- La estrategia del aprendizaje.
- La representación del conocimiento (generación del modelo).

El aprendizaje va desarrollando un árbol de decisión de arriba (nodo raíz) hacia abajo (hojas del árbol), guiados por la clasificación realizada en forma de reglas.[Buchanan and Feigenbaum, 1978]

Las reglas $if - else$ que darán lugar al árbol de decisión se formaran a través de los atributos que proporciona el conocimiento adquirido, al analizar los datos con los que estamos construyendo el árbol y el proceso de aprendizaje.

Los datos que utilizamos para el diseño del árbol de decisión pueden provenir de bases de datos , historial de observaciones,..

El conocimiento generado a través de estos aprendizajes en forma de árboles de decisión nos proporcionara la capacidad de resolver problemas difíciles de una forma algo más sencilla.

Para tener un aprendizaje lo más optimo posible hay que minimizar el coste de clasificar un objeto.

Un árbol de decisión es una estructura jerárquica (una regla se implementa después de otra) que organiza la información a través de un nodo raíz, ramas formadas por nodos internos y nodos hoja.

Cada rama representa un camino a la solución y cada unos de los nodos internos las posibles acciones para llegar al nodo hoja, que es la solución correspondiente a la rama.

Los nodos no terminales (decisión y azar) son nodos observados, y los terminales (hojas) son decisiones.

Ventajas de los árboles de decisión:

- Simplifican la información de partida.
- Son muy fáciles de interpretar y ayudan a la toma de decisión.
- Reduce el número de variables independientes.
- Las diferentes acciones se encuentran en cada una de las ramas del árbol.
- Todas las opciones pueden ser analizadas.
- Podemos ver el coste y efectividad para cada una de las opciones que tengamos.
- Ayudan a visualizar los mejores resultados.

Entre las desventajas es que se necesita gran cantidad de información para obtener un árbol óptimo.

1.6.2 Árboles de estrategia

La característica principal de un árbol de estrategia es que esta formado por nodos internos de decisión que representan una solución intermedia antes de llegar a la solución definitiva mostrada en el nodo hoja.

Los nodos de un árbol de estrategia son:

- Nodos de decisión (forma cuadrada). Resultados tras algún tipo de evento.
- Nodos de azar (forma ovalada). Indican las alternativas que se pueden tomar.
- Nodos hoja. Resultado final de cada rama.

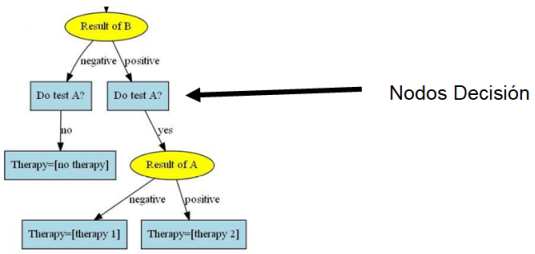

Figure 1.7: Nodos decisión en un árbol de estrategia.

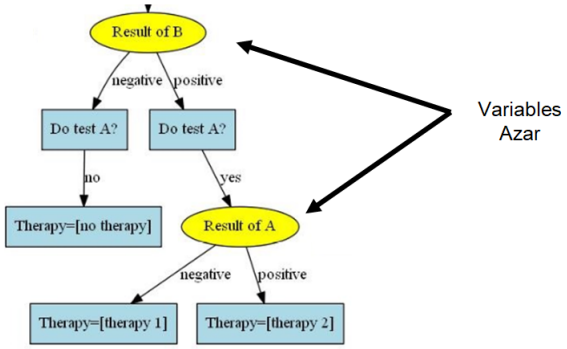

Figure 1.8: Nodos Azar en un árbol de estartegia.

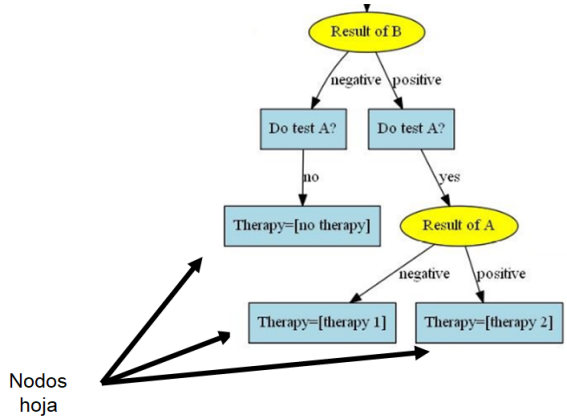

Figure 1.9: Nodos Hoja.

Las principales características de los arboles de estrategia son:[Luque et al., 2017]

- Tienen tres tipos de nodos (azar, decisión y hojas)
- Los nodos no terminales representan una variable de decisión o de azar correspondiente al ID.
- Varios nodos del árbol se pueden corresponder con la misma variable del ID.
- Cada Nodo de decisión tiene un solo hijo.
- Los nodos de azar son binarios, tienen solo dos posibles alternativas a seguir.
- Cada arco procedente de un nodo se etiqueta con el estado correspondiente.

1.6.2.1 Relación entre árboles de estrategia y árboles de decisión

Un árbol de estrategia es un árbol de decisión con nodos no terminales tomando decisiones.[Luque et al., 2017]

En el árbol de decisión solo los nodos terminales eran decisiones, en este tipo de arboles tenemos decisiones intermedias que nos llevan a la decisión final en cada rama.

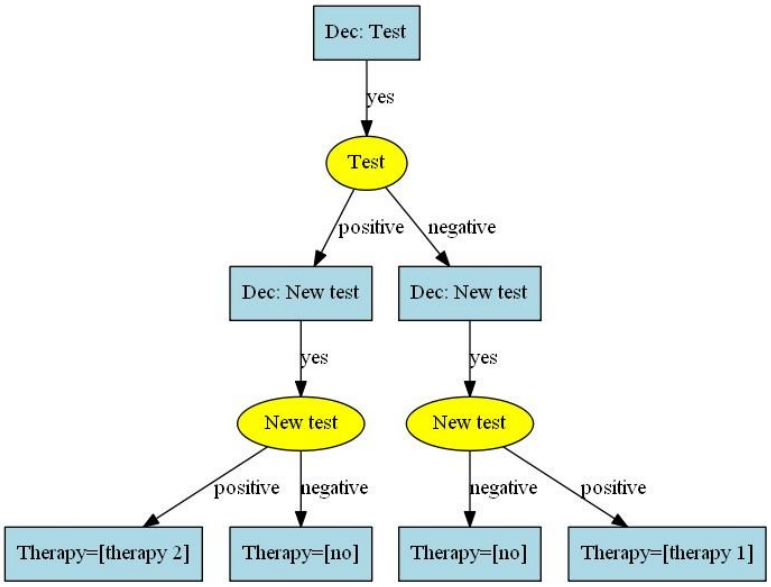

Figure 1.10: Ejemplo de árbol de estrategia para diagramas de influencia en OpenMarkov.

En redes de análisis de decisiones (DAN) el concepto de estrategia es más complejo ya que el orden de las decisiones puede ser diferente en distintas ramas. [Díez et al., 2018]

El árbol de estrategia correspondiente al análisis de coste-efectividad, tiene como nodo raíz lambda y como ramas la intervención óptima para cada intervalo.[Arias and Díez, 2014, 2011]

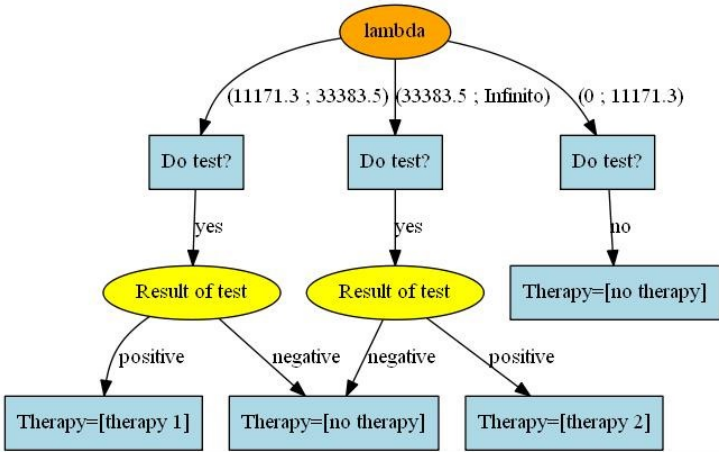

Figure 1.11: Ejemplo árbol de estrategia para análisis coste-efectividad

1.6.2.2 Árboles de estrategia en modelos gráficos probabilistas: OpenMarkov

OpenMarkov OpenMarkov es un programa libre para modelos gráficos probabilistas diseñado y supervisado por el Centro de Investigación sobre Sistemas Inteligentes de Ayuda a la Decisión de la Universidad Nacional de Educación a Distancia (UNED).

OpenMarkov trabaja con redes $.pgmx$, estas redes son el resultado del diseño de algún modelo gráfico probabilista (PGM).[Arias et al., 2019]

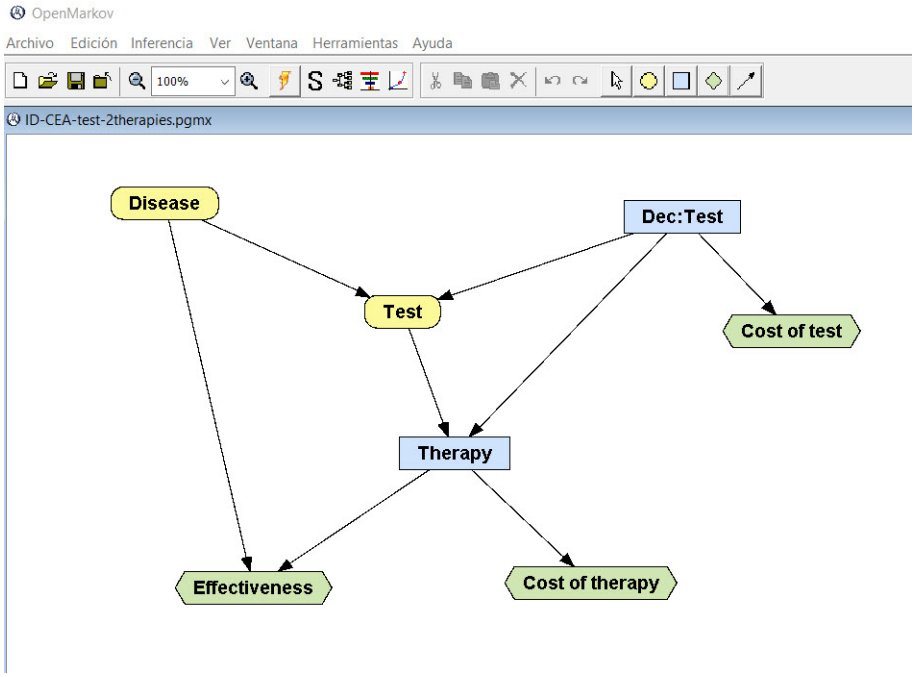

Figure 1.12: Interfaz OpenMarkov.

A través de su interfaz y con la implementación de una hoja de cálculo podremos obtener un modelo a analizar (ID, DAN, árboles decisión,..).

Entre las funciones que podemos encontrar dentro de OpenMarkov, tenemos:[Arias, 2009, Arias et al., 2017a, 2019]

- Diseño de redes bayesianas.
- Diagramas de influencia.
- Arboles de decisión.
- Análisis de coste-efectividad.

Árboles de estrategia en modelos gráficos probabilistas En OpenMarkov los árboles de estrategia son objetos de la clase StrategyTree.

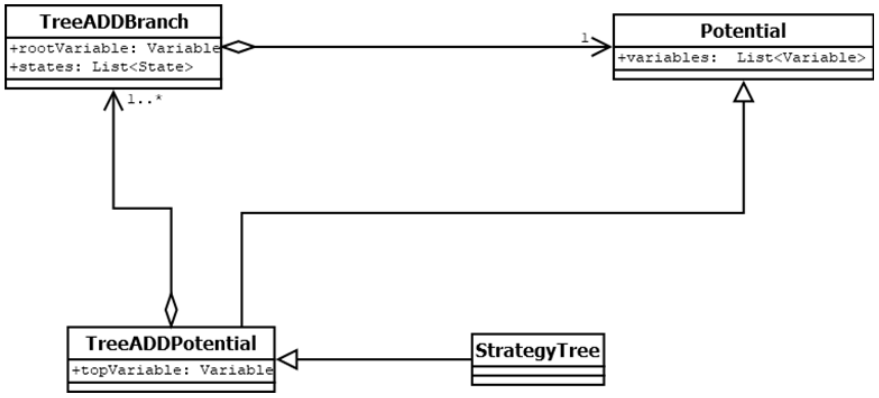

Figure 1.13: Diagrama de clases StrategyTree

Un StrategyTree representa reglas $if\ then$ con los diferentes caminos (decisiones) que el árbol puede tomar para llegar a las posibles soluciones.

Un ejemplo de StrategyTree en OpenMarkov, sería la red mediastinet:

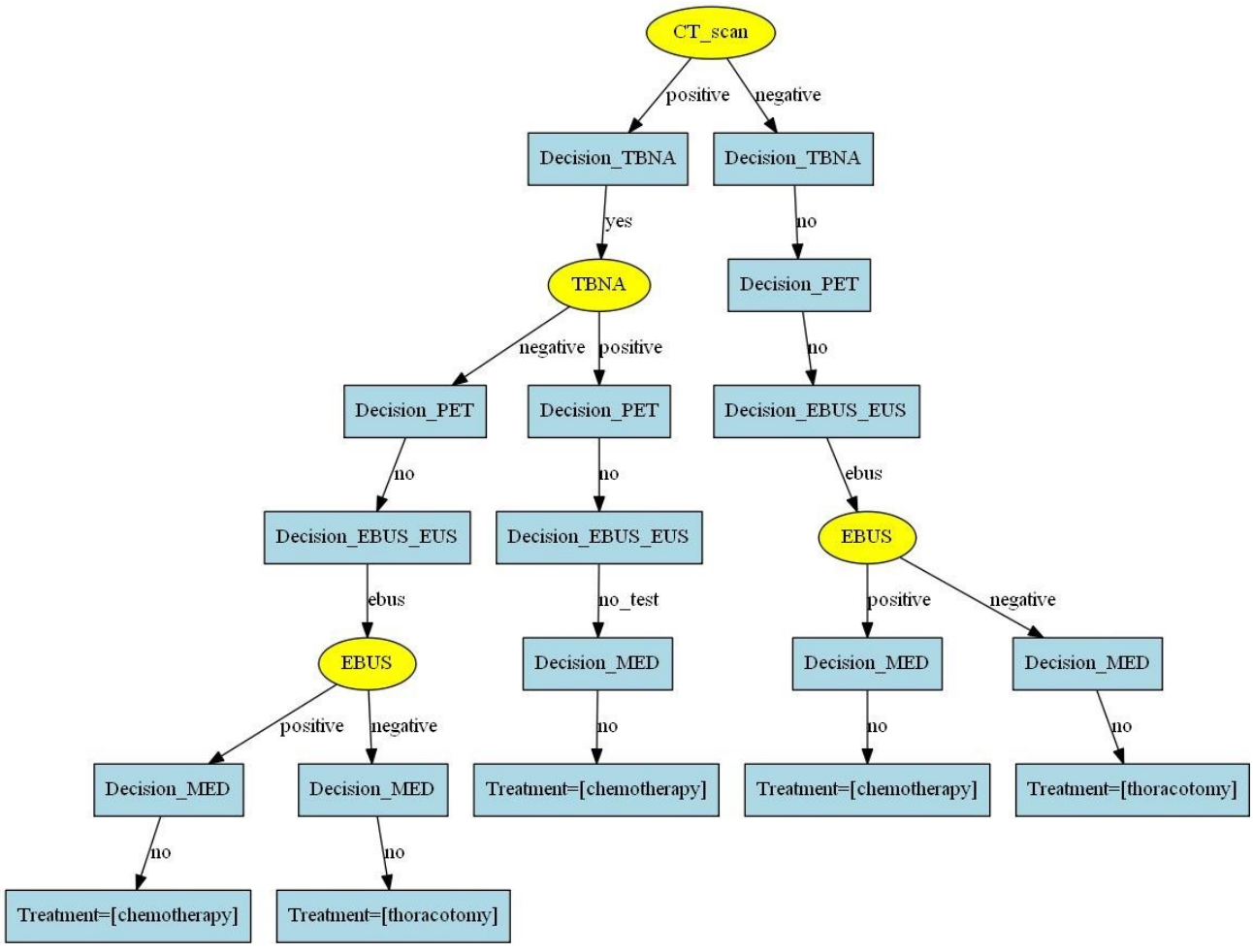

Figure 1.14: Ejemplo: StrategyTree - Mediastinet.

$$CT_scan = negative -> Decision_TBNA = no -> Decision_PET = no -> Decision_EBUS_EUS =$$

$ebus- > IFEBUS = negative- > Decision_M ED = no- > Treatment = thoracotomy IFEBUS = positive- > Decision_M ED = no- > Treatment = chemotherapy CT_s can = positive- > Decision_T BNA = yes- > IFTBNA = negative- > Decision_P ET = no- > Decision_E BUS_E US = ebus- > IFEBUS = negative- > Decision_M ED = no- > Treatment = thoracotomy IFEBUS = positive- > Decision_M ED = no- > Treatment = chemotherapy IFTBNA = positive- > Decision_P ET = no- > Decision_E BUS_E US = no_t est- > Decision_M ED = no- > Treatment = chemotherapy$

En el ejemplo correspondiente a una red unicriterio generada a partir de un ID observamos que las decisiones llevan el mismo orden en todas sus ramas.

Esto se complica cuando se utilizan redes de análisis de decisiones ya que el orden de decisiones puede ser distinto en distintas ramas. [Díez et al., 2018]

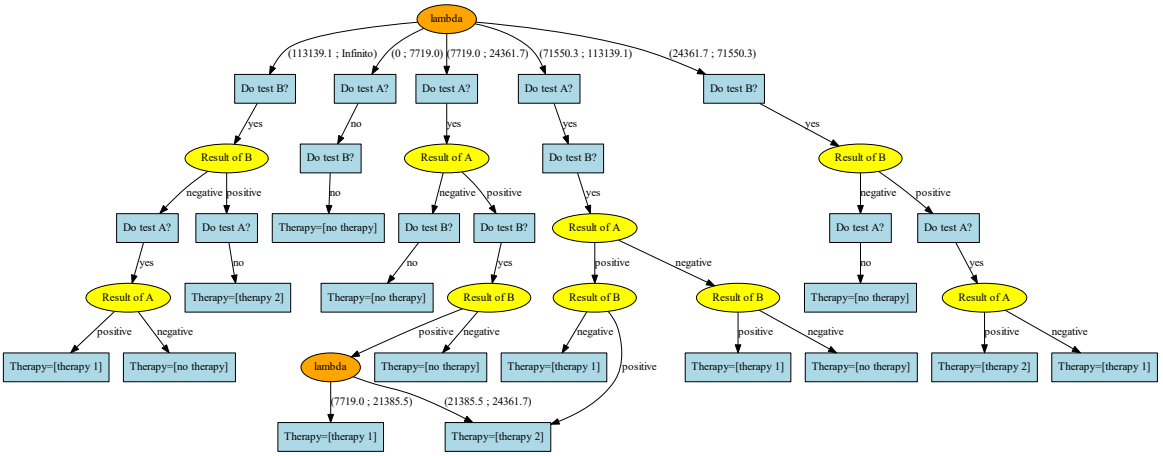

Figure 1.15: Estrategia en red análisis de decisiones.

Árboles de estrategia en análisis de coste-efectividad La gestión idónea de los recursos sanitarios pasa por un análisis coste-efectividad cuyo fin es aumentar la esperanza de vida de los pacientes, con la mejor calidad de vida y al menor coste posible.

Por tanto es necesario incluir una evaluación económica de los costes asociados a cada intervención o tratamiento, para disponer de un punto de vista económico que ayude a la toma de la decisión. [Arias and Díez, 2011]

Con un estudio de análisis coste - efectividad se busca poder afrontar diferentes enfermedades, intentando repartir los gastos de tal manera que todos los pacientes puedan acceder a los diferentes tratamientos disponibles. [Kuntz and Weinstein, 2001]

Supongamos que estamos analizando que tratamiento es el más conveniente aplicar a un paciente. Si disponemos de una enfermedad a la que se le pueden aplicar dos tipos de terapias, tendremos que ver cual es la que mejor se ajusta al paciente.

Cada terapia tiene asociada un coste y unos posibles efectos secundarios que pueden afectar a la calidad de vida del paciente.

Es importante tener claro que terapia es más apropiada para el paciente, y no solo por el coste que suponga, sino por los efectos secundarios que puedan generar.

Para poder aplicar cualquier terapia lo primero sería confirmar si el paciente tiene la enfermedad, pero en muchas ocasiones las pruebas que se realizan para verificar la enfermedad no dan valores muy fiables, por lo que es un riesgo aplicar cualquiera de ellas.

Por este motivo disponer de un estudio CEA para la enfermedad y los tratamientos a utilizar, sería lo más aconsejable.

Con este estudio obtenemos λ, un parámetro que indicará la disponibilidad a pagar en función de la efectividad.

CEA a través de λ relaciona que acción tomar teniendo en cuenta el coste y la efectividad.

En rangos elevados de λ esto se convierte en una tarea bastante complicada, por no decir imposible, por esta razón surge el concepto de particiones de coste-efectividad (CEP).[Arias and Díez, 2011]

CEP realiza un análisis por los intervalos de λ viendo que intervención es más óptima.

Cada intervención tiene un coste y una efectividad, por lo que tendremos en cuenta estos valores para poder obtener la mejor intervención para un intervalo de valores de λ.

Aplicando el análisis CEA determinista obtenemos un árbol de estrategia cuyo nodo raíz es λ y cada una de las ramas la mejor estrategia para el intervalo de λ en el que nos encontramos:[Arias and Díez, 2014, 2011]

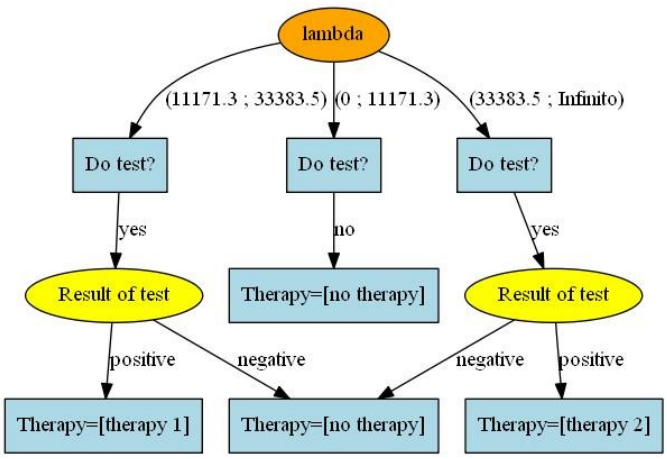

Figure 1.16: Árbol de estrategia en análisis coste-efectividad

1.7 Aspectos éticos

1.7.1 Adquisición y uso de datos

Todos los modelos de diagramas de influencia y redes de análisis de decisiones utilizados en esta tesis están abiertos y disponibles públicamente en Internet en el repositorio https://bitbucket.org/cisiad/org.p
No contienen información personal que pueda asociarse a ninguna persona en particular.

1.7.2 Reproducibilidad

Todo el software desarrollado en esta tesis es de código abierto, como el software OpenMarkov que se ha utilizado para la implementación de los algoritmos y los experimentos. El formato de los modelos en archivos * .pgmx se codifica siguiendo un formato xml de código abierto, cuya descripción se explica en detalle en www.promodelxml.org. Por lo tanto, la implementación y experimentación de esta tesis permite en gran medida la reproducibilidad de los resultados.

1.7.3 Implicaciones éticas

Esta tesis de maestría se ha guiado por la importancia de la explicabilidad a la hora de desarrollar modelos expertos en Inteligencia Artificial y al presentar los resultados a un ser humano. El objetivo principal de esta tesis fue encontrar representaciones en forma de árbol que puedan ayudar al usuario de una herramienta de software como OpenMarkov a comprender mejor las recomendaciones calculadas por algoritmos numéricos complejos al resolver un problema de decisión incierta. De esta forma, pensamos que este trabajo ha respetado profundamente el hecho de que la interpretabilidad de los resultados es cada vez más valorada por la sociedad en una aplicación informática.

Chapter 2

Optimización de árboles de estrategias para problemas de decisión

2.1 Acciones

En este capitulo detallaremos las técnicas utilizadas para poder podar los árboles de estrategia durante el proceso de optimización.

Las técnicas que se van a utilizar para la poda del árbol y conseguir un resultado óptimo son:

- Intercambio de variables.
- Eliminación de redundancias.
- Agrupamiento de ramas con lambda.

Para poder realizar el proceso de optimización vamos a estudiar diferentes algoritmos y analizar el espacio de búsqueda que generan y el tiempo de respuesta, es decir, el rendimiento de cada uno de ellos para quedarnos con el que nos proporcione una solución óptima en las mejores condiciones de uso.

Para optimizar un árbol hay que tener en cuenta que el árbol optimizado tiene que ser equivalente al árbol original, es decir las decisiones deben ser las mismas.

Se pretende que el árbol sea más pequeño, pero cada cambio tiene que ser equivalente al anterior.

2.1.1 Intercambio de variables

Una de las características que se va a aplicar al árbol para poder simplificarlo es el intercambio de variables.

Los tres tipos de intercambios que nos podemos encontrar son:

- Intercambio entre variables tipo chance.
- Intercambio entre variables tipo decisión.
- Intercambio entre variable chance y variables decisión.

2.1.1.1 Intercambio entre variables tipo CHANCE

Para poder realizar el intercambio entre dos variables tipo chance, se tiene que cumplir:

- Que los hijos sean iguales, es decir, mismo nodo y mismos estados.
- Tanto nodo padre, como nodos hijos deben tener los mismos estados

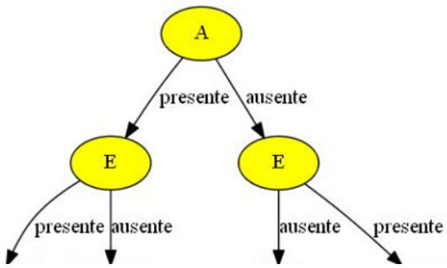

Figure 2.1: Ejemplo intercambio variables tipo chance.

El resultado que tenemos al aplicar intercambio entre las variables tipo chance es:

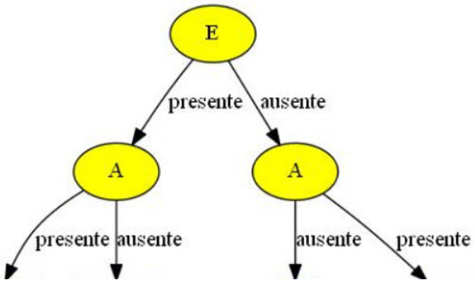

Figure 2.2: Resultado aplicar intercambio variables tipo chance.

2.1.1.2 Intercambio entre variables tipo DECISION

La única condición para intercambiar un nodo padre con el nodo hijo, es que los dos sean de tipo DECISION.

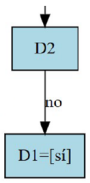

Figure 2.3: Ejemplo intercambio variables tipo decisión.

El intercambio de variables resultante es:

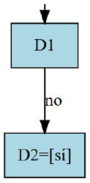

Figure 2.4: Resultado intercambio variables tipo decisión.

2.1.1.3 Intercambio entre variables de diferente tipo

Este caso lo encontramos cuando el nodo padre es de tipo chance y sus hijos de tipo decisión.

Para poder realizar el intercambio se tiene que cumplir:

- Los hijos sean iguales y con el mismo estado.
- Los nietos no influyen en el intercambio.

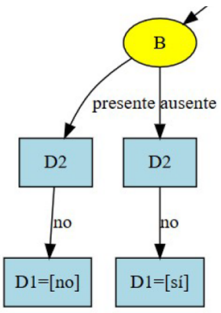

Figure 2.5: Ejemplo intercambio variables chance y decisión.

Como se cumplen las características descritas previamente, tendremos como resultado:

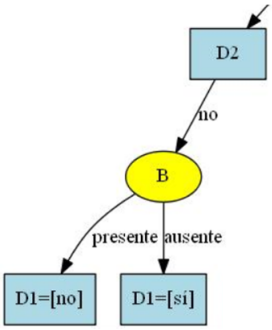

Figure 2.6: Resultado intercambio variables chance y decisión.

Para poder aplicar intercambio de variables dispondremos de la interface Actions y la clase ExchangeVariables.

ExchangeVariables hereda de la clase Action y dispone de los siguientes métodos:

- void doAction(). Aplica el intercambio de variables en el nodo correspondiente.
- boolean isApplicable(). Comprueba si la acción se puede aplicar al nodo.

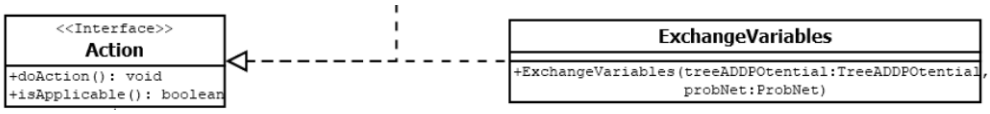

Figure 2.7: Diagrama clases ExchangeVariables.

2.1.2 Eliminar redundancias

En el árbol podemos encontrar nodos con dos ramas iguales. Estas redundancias provocan una dificultad a la hora de interpretar el árbol y por lo tanto pueden ser eliminadas para obtener un árbol más óptimo.

Tendremos redundancia cuando de una variable CHANCE nacen dos ramas iguales.

Ejemplo de redundancia:

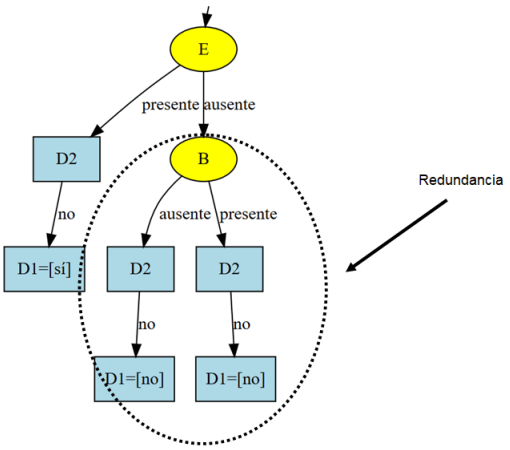

Figure 2.8: Ejemplo de redundancia.

Al aplicar la eliminación de redundancia en el nodo CHANCE B, tendremos:

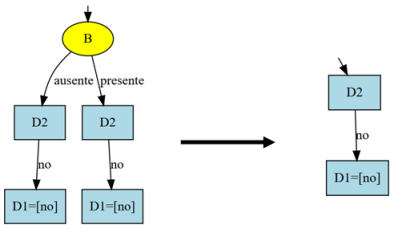

Figure 2.9: Resultado de aplicar redundancia en el nodo B.

El resultado final de aplicar redundancia al ejemplo mostrado, será:

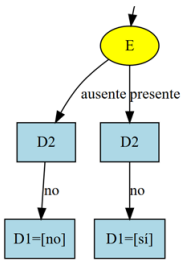

Figure 2.10: Resultado final tras aplicar eliminación de redundancia.

Para poder aplicar eliminación de redundancias dispondremos de la interface Actions y la clase RemoveRedundancy.

RemoveRedundancy hereda de la clase Action y dispone de los siguientes métodos:

- void doAction(). Aplica el intercambio de variables en el nodo correspondiente.
- boolean isApplicable(). Comprueba si la acción se puede aplicar al nodo.

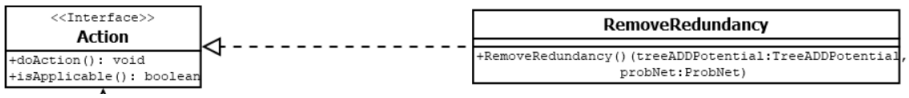

Figure 2.11: Diagrama clases RemoveRedundancy.

2.1.3 Agrupamiento de ramas con lambda

Con la información que nos proporciona CEA determinista a través de cada CEP generado para cada intervalo de lambda, procedemos a diseñar el árbol que tendrá como nodo raíz "lambda" y como ramas la intervención óptima para cada intervalo.

A través de las técnicas vistas anteriormente y la que vamos a describir en esta sección procederemos a optimizar el árbol.

Para poder aplicar un agrupamiento de los nodos y ramas del árbol, a partir de "lambda", tendremos en cuenta:

2.1.3.1 Intercambio "lambda" (Nodo raíz) con nodos DECISION

Lo primero de todo es que la variable raíz "lambda" no se mueve, permanece en su posición de nodo padre, y si tiene nodos de decisión iguales y con el mismo estado, lo que hace es agrupar las ramas en una única rama.

Si partimos del ejemplo de la siguiente figura:

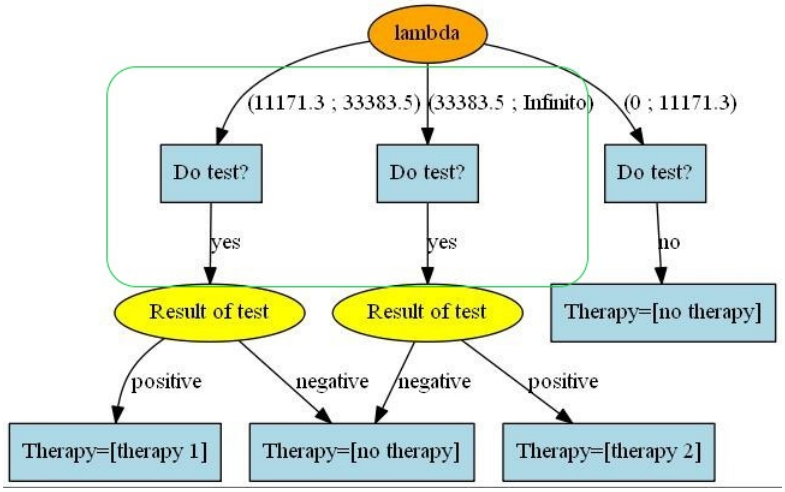

Figure 2.12: Ejemplo intercambio lambda con nodos decisión.

Tenemos dos ramas con dos nodos DECISION (Do test?) con el mismo estado.

Las dos ramas se podrían agrupar de tal forma que tendríamos un único nodo DECISION y seguidamente vendría la variable lambda con el resto de la información de cada rama, para cada uno de sus estados.

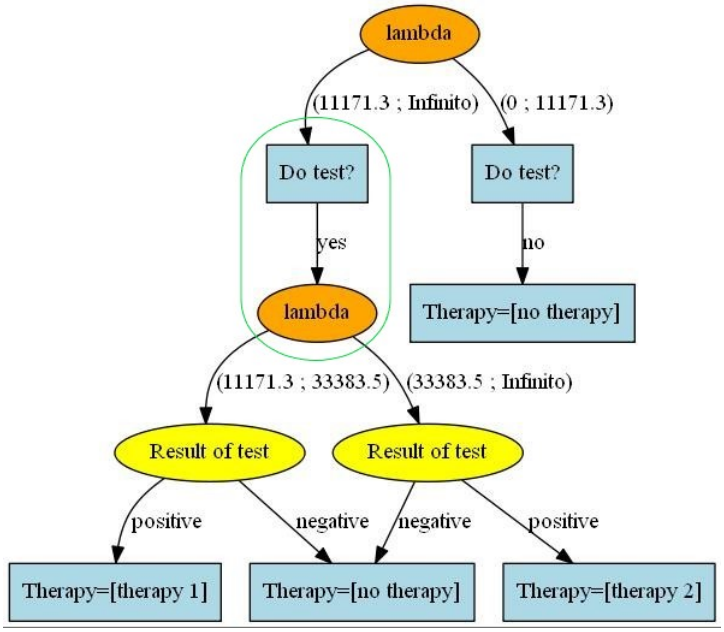

Figure 2.13: Intercambio lambda (Nodo raíz) con nodos decisión.

Otro intercambio que podemos hacer es cuando lambda no es el nodo raíz y sus hijos son dos nodos DECISION iguales y con el mismo estado.

2.1.3.2 Intercambio "lambda" con nodos DECISION

En este caso lambda es un nodo que va buscando desplazarse a lo largo del árbol para ir agrupando los nodos y ramas a través de lambda.

A diferencia con lo que sucede en el caso anterior, aquí lambda no se mantiene, lo desplazamos a un nuevo nivel del árbol.

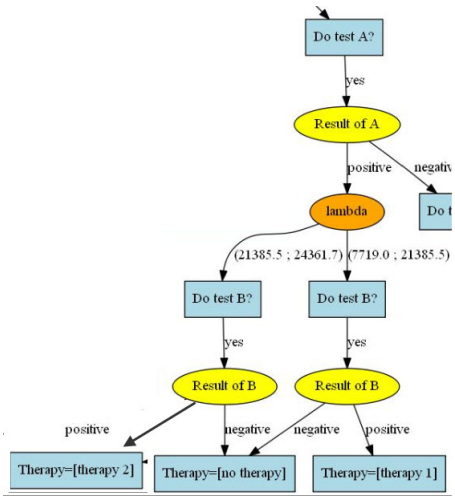

Figure 2.14: Intercambio lambda con nodos decisión.

Quitamos un nodo DECISION (DO Test B?) y lo subimos por encima de lambda, de esta manera desplazamos lambda y vamos agrupando los nodos comunes

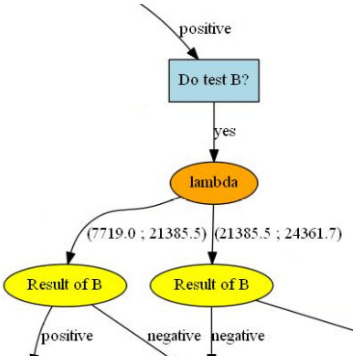

Figure 2.15: Resultado del intercambio lambda con nodos decisión.

2.1.3.3 Intercambio de lambda con variables CHANCE

Podemos aplicar el intercambio cuando tengamos lambda con dos variables CHANCE iguales, y al menos con una rama en común en cada uno de los nodos CHANCE, para poder agrupar en función de lambda.

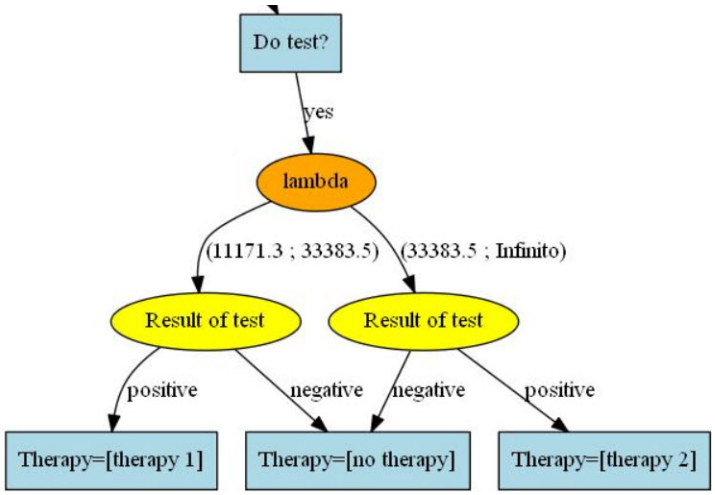

Figure 2.16: Intercambio lambda con variables CHANCE.

La rama en común se queda como rama del nodo CHANCE y las ramas diferentes pasan a ser las nuevas ramas del nodo lambda, cada una con su intervalo λ correspondiente.

El resultado que obtenemos es:

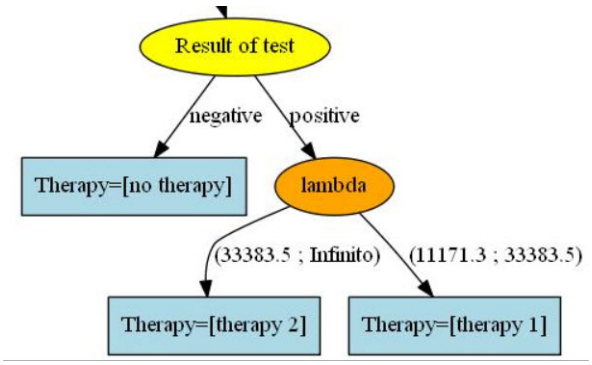

Figure 2.17: Resultado intercambio lambda con variables CHANCE.

2.2 Estrategias de búsqueda

El problema a resolver es, dado un árbol de estrategias t, encontrar un árbol de estrategias que sea compatible con t y que maximice una determinada función objetivo relacionada con lo bien que un usuario entendería las decisiones prescritas por t. Hemos considerado que cuantos menos nodos tenga un árbol, más fácil será de entender. Por lo tanto, la función objetivo f a maximizar por nuestro algoritmo de solución es f (t) igual menos el número de nodos en t.

El espacio de búsqueda es el conjunto de árboles de estrategias que son compatibles con el árbol de estrategias inicial t y son accesibles desde t aplicando un conjunto finito de acciones de las descritas en \ refsec {acciones}.

Para garantizar que la búsqueda finalice para cada árbol de estrategias en un tiempo razonable al ejecutar en una computadora, decidimos incorporar un límite de profundidad al

algoritmo.

Algorithm 1: Search on a state space.

Input: An initial state s_0

Result: $bestSolution$: The best state satisfying certain properties

```
1  open ← {s_0};
2  closed ← ∅;
3  bestSolution ← {s_0};
4  while open ≠ ∅ do
5      s ← select a state from open for expansion;
6      open ← open \ {s};
7      closed ← closed ∪ {s};
8      bestSolution ← updateBestSolution(bestSolution, s);
9      foreach s' ∈ children(s) do
10         if s' ∉ closed then
11             open ← open ∪ {s'};
12         end
13     end
14 end
```

Figure 2.18: Algoritmo de búsqueda.

Hemos decidido utilizar tres algoritmos:

- Búsqueda en amplitud, que se denominará BFS.
- Búsqueda en profundidad primero, que se denominará DFS.
- Búsqueda del mejor primero, utilizando como función heurística para guiar la búsqueda el número de nodos en el árbol de estrategia; este algoritmo se denominará Heur.

Considerar decir que en la acción de reemplazo de nodo no se ha aplicado el chequear árboles compatibles, sino solo árboles iguales, ya que en OpenMarkov se tenía esta función, y no la otra.

Para buscar la solución más óptima vamos a representar una búsqueda de espacio de estados.

El estado va a representar una solución parcial dentro del espacio de búsqueda y representa la acción que aplicamos sobre el estado padre. \

Este espacio de búsqueda que vamos generando lo vamos a representar cono un grafo dirigido, donde cada estado es un nodo del grafo y cada arco la acción que vamos a aplicar.

Vamos a utilizar el siguiente árbol como ejemplo para ver como desarrollamos el esapcio de búsqueda:

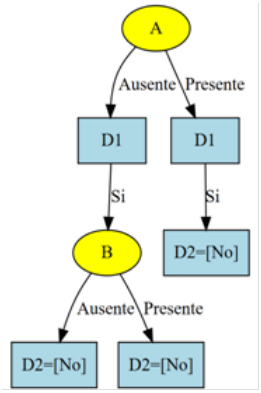

Figure 2.19: Ejemplo árbol decisión.

En este ejemplo se han incluido intercambio de variables y redundancia, para poder tener una visión más amplia de la optimización del árbol.

Aplicando las diferentes posibles acciones sobre este árbol se crea un espacio de búsqueda que se irá recorriendo de diferentes maneras según el algoritmo que estemos utilizando.

Partiendo del Estado 0 (el árbol en su estado inicial), tendremos las siguientes acciones:

- Intercambio de variables entre la variable CHANCE A y las de DECISION D1.
- Intercambio entre las variables de DECISION D1 y D2.
- Eliminar redundancia en la variable CHANCE B.

Los estados que surgen al aplicar las acciones a partir del ESTADO 0 son:

ESTADO 1:

Para generar el ESTADO 1 a partir del ESTADO 0 se ha aplicado una acción de intercambio de variables entre CHANCE A y sus hijos DECISION D1.

Los hijos de CHANCE A tienen el mismo valor y mismos estados, por lo que se puede aplicar este intercambio de variables.

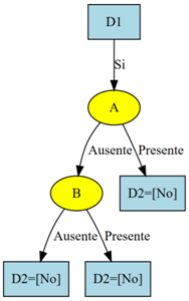

Figure 2.20: Estado 1 - Resultado de aplicar Intercambio de variables entre variable chance A y decisión D1.

ESTADO 2:

El ESTADO 2 es el resultado de aplicar intercambio de variables entre las variables de decisión D1 y D2.

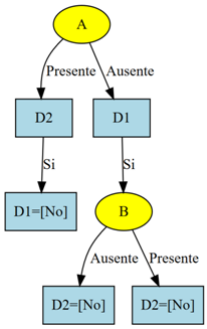

Figure 2.21: Estado 2 - Intercambio entre variables de decisión D1 y D2.

ESTADO 3

Aplicamos la eliminación de redundancia que tiene el nodo B, variable CHANCE con dos ramas iguales.

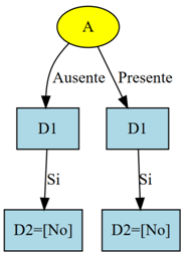

Figure 2.22: Estado 3 - Eliminación de redundancia en la variable CHANCE B.

Con estos tres estados tendremos el primer nivel del árbol de búsqueda generado tras aplicar las posibles acciones al ESTADO 0 (árbol que queremos optimizar).

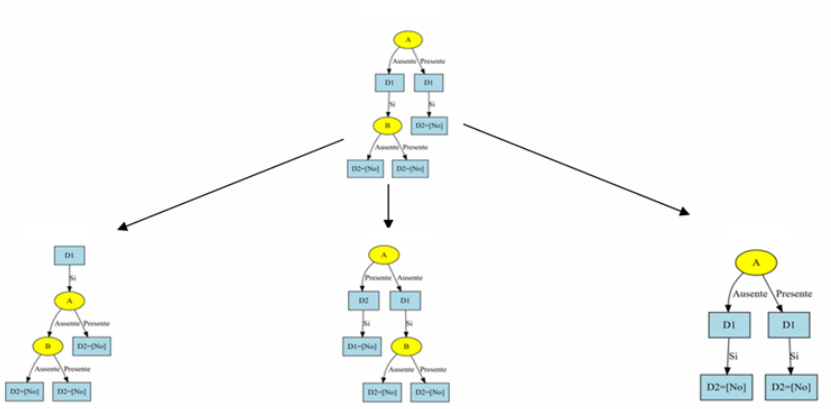

Figure 2.23: Árbol de búsqueda con el primer nivel del espacio de búsqueda.

El siguiente nivel del árbol se va a generar con las acciones que apliquemos a los estados 1, 2 y 3.

Sucesivamente y según vamos aplicando acciones vamos diseñando el espacio de búsqueda.

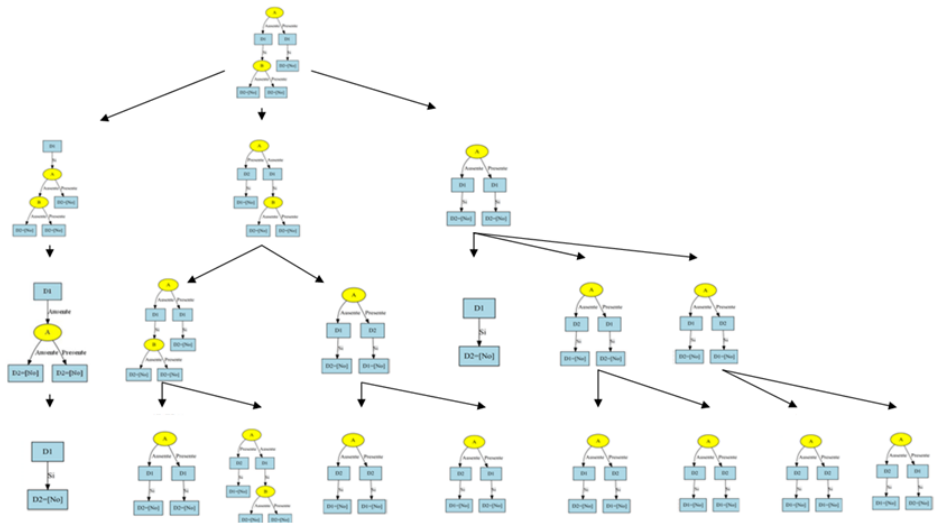

Figure 2.24: Ejemplo de espacio de búsqueda con cuatro niveles.

En estos cuatro niveles del árbol de búsqueda que hemos generado encontramos dos soluciones óptimas al árbol que estamos optimizando, las soluciones las tenemos:

- Nivel 2. Acción de eliminación de redundancia al ESTADO 3.
- Nivel 3. Aplicando una eliminación de redundancia al ESTADO 4.

Tenemos que ver como llegar de la forma más directa y usando el menor tiempo y recursos de memoria a una de las soluciones óptimas generadas en el espacio de búsqueda. De esto va a depender el tipo de algoritmo que apliquemos.

2.2.1 Búsqueda no informada

Sin ningún tipo de información buscamos la solución óptima.

Para poder utilizar los algoritmos de búsqueda tendremos las siguientes consideraciones:

- Generamos nuevos estados aplicando las posibles acciones al árbol.
- Necesitamos crear una copia en profundidad del estado inicial y a partir de esa copia realizar los cambios.
- Vamos generando un espacio de búsqueda con cada estado que vamos creando.
- Vamos a expandir hasta alcanzar una solución.

Vamos a analizar y realizar pruebas con los siguientes tipos de algoritmos:

- Búsqueda en anchura (BreadthFirst).
- Búsqueda en profundidad (DepthFirst).

2.2.1.1 Búsqueda en anchura BFS

Características:

- Comenzamos expandiendo el nodo raíz y luego cada uno de sus hijos.
- Vamos expandiendo por niveles.
- Se expanden los nodos de cada nivel antes de comenzar con el siguiente nivel.
- Utiliza una lista FIFO

Lista FIFO

Tomamos los estados que obtenemos al aplicar las acciones al ESTADO 0 y los introducimos en una pila FIFO.

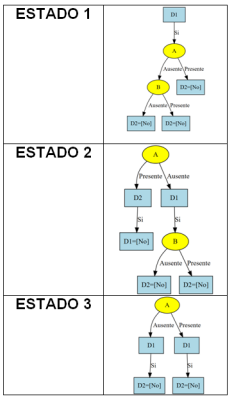

Figure 2.25: FIFO. Almacenados primeros estados.

Al no encontrar la solución y ser una pila FIFO el primero en entrar es el primero en salir, por lo que sacamos el ESTADO 1 e introducimos sus hijos, que en este caso es el ESTADO 4.

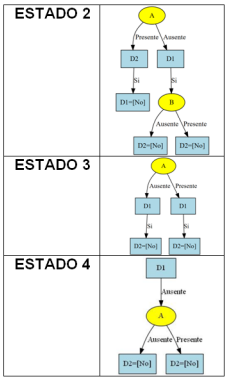

Figure 2.26: FIFO. Introducidos los hijos tras quitar el ESTADO 1.

Hacemos lo mismo con el ESTADO 2.

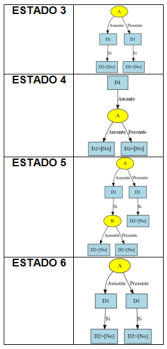

Figure 2.27: FIFO. Sacamos el ESTADO 2 e introducimos sus hijos (ESTADOS 5 y6).

Como seguimos sin encontrar una solución óptima continuamos con el proceso de búsqueda a través de la pila FIFO, en este caso eliminamos el ESTADO 3 de la pila e introducimos sus hijos.

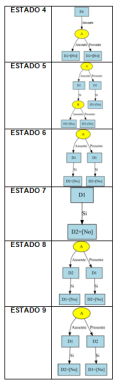

Figure 2.28: FIFO. Sacamos el ESTADO 3 e introducimos sus hijos (ESTADOS 7, 8 y 9).

El ESTADO 7 es la solución óptima que estamos buscando por lo que continuamos con el proceso hasta que a través de la cola FIFO lleguemos a este estado.

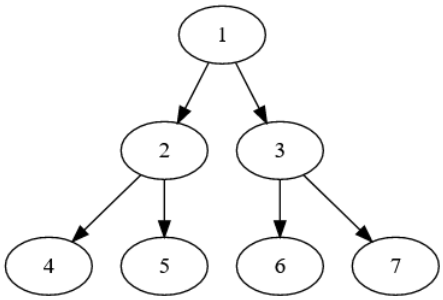

Figure 2.29: Secuencia de búsqueda utilizando BFS.

Proceso BFS:

Guardamos en la cola QUEQUE el ESTADO 0 correspondiente con la StrategyTree original.

A partir de este momento realizamos un bucle, donde realizamos las siguientes acciones: DO

- Comprobar si la cola tiene datos.
- Sacamos el primer estado introducido en la QUEQUE (FIFO).
- Analizamos el ESTADO N que hemos obtenido con queque.poll().
- Generamos los hijos correspondientes a este estado tras aplicar las acciones.
- IF NO genera hijos, hemos llegado a la solución.
- ELSE iremos almacenando en QUEQUE los hijos del STATE N.

WHILE (No tengamos la solución OR cola tenga datos)

2.2.1.2 Búsqueda en profundidad DFS

Características:

- Se expande el nodo raíz, seguidamente uno de sus hijos y el hijo de este y así sucesivamente hasta llegar a la hoja del árbol.
- Si al llegar a la hoja no hay solución volvemos hacia el nodo padre, para probar con el siguiente hijo.
- Utiliza lista LIFO (último en entrar, primero en salir).
- Puede no encontrar una solución óptima, puede probar muchos caminos sin solución.
- Necesita poco espacio de memoria, solo almacena el camino que está analizando.

Lista LIFO

Tomamos los estados 1, 2 y 3 que son los hijos del ESTADO 0 y los introducimos en una pila LIFO.

Al no encontrar la solución y ser una pila LIFO el último en entrar es el primero en salir, por lo que sacamos el ESTADO 3 e introducimos sus hijos, que en este caso son los estados 7, 8 y 9.

Al sacar cada estado de la pila, comprobamos si es la solución óptima y podemos parar el proceso.

En caso de no encontrar lo que buscamos, introducimos sus hijos en la pila.

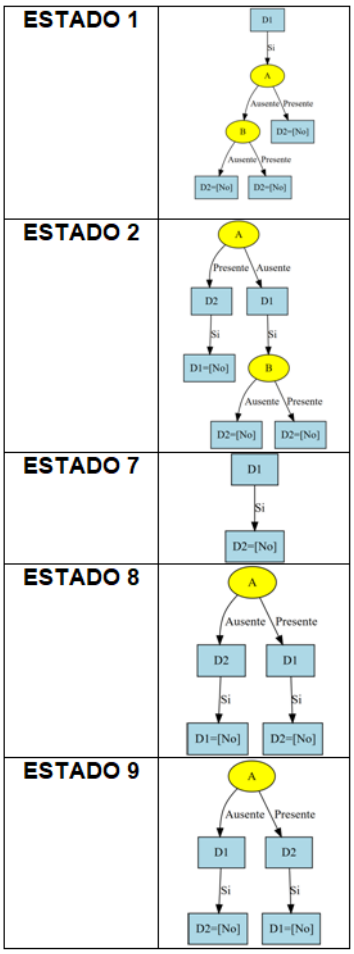

Figure 2.30: LIFO. Quitamos el ESTADO 3 e introducimos sus hijos (ESTADOS 7, 8 y 9).

El ESTADO 7 es la solución que estamos buscando pero el próximo estado a alizar es el ESTADO 9.

Continuamos con el proceso hasta encontrar la solución o llegar a una profundidad indicada.

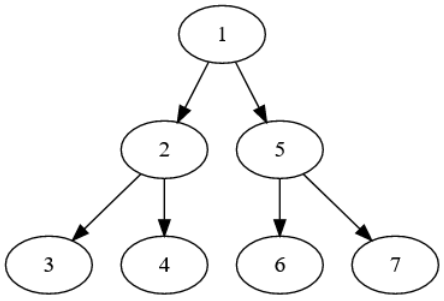

Figure 2.31: Estrategia de búsqueda con DFS.

Proceso DFS:

Guardamos en la pila STACK el STATE 0 correspondiente con la StrategyTree original.

A partir de este momento realizamos un bucle, donde realizamos las siguientes acciones:
DO

- Comprobar si la pila tiene datos
- Sacamos el último estado introducido en la pila STACK (LIFO)
- Analizamos el STATE N que hemos obtenido con stack.pop()
- IF NO genera hijos tenemos la solución que andamos buscando.
- ELSE IF Guardamos los hijos en la STACK y bajamos un nivel en el espacio de búsqueda.
- IF llegamos a la profundidad que se ha indicado y no hemos encontrado la solución retrocedemos para buscar la solución por otro camino.

WHILE solution KO

2.2.2 Heurística y búsqueda informada

Vamos a utilizar una función de evaluación para ir comparando los estados hasta llegar a la solución más óptima.

Aplicamos directamente acciones, para no tener que usar copias en profundidad, ya que no es necesario generar todo el espacio de búsqueda.

La función de evaluación va a determinar la reducción de nodos que aplica la acción correspondiente sobre el nodo padre.

La mejor evaluación indicará la mejor acción que podemos aplicar.

Realizamos una búsqueda en profundidad, recorriendo cada uno de los hijos o estados tras aplicar las acciones correspondientes.

A cada estado (StrategyTree) se le aplica una lista de posibles acciones, y cada una de estas acciones generara un nuevo estado (hijo del estado actual).

A estos estados generados se le aplica una evaluación para quedamos con el que mejor cumpla con los requisitos evaluados, en nuestro caso el que tenga menor número de nodos tras aplicar alguna de las acciones.

Características:

- El proceso se irá realizando hasta llegar a la solución que buscamos o llegar a una profundidad máxima en la generación del espacio de búsqueda.
- El espacio de búsqueda que vamos creando no se almacena en memoria por lo que conseguimos una mejora con respecto a los algoritmos de búsqueda no informada.
- Se consiguen mejores tiempos de respuesta.

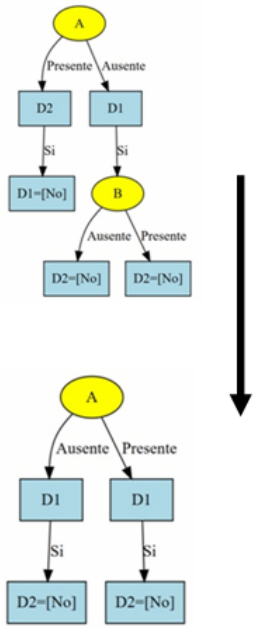

Figure 2.33: Estrategia de búsqueda informada con heurística.

Partiendo del ejemplo mutilizado para generar el espacio de búsqueda tendremos los siguientes estados tras aplicar las correspondientes acciones al Estado 0:

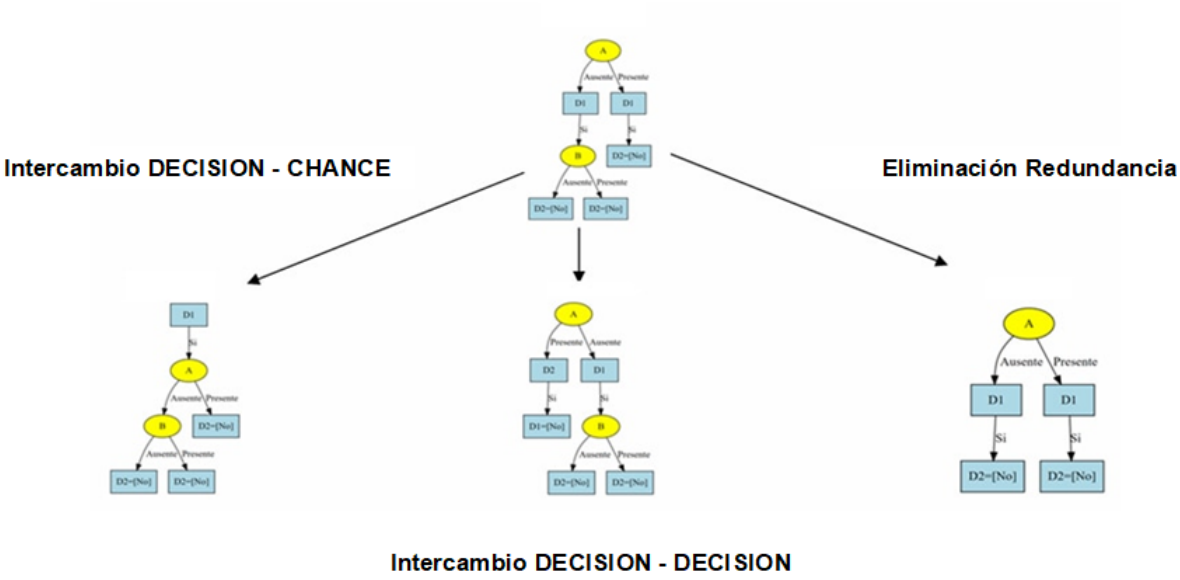

Figure 2.32: Acciones aplicadas al Estado 0 (StrategyTree Ejemplo).

Si aplicamos una función de evaluación que se quede con el estado que tenga menor número de nodos, nos quedaremos con el genedaro tras aplicar la acción de eliminación de redundancia.

Si aplicamos a este estado la acción que mejor resultado muestre tras aplicar la acción de evaluación, tendremos:

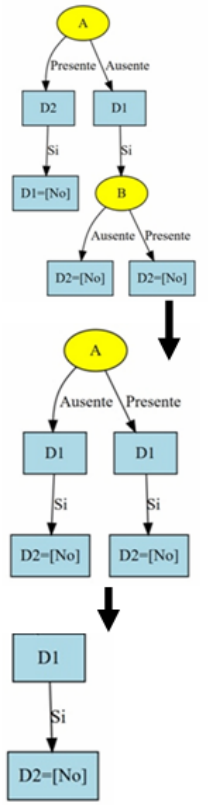

Figure 2.34: Ejemplo de estrategia de búsqueda informada con heurística.

Con la función de evaluación en la búsqueda en profundidad conseguimos reducir los tiempos de respuesta.

2.3 Resultados y Discusión

Vamos a realizar dos tipos de pruebas:

- Problemas de decisión unicriterio.
- Problemas de decisión coste-efectividad.

Problemas de decisión unicriterio

Vamos a realizar pruebas sobre las siguientes redes:

- ID-mediastinet-ce.pgmx
- ID-mediastinet-first-ebus-ce.pgmx
- ID-mediastinet-first-eus-ce.pgmx
- ID-arthronet.pgmx

Problemas de decisión coste-efectividad

Las redes DAN con las que vamos a trabajar son:

- DAN-decide-test-2therapies.pgmx
- DAN-2tests.pgmx
- DAN-3tests.pgmx
- DAN-CE-4-test-problem.pgmx

2.3.1 Problemas de decisión unicriterio

A continuación vamos a valorar cada uno de las estrategias de búsqueda propuestas para decidir cual podemos aplicar finalmente en nuestro proceso de optimización.

ID-mediastinet-ce

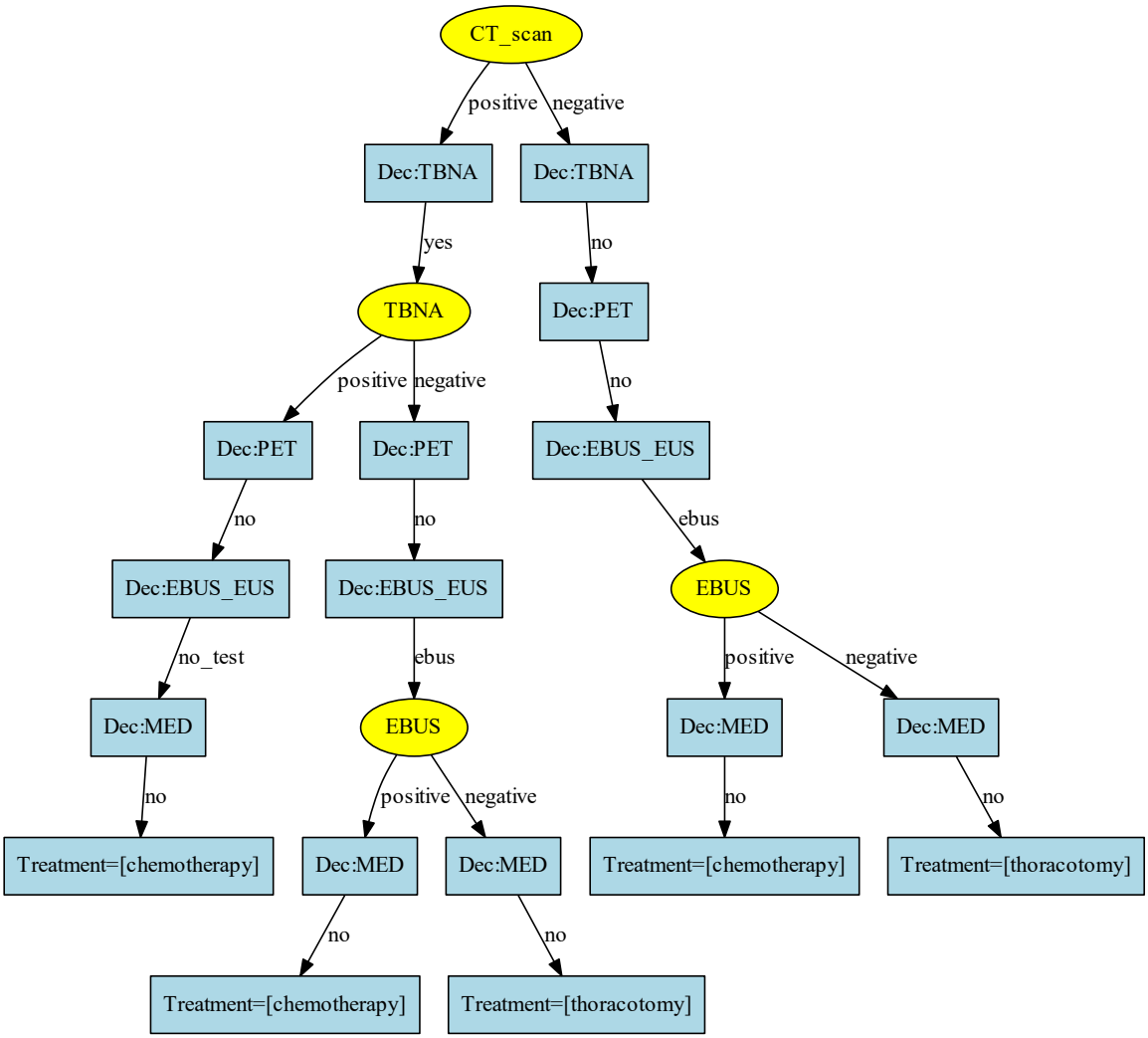

Figure 2.35: StrategyTree ID-mediastinet-ce.

Aplicando las estrategias de búsqueda para la optimización, tendremos:

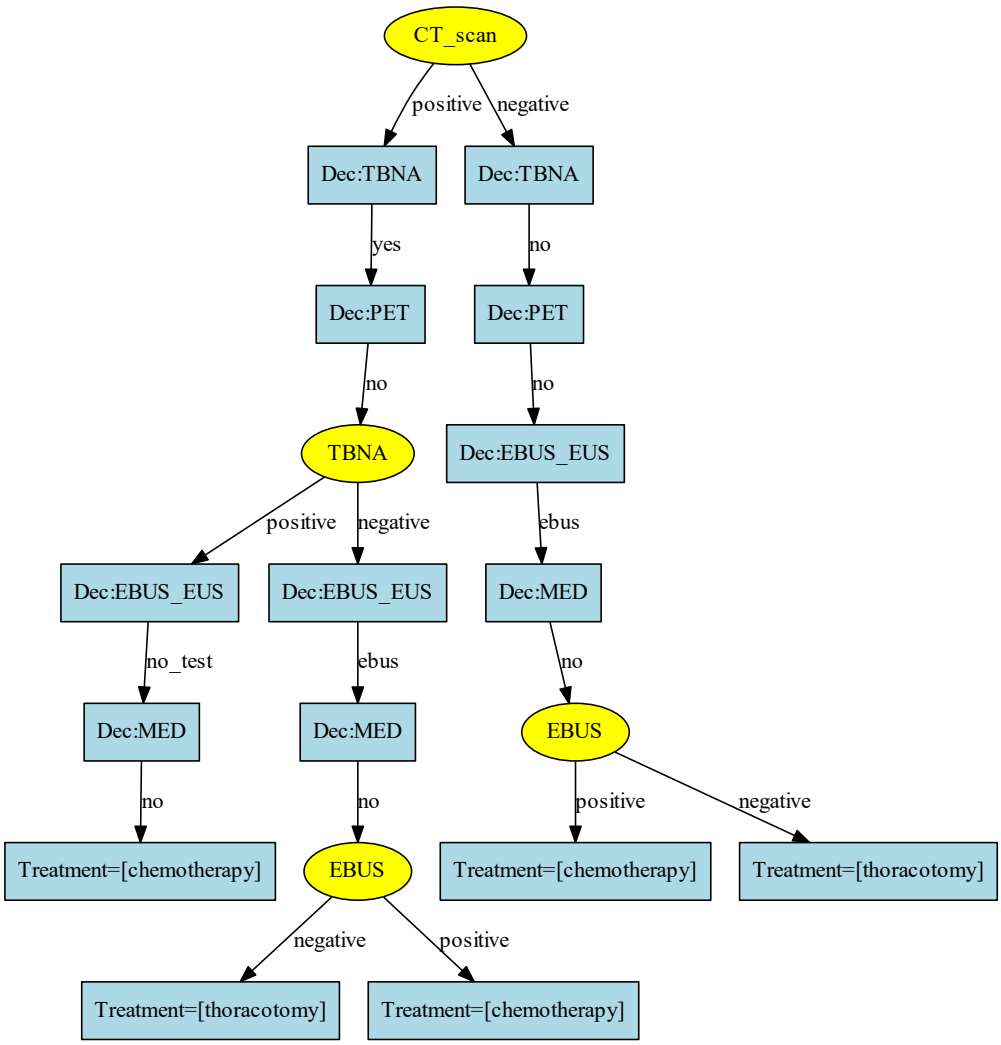

Figure 2.36: StrategyTree ID-mediastinet-ce Optimizado.

Nodos Iniciales	Nodos Finales	% Optimización
22	19	14

Figure 2.37: Resultados optimización StrategyTree ID-mediastinet-ce.

Aplicando el proceso de optimización en la red Arthronet, vemos con mayor claridad la importancia de la optimización en este nuevo proceso de OpenMarkov.

Arthronet

Debido al tamaño que ocupa la red la hemos dividido en tres partes para poder apreciar mejor el proceso.

La red Arthronet tienen como nodo raiz Alergia ATB.

Figure 2.38: Nodo raiz Arthronet.

De este nodo parten dos estados “no” y “si”.

Del estado “no” llegamos al nodo diabetes con los estados “ausente” y “presente”.

Vamos a mostrar los siguientes nodos de Arthronet:

- Nodo1 de Arthronet, que se corresponde con el estado “si” de Alergia ATB.

- Nodo 2 correspondiente al estado “ausente” del nodo diabetes.

- Nodo3 correspondiente al estado “presente” del nodo diabetes.

Arthronet - Nodo 1

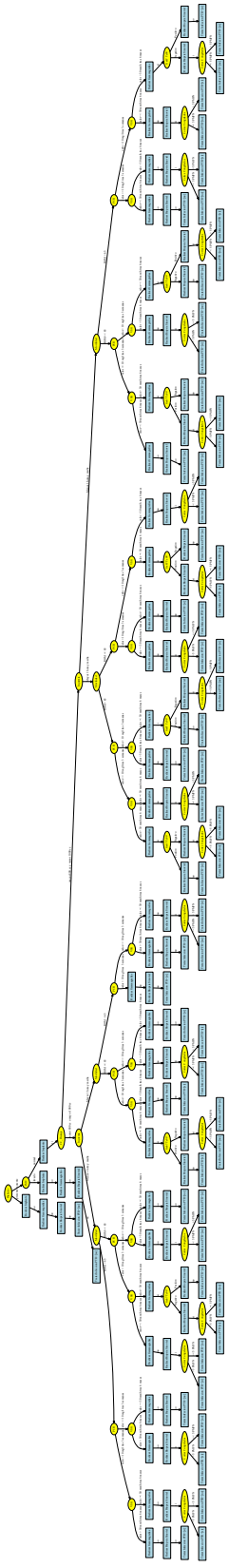

Figure 2.39: StrategyTree Arthronet - Nodo 1.

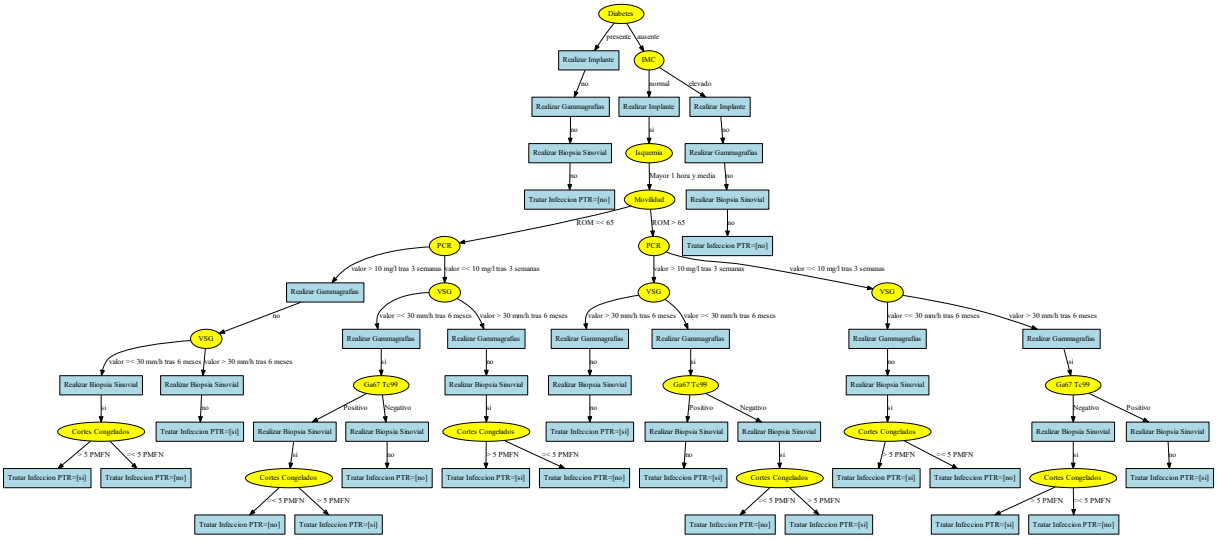

Figure 2.40: StrategyTree Arthronet - Nodo 1 Optimizado.

Arthronet - Nodo 2

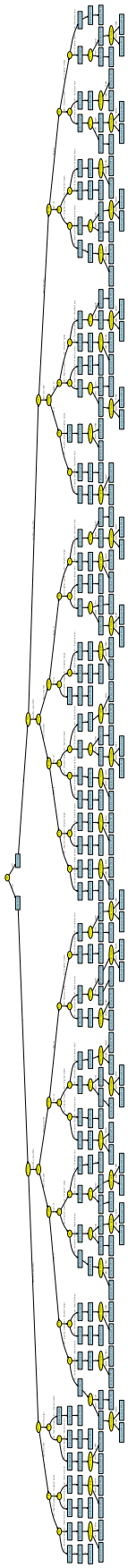

Figure 2.41: StrategyTree Arthronet - Nodo 2.

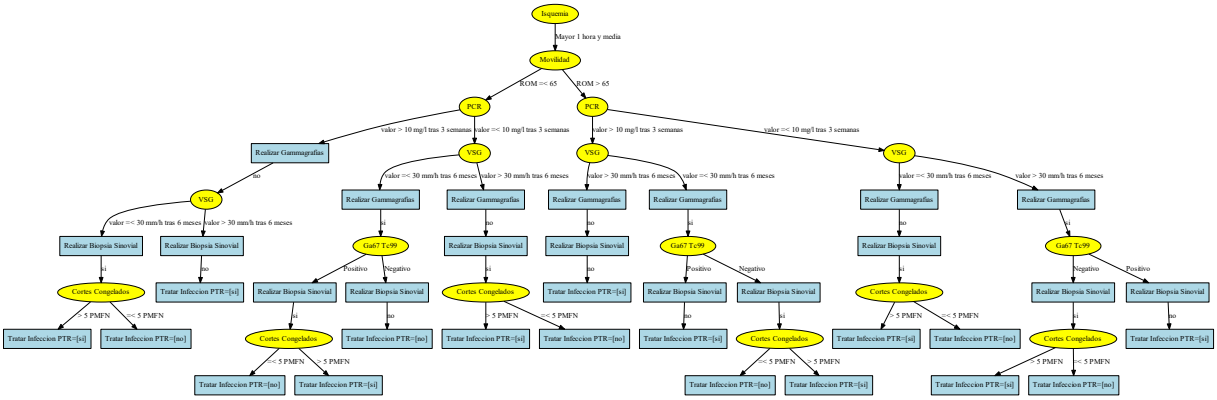

Figure 2.42: StrategyTree Arthronet - Nodo 2 Optimizado

Arthronet - Nodo 3

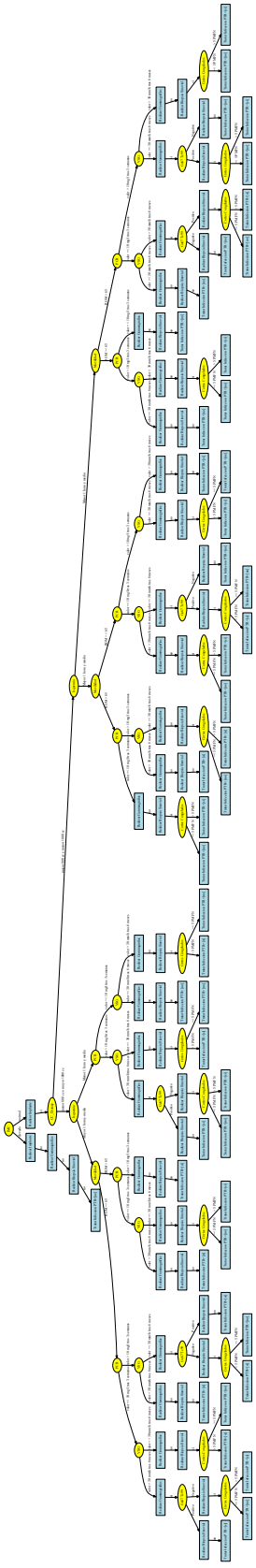

Figure 2.43: StrategyTree Arthronet - Nodo 3

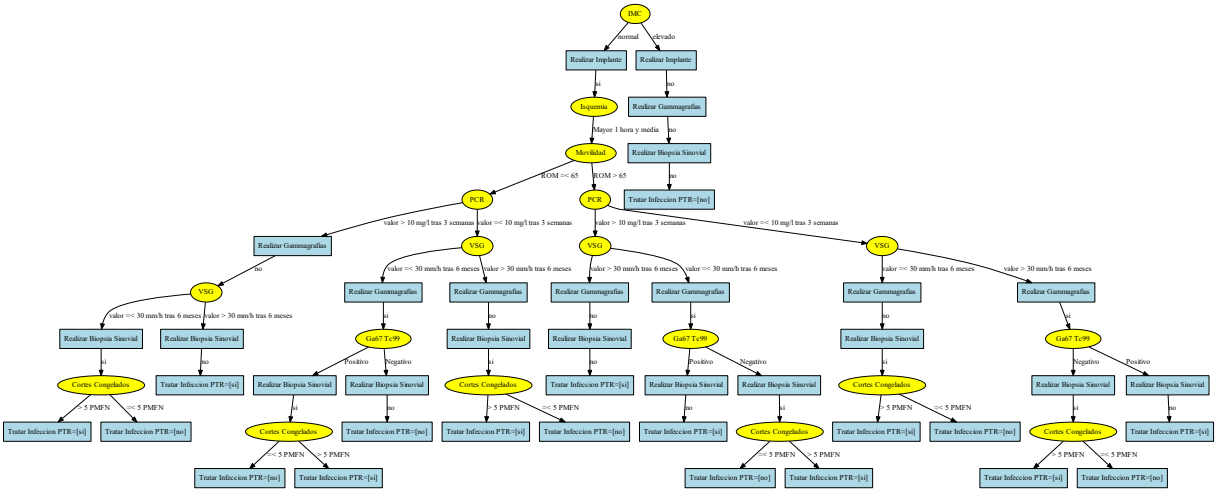

Figure 2.44: StrategyTree Arthronet - Nodo 2 Optimizado

Nodos Inicial	Nodos Final	% Optimización
680	398	42%

Figure 2.45: Resultados optimización StrategyTree Arthronet.

2.3.1.1 Búsqueda en anchura BFS

Obtenemos los siguientes resultados:

Red	Tiempo (ms)	%Optimización	Número de nodos inicial	Número de nodos final
ID-mediastinet-ce.pgmx	257	14%	22	19
ID-mediastinet-first-ebus-ce.pgmx	171	10%	32	29
ID-mediastinet-first-eus-ce.pgmx	151	13%	24	21
ID-arthronet.pgmx	$>$ 15 minutos	42%	680	398

Table 2.1: Resultados de aplicar el algoritmo búsqueda en anchura.

Para redes grandes como Arthronet los tiempos son muy elevados y el rendimiento del algoritmo no es muy apropiado al consumir bastante memoria por el uso de copias en profundidad a la hora de generar nuevos estados.

2.3.1.2 Búsqueda en profundidad DFS

Probando con las cuatro redes de openmarkov que hemos utilizado en el algoritmo anterior, obtenemos los siguientes resultados:

Red	Tiempo(ms)	%Optimización	Nodos inicial	Nodos final
ID-mediastinet-ce.pgmx	141	14%	22	19
ID-mediastinet-first-ebus-ce.pgmx	110	10%	32	29
ID-mediastinet-first-eus-ce.pgmx	120	13%	24	21
ID-arthronet.pgmx	6182	42%	680	398

Table 2.2: Resultados de aplicar el algoritmo búsqueda en profundidad.

Los tiempos con respecto a la búsqueda en anchura son mucho menores y se aprecia aún más en la red arthronet que es la que tiene un mayor número de nodos y ramas.

2.3.1.3 Búsqueda informada (Heurística)

Los resultados que obtenemos con las redes que estamos analizando para las pruebas son:

Red	Tiempo(ms)	%Optimización	Nodos inicial	Nodos final
ID-mediastinet-ce.pgmx	78	14%	22	19
ID-mediastinet-first-ebus-ce.pgmx	88	10%	32	29
ID-mediastinet-first-eus-ce.pgmx	92	13%	24	21
ID-arthronet.pgmx	433	42%	680	400

Table 2.4: Resultados de aplicar estrategia de búsqueda con heurística.

La reducción de tiempos es bastante considerable con respecto a los algoritmos anteriores.

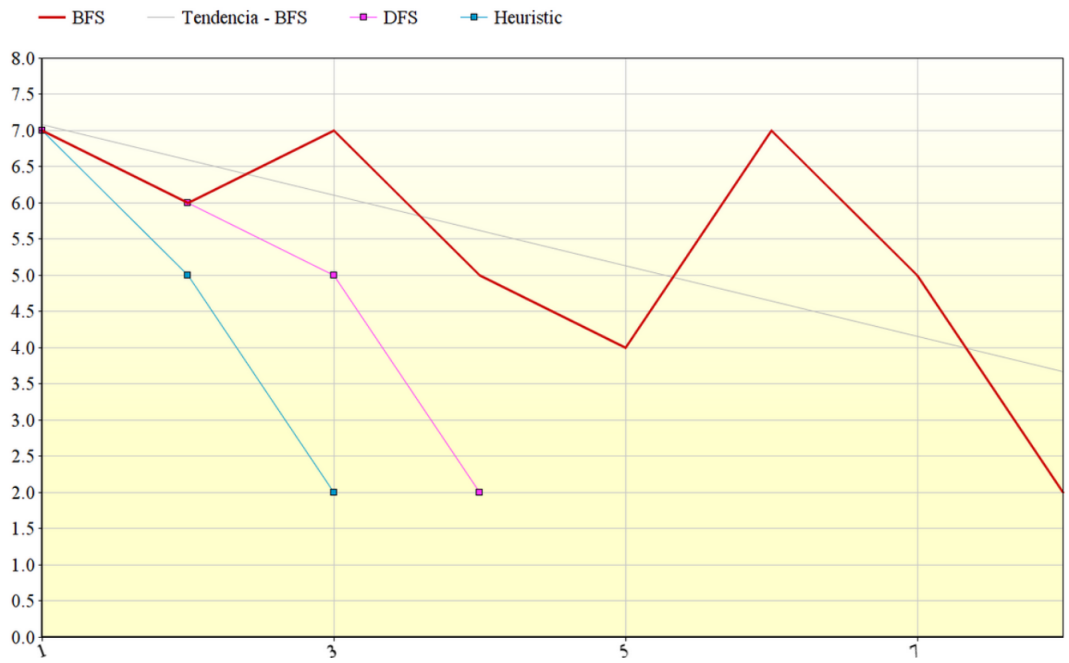

Figure 2.46: Grafico comportamiento estrategias de búsqueda.

Conclusiones:

- Se mejoran los tiempos de respuesta.
- No se genera espacio en memoria al no realizar copias en profundidad.

2.3.2 Problemas de decisión coste-efectividad

Una vez que tenemos el proceso de optimización con la estrategia de búsqueda con heurística vamos a realizar pruebas sobre redes DAN.

2.3.2.1 DAN-decide-test-2therapies.pgmx

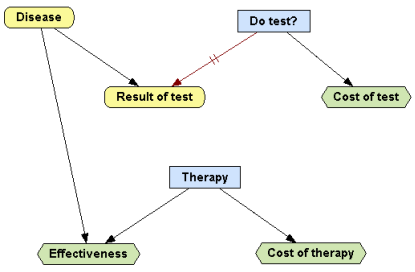

Figure 2.47: ID-CEA-test-2therapies.

Mediante el uso de la herramienta de OpenMarkov se aplica un análisis de coste – efectividad: análisis determinista, obteniendo los siguientes resultados:

λ mínimo	λ máximo	Coste	Efectividad	Árbol decisión (StrategyTree)
0.0	11171.3	0.0	8.768	Do test? = no -> Therapy = no therapy
11171.3	33383.5	3874.0	9.11478	Do test? = yes -> IF Result of test = negative -> Therapy = no therapy IF Result of test = positive -> Therapy = therapy 1
33383.5	Infinito	13184.0	9.39366	Do test? = yes -> IF Result of test = negative -> Therapy = no therapy IF Result of test = positive -> Therapy = therapy 2

Table 2.5: Resultados del análisis coste-efectividad.

Cada intervalo de λ mostrará el coste, efectividad y árbol de decisión correspondiente a dicho intervalo.

Con los datos de la tabla se diseña un árbol de decisión con nodo raíz λ.

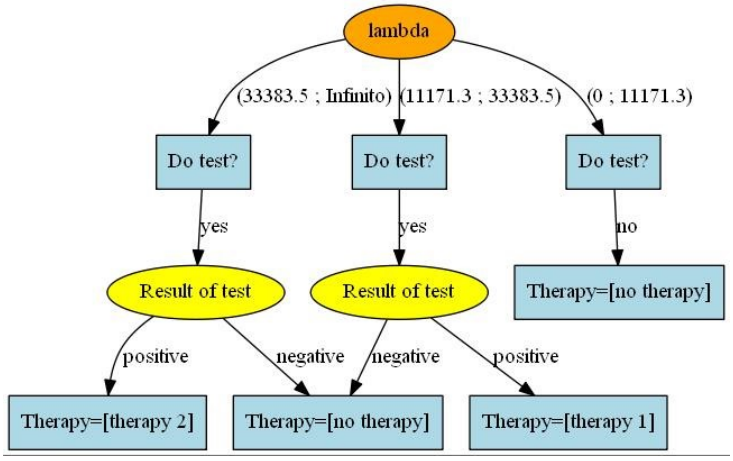

Figure 2.48: Árbol decisión con nodo raíz λ y como ramas cada intervención.

Una vez que tenemos el árbol aplicamos el proceso de optimización, de esta manera obtendremos un árbol equivalente al original, pero mucho más simplificado y por tanto más útil de analizar.

El resultado final para el ejemplo que estamos analizando será:

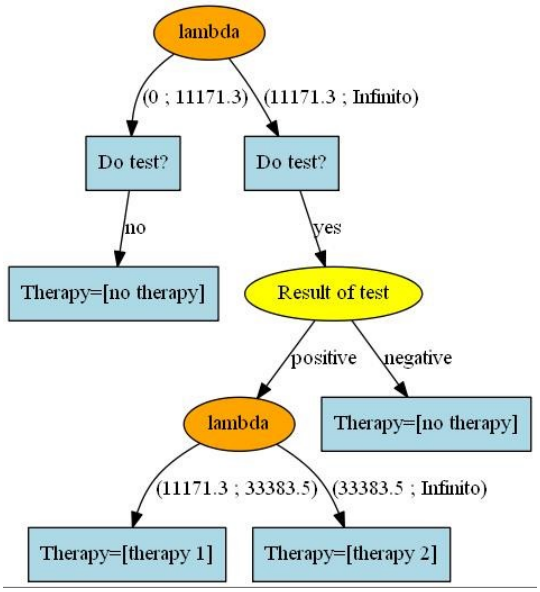

Figure 2.49: Resultado al reorganizar el árbol.

Este es un ejemplo sencillo, pero cuando disponemos de árboles mucho más complicados, esta optimización representa una gran ayuda para elegir la decisión más acertada en cada situación.

Vamos a ver el resultado aplicado en otros DAN tomados como ejemplo y donde iremos aumentando su complejidad.

2.3.2.2 DAN-2tests.pgmx

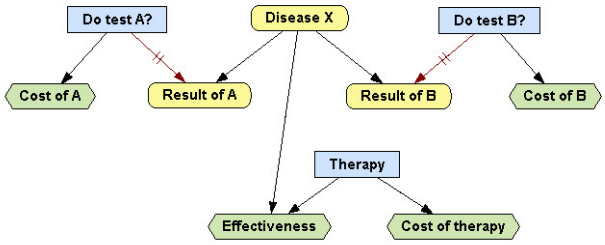

Figure 2.50: DAN-2tests.

Aplicando a través de OpenMarkov el análisis CEA determinista, obtenemos los siguientes resultados:

λ inf.	λ sup.	Coste	Efectividad	Intervención
0.0	7718.95	0.0	8.768	OD = Do test A? -> Do test A? = no -> Do test B? = no -> Therapy = no therapy
7718.95	21385.5	2119.95	9.04264	OD = Do test A? -> Do test A? = yes -> IF Result of A = negative -> Do test B? = no -> Therapy = no therapy IF Result of A = positive -> Do
21385.5	24361.7	7304.85	9.28509	OD = Do test A? -> Do test A? = yes -> IF Result of A = negative -> Do test B? = no -> Therapy = no therapy IF Result of A = positive -> Do
24361.7	71550.3	9062.25	9.35723	OD = Do test B? -> Do test B? = yes -> IF Result of B = negative -> Do test A? = no -> Therapy = no therapy IF Result of B = positive -> Do
71550.3	113139.0	10734.9	9.38061	OD = Do test A? -> Do test A? = yes -> IF Result of A = negative -> Do test B? = yes -> IF Result of B = negative -> Therapy = no therapy
113139.0	+∞	14856.7	9.41704	OD = Do test B? -> Do test B? = yes -> IF Result of B = negative -> Do test A? = yes -> IF Result of A = negative -> Therapy = no therapy

Figure 2.51: Análisis CEA determinista en DAN-2tests.

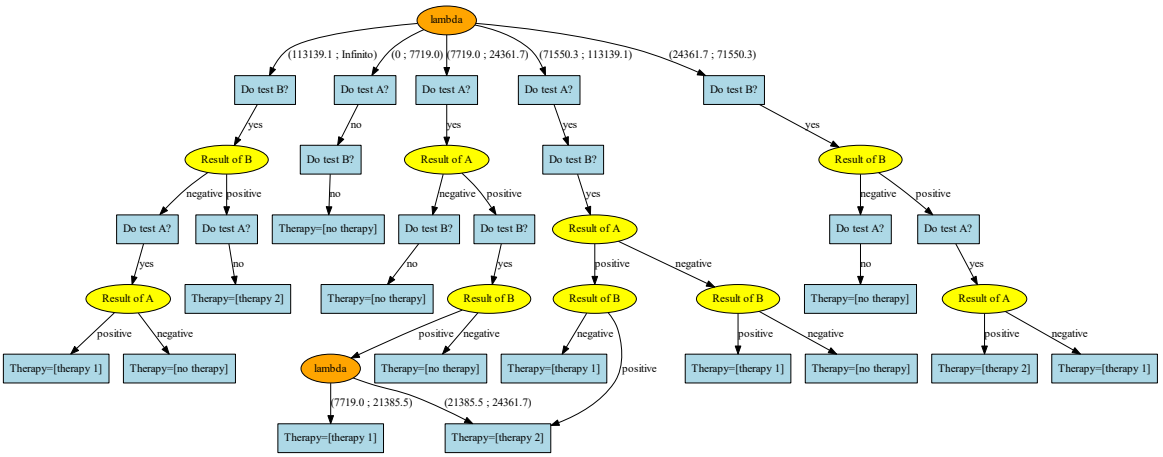

Figure 2.53: StrategyTree DAN-2tests Optimizado.

Con estos datos generamos el árbol de decisión resultado del análisis CEA y que posteriormente optimizaremos.

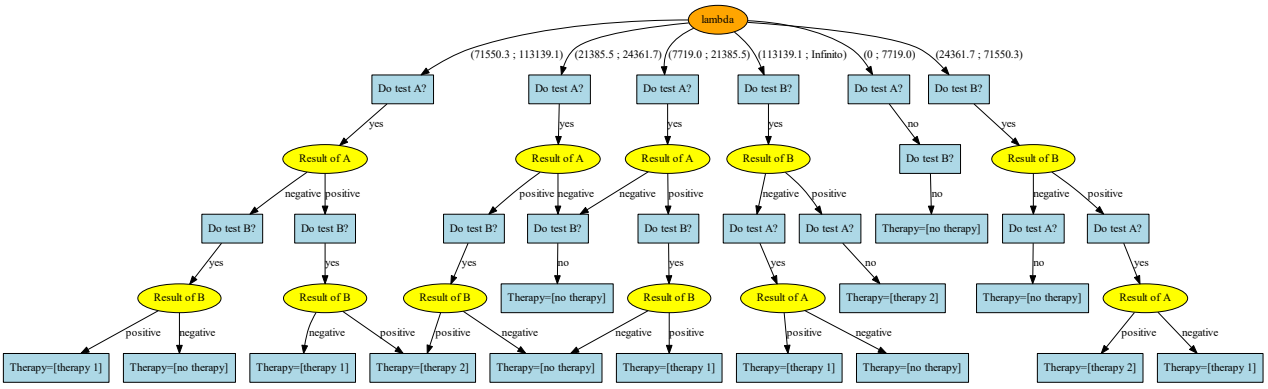

Figure 2.52: StrategyTree DAN-2tests.

Aplicamos el proceso de optimización y tendremos el siguiente resultado:

El siguiente ejemplo representa un árbol más complejo y donde se puede observar mejor las ventajas de la optimización para poder analizarlo.

λ inf.	λ sup.	Coste	Efectividad	Intervención
0.0	7518.2	0.0	8.768	OD = Do test C? -> Do test C? = no -> OD = Do test A? (DA) -> Do test A? (DA) = no -> Do test B? (DB) = no -> Therapy (T) ...
7518.2	8866.28	1757.33	9.00174	OD = Do test A? (DA) -> Do test A? (DA) = yes -> IF Result of A (RA) = negative -> OD = Do test C? -> Do test C? = no -> Do..
8866.28	14232.0	2119.95	9.04264	OD = Do test C? -> Do test C? = no -> OD = Do test A? (DA) -> Do test A? (DA) = yes -> IF Result of A (RA) = negative -> Do..
14232.0	20534.7	3035.49	9.10697	OD = Do test B? (DB) -> Do test B? (DB) = yes -> IF Result of B (RB) = negative -> OD = Do test C? -> Do test C? = no -> Do ...
20534.7	26466.2	7304.03	9.31484	OD = Do test B? (DB) -> Do test B? (DB) = yes -> IF Result of B (RB) = negative -> OD = Do test C? -> Do test C? = no -> Do ...
26466.2	26500.1	7934.73	9.33867	OD = Do test C? -> Do test C? = yes -> IF Result of C = negative -> OD = Do test A? (DA) -> Do test A? (DA) = yes -> IF Res..
26500.1	40776.0	8851.09	9.37325	OD = Do test A? (DA) -> Do test A? (DA) = yes -> IF Result of A (RA) = negative -> OD = Do test C? -> Do test C? = yes -> I...
40776.0	110133.0	10850.9	9.4223	OD = Do test A? (DA) -> Do test A? (DA) = yes -> IF Result of A (RA) = negative -> OD = Do test C? -> Do test C? = yes -> I...
110133.0	118199.0	11710.8	9.4301	OD = Do test B? (DB) -> Do test B? (DB) = yes -> IF Result of B (RB) = negative -> OD = Do test A? (DA) -> Do test A? (DA) =...
118199.0	149957.0	11710.8	9.4301	OD = Do test A? (DA) -> Do test A? (DA) = yes -> IF Result of A (RA) = negative -> OD = Do test B? (DB) -> Do test B? (DB) =..
149957.0	+∞	13247.6	9.44035	OD = Do test A? (DA) -> Do test A? (DA) = yes -> IF Result of A (RA) = negative -> OD = Do test B? (DB) -> Do test B? (DB) =..

Figure 2.55: Análisis CEA determinista en DAN-3tests.

2.3.2.3 DAN-3tests.pgmx

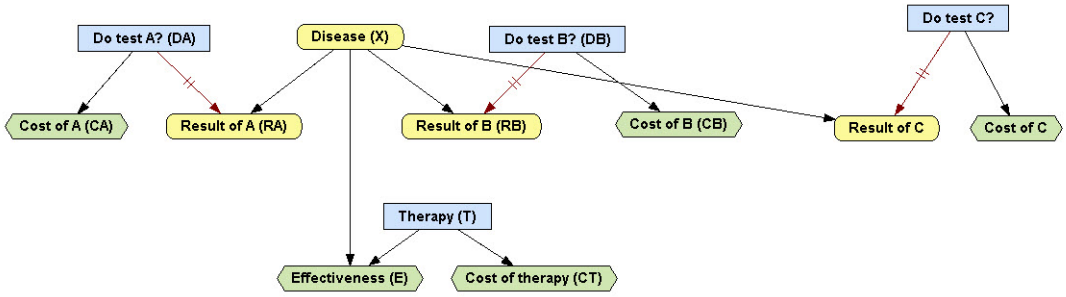

Figure 2.54: DAN-3tests.

Aplicando CEA determinista, obtendremos:

El árbol inicial que generamos con cada intervalo de λes:

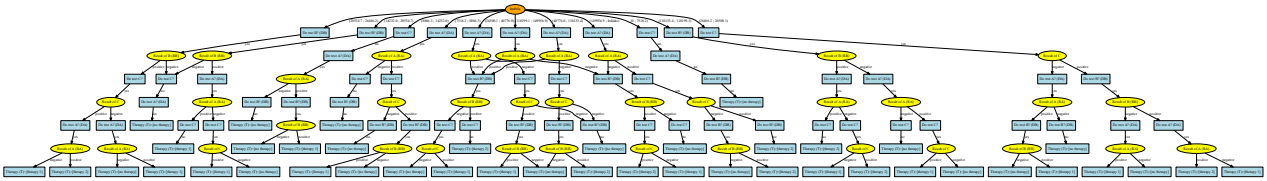

Figure 2.56: StrategyTree DAN-3tests.

El resultado tras aplicar la optimización es:

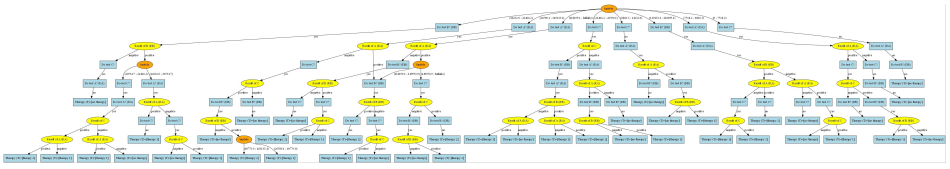

Figure 2.57: StrategyTree DAN-3tests Optimizado.

Para poder visualizar mejor el agrupamiento de ramas, vamos a mostrar las ramas que se agrupan por λ.

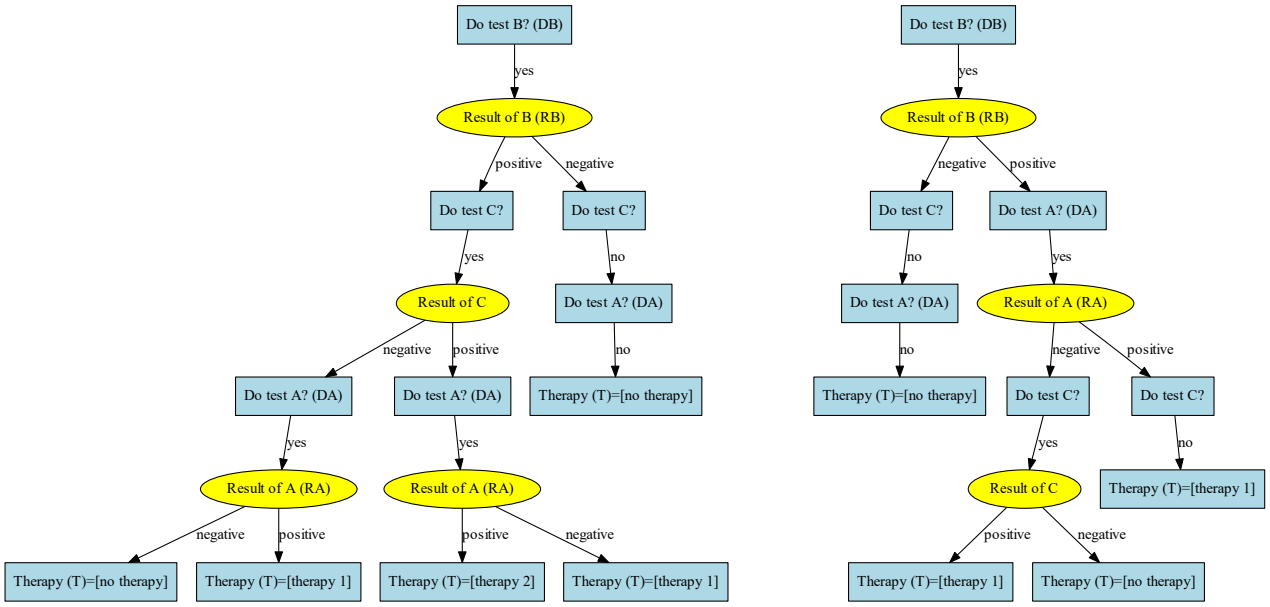

Figure 2.58: Ramas intervalos (142320-20534.7) y (20534.7-264662.2).

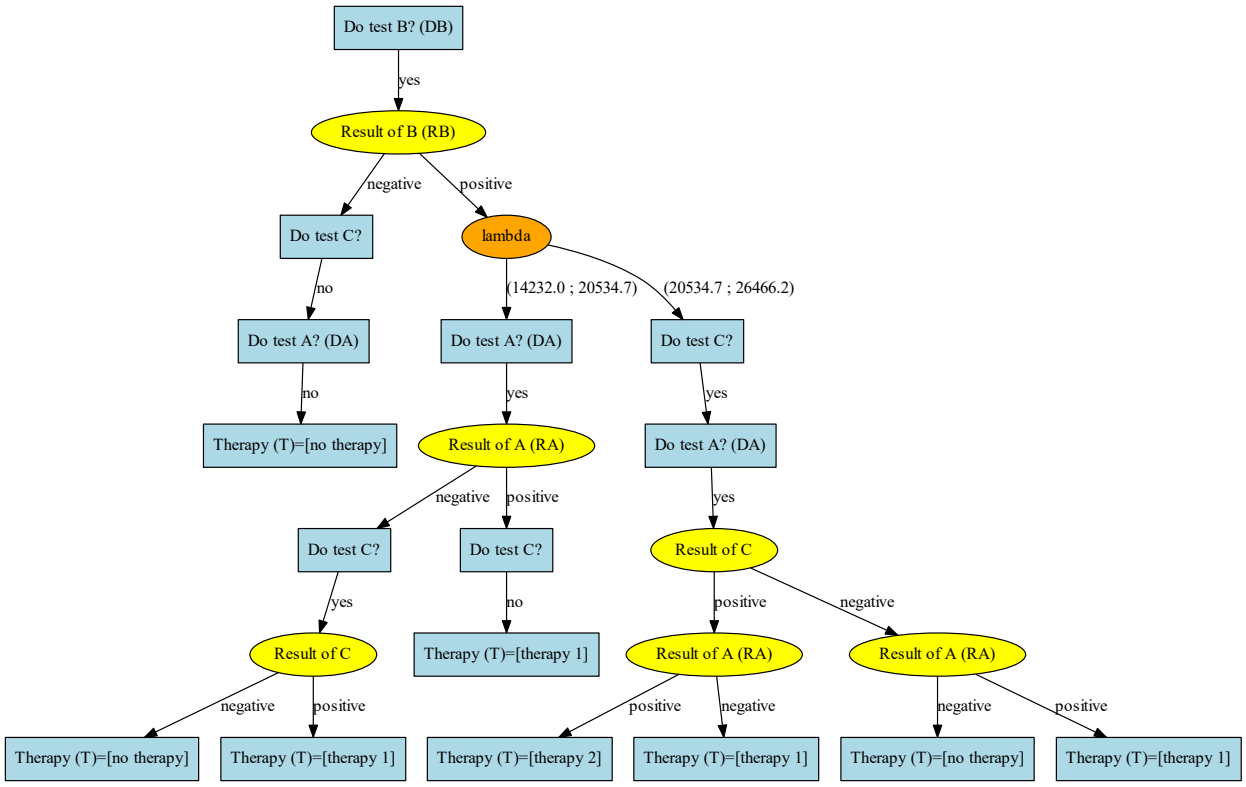

Figure 2.59: Nueva rama en el intervalo (142320 - 264662.2).

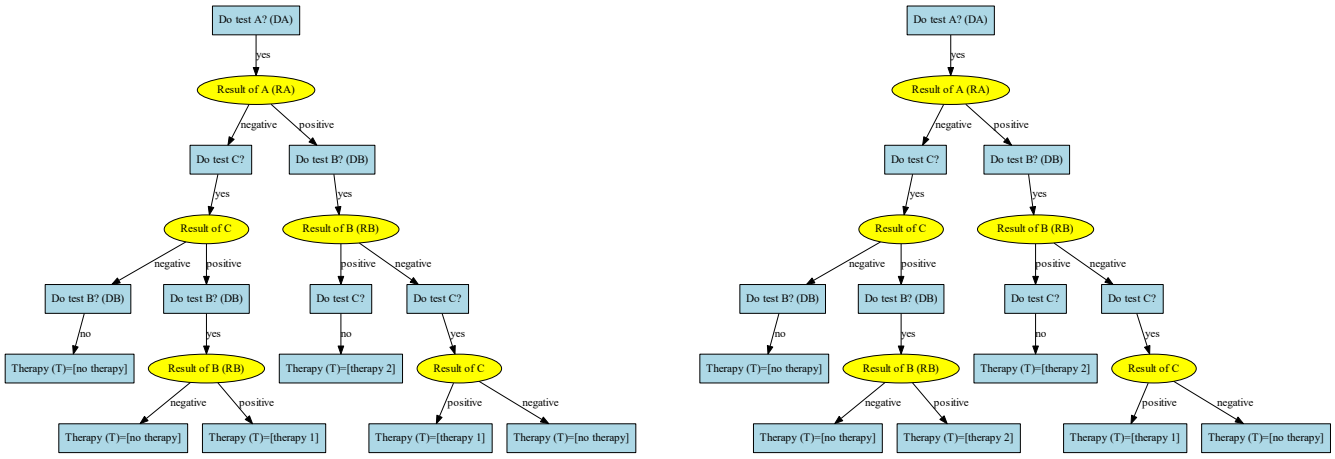

Figure 2.60: Ramas intervalos (26500.1-40776.0) y (40776.0-110133.4).

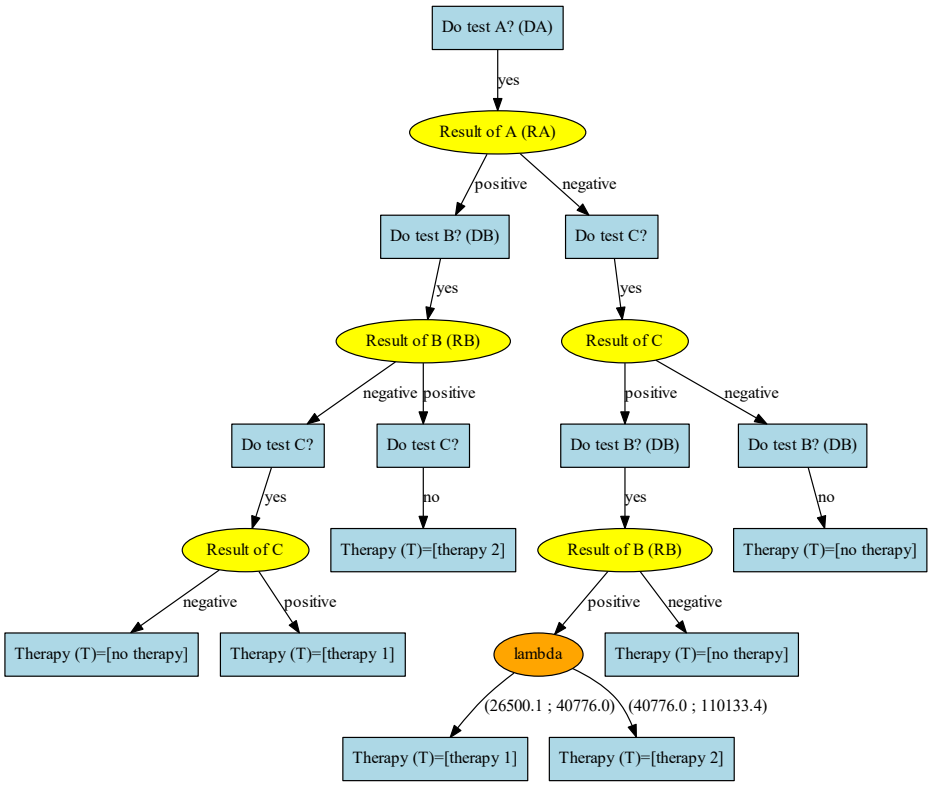

Figure 2.61: Nueva rama en el intervalo (26500.1 - 110133.4).

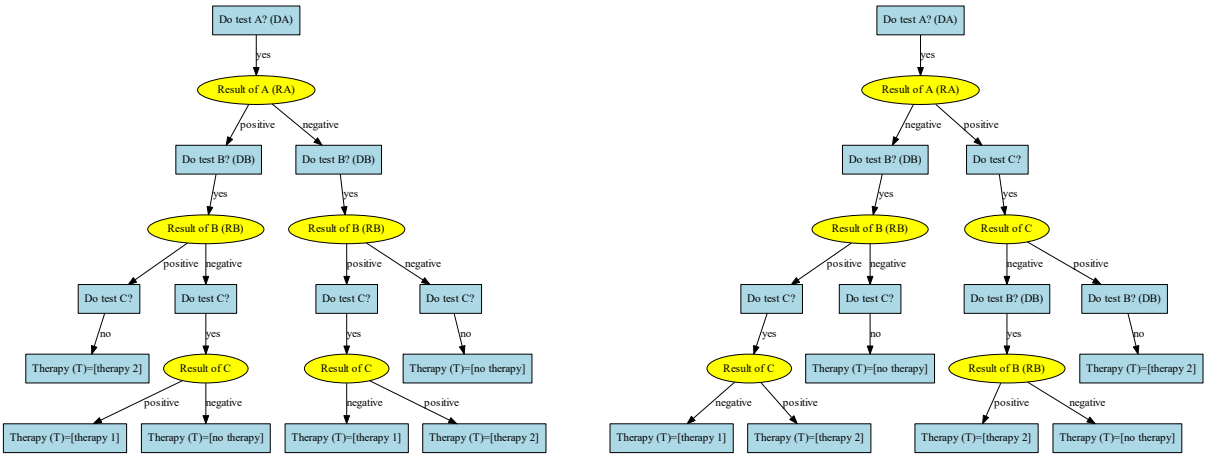

Figure 2.62: Ramas intervalos (118199.1-149956.9) y (149956.9-Infinito).

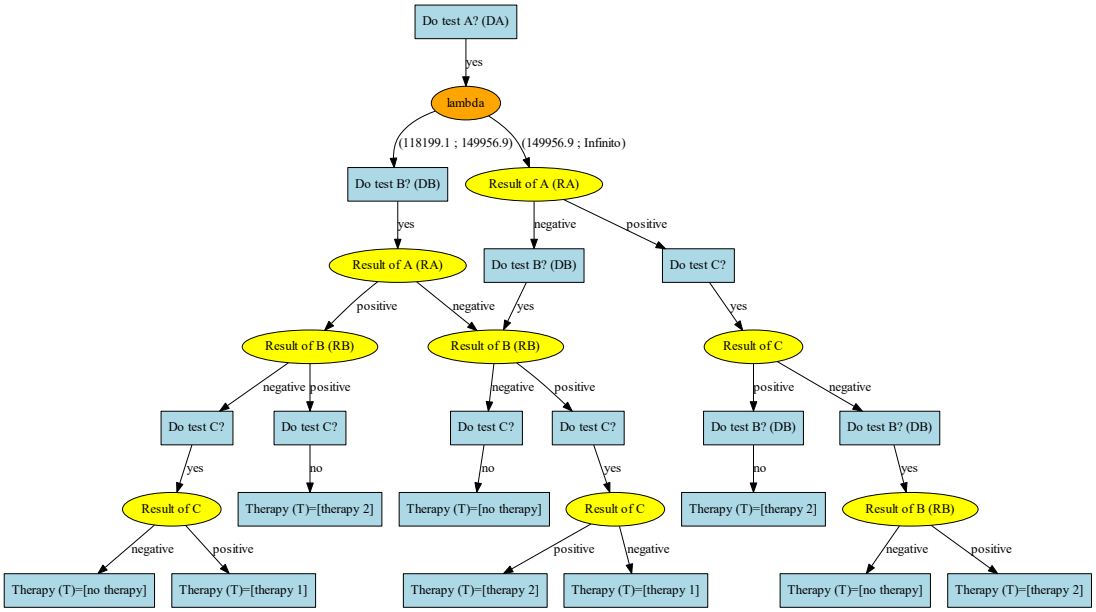

Figure 2.63: Nueva rama en el intervalo (118199.1- Infinito).

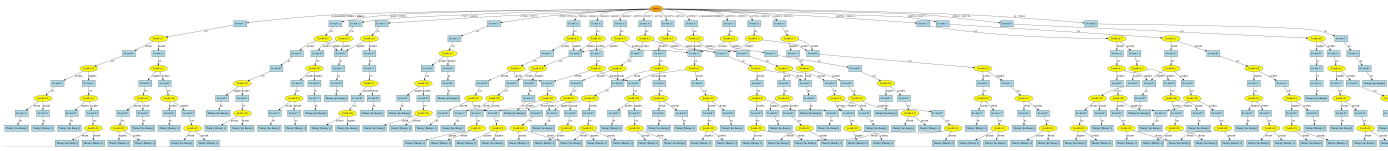

Figure 2.66: Árbol correspondiente a DAN-CE-4-test-problem.

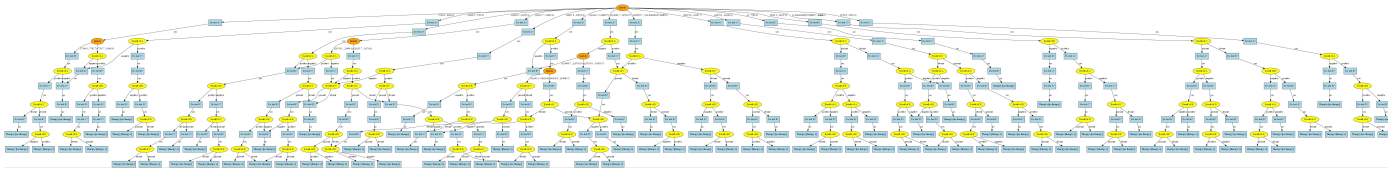

Figure 2.67: Optimización árbol DAN-CE-4-test-problem.

2.3.2.4 DAN-CE-4-test-problem.pgmx

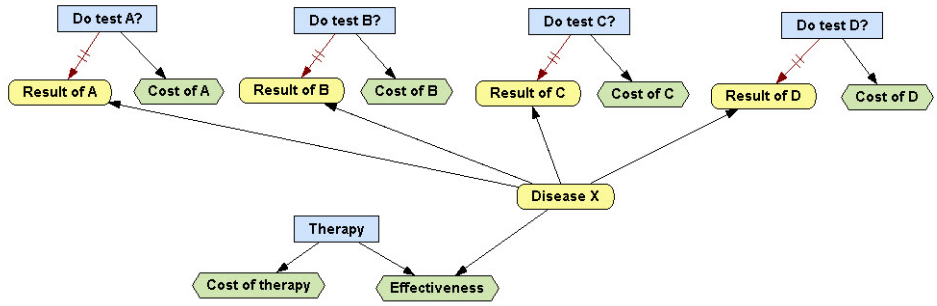

Figure 2.64: DAN-CE-4-test-problem.

Aplicando CEA determinista, obtendremos:

λ inf.	λ sup.	Coste	Efectividad	Intervención
0.0	7500.65	0.0	8.768	OD = Do test D? -> Do test D? = no -> OD = Do test C? -> Do test C? = no -> OD = Do test A? -> Do test A? = no -> Do test B? = no -> Therapy..
7500.65	7585.69	1856.43	9.0155	OD = Do test A? -> Do test A? = yes -> IF Result of A = negative -> OD = Do test D? -> Do test D? = no -> OD = Do test C? -> Do test C? = no ..
7585.69	8206.89	2044.72	9.04032	OD = Do test A? -> Do test A? = yes -> IF Result of A = negative -> OD = Do test D? -> Do test D? = no -> OD = Do test C? -> Do test C? = no ..
8206.89	9709.85	2044.72	9.04032	OD = Do test A? -> Do test A? = yes -> IF Result of A = negative -> OD = Do test D? -> Do test D? = no -> OD = Do test C? -> Do test C? = no ..
9709.85	9992.21	2044.72	9.04032	OD = Do test C? -> Do test C? = no -> OD = Do test A? -> Do test A? = yes -> IF Result of A = negative -> OD = Do test D? -> Do test D? = no ..
9992.21	11037.8	2732.45	9.10915	OD = Do test B? -> Do test B? = yes -> IF Result of B = negative -> OD = Do test D? -> Do test D? = no -> OD = Do test C? -> Do test C? = no -..
11037.8	14871.8	3028.39	9.13596	OD = Do test A? -> Do test A? = yes -> IF Result of A = negative -> OD = Do test C? -> Do test C? = yes -> IF Result of C = absent -> OD = D...
14871.8	20215.7	3063.29	9.13831	OD = Do test C? -> Do test C? = yes -> IF Result of C = absent -> OD = Do test A? -> Do test A? = yes -> IF Result of A = negative -> OD = D...
20215.7	21558.1	7484.94	9.35703	OD = Do test A? -> Do test A? = yes -> IF Result of A = negative -> OD = Do test C? -> Do test C? = yes -> IF Result of C = absent -> OD = D...
21558.1	23409.3	7999.51	9.3809	OD = Do test A? -> Do test A? = yes -> IF Result of A = negative -> OD = Do test C? -> Do test C? = yes -> IF Result of C = absent -> OD = D...
23409.3	32070.1	9398.78	9.44068	OD = Do test A? -> Do test A? = yes -> IF Result of A = negative -> OD = Do test C? -> Do test C? = yes -> IF Result of C = absent -> OD = D...
32070.1	32636.2	10095.8	9.46241	OD = Do test C? -> Do test C? = yes -> IF Result of C = absent -> OD = Do test A? -> Do test A? = yes -> IF Result of A = negative -> OD = D...
32636.2	68136.0	10321.5	9.46933	OD = Do test A? -> Do test A? = yes -> IF Result of A = negative -> OD = Do test B? -> Do test B? = yes -> IF Result of B = negative -> OD = ...
68136.0	114941.0	10419.0	9.47076	OD = Do test A? -> Do test A? = yes -> IF Result of A = negative -> OD = Do test B? -> Do test B? = yes -> IF Result of B = negative -> OD = ...
114941.0	228794.0	10651.1	9.47278	OD = Do test A? -> Do test A? = yes -> IF Result of A = negative -> OD = Do test B? -> Do test B? = yes -> IF Result of B = negative -> OD = ...
228794.0	315859.0	11168.0	9.47504	OD = Do test A? -> Do test A? = yes -> IF Result of A = negative -> OD = Do test C? -> Do test C? = no -> OD = Do test B? -> Do test B? = yes .
315859.0	3.11461E15	12215.7	9.47835	OD = Do test A? -> Do test A? = yes -> IF Result of A = negative -> OD = Do test C? -> Do test C? = yes -> IF Result of C = absent -> OD = D...
3.11461E15	+∞	12221.2	9.47835	OD = Do test C? -> Do test C? = yes -> IF Result of C = absent -> OD = Do test B? -> Do test B? = yes -> IF Result of B = negative -> OD = Do..

Figure 2.65: Análisis CEA determinista en DAN-CE-4-test-problem.

Árbol generado con cada intervalo de λ:

El resultado tras aplicar la optimización es:

Podemos observar las ramas agrupadas por λen las siguientes figuras:

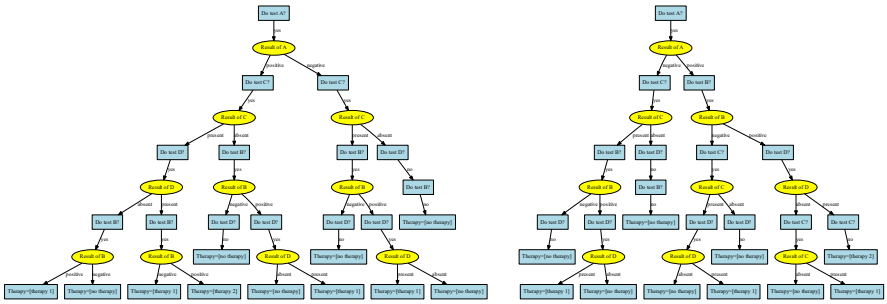

Figure 2.68: Ramas intervalos (20215.7-215581) y (21558.1-23409.3).

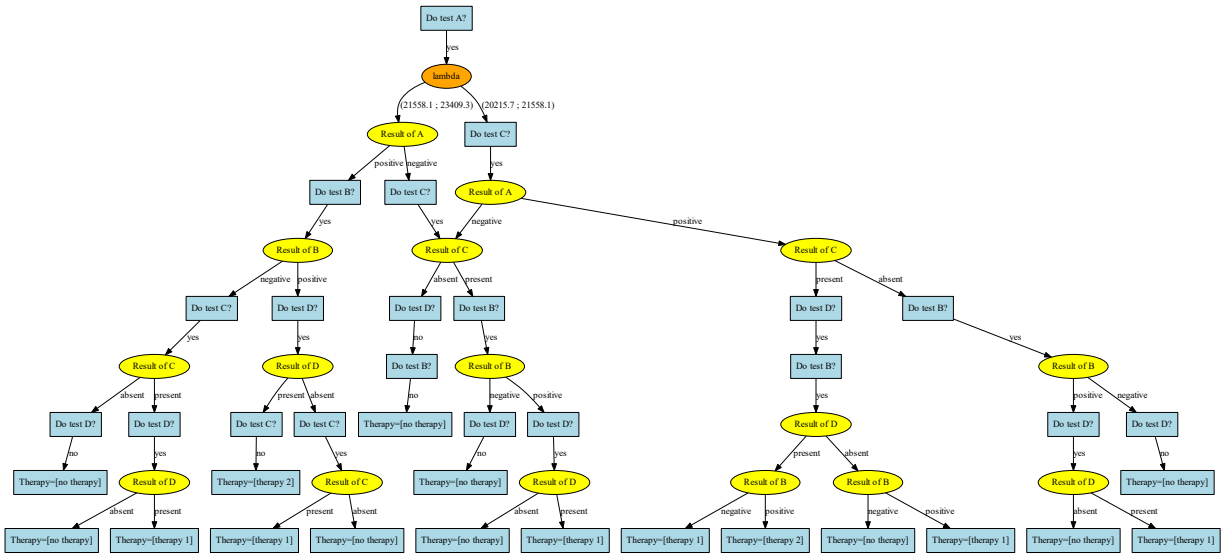

Figure 2.69: Nueva rama en el intervalo (20215.7- 23409.3).

El resto de ramas que tenemos tras al agrupación con λson:

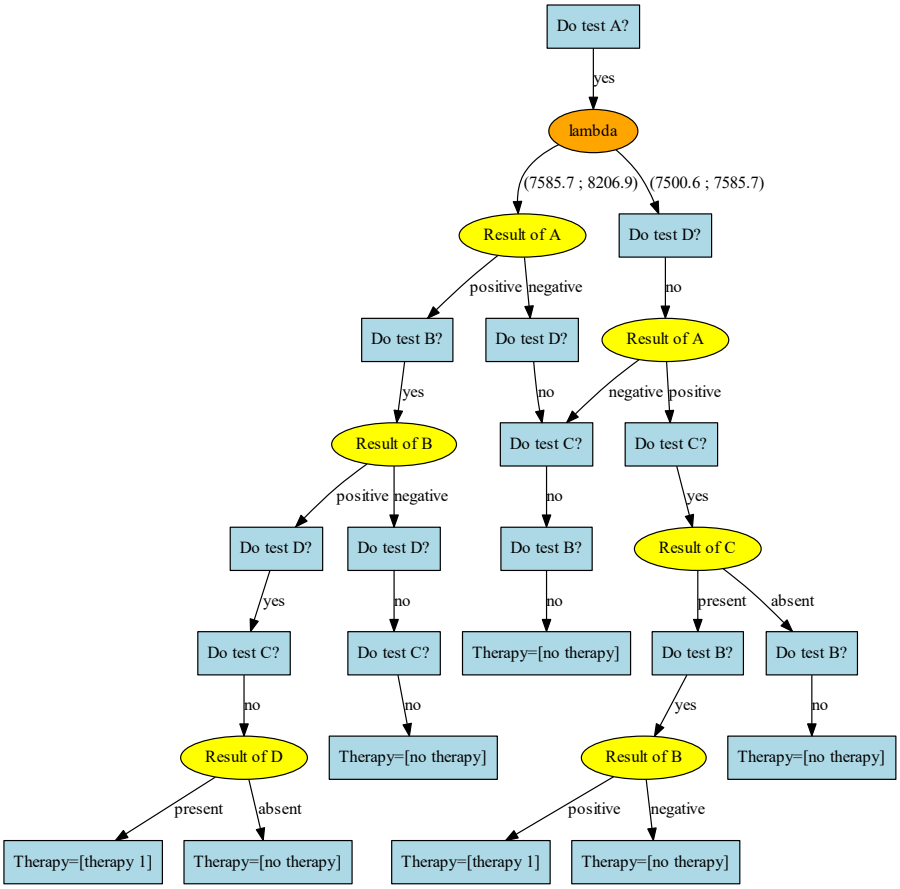

Figure 2.70: Nueva rama en el intervalo (75006-8206.9).

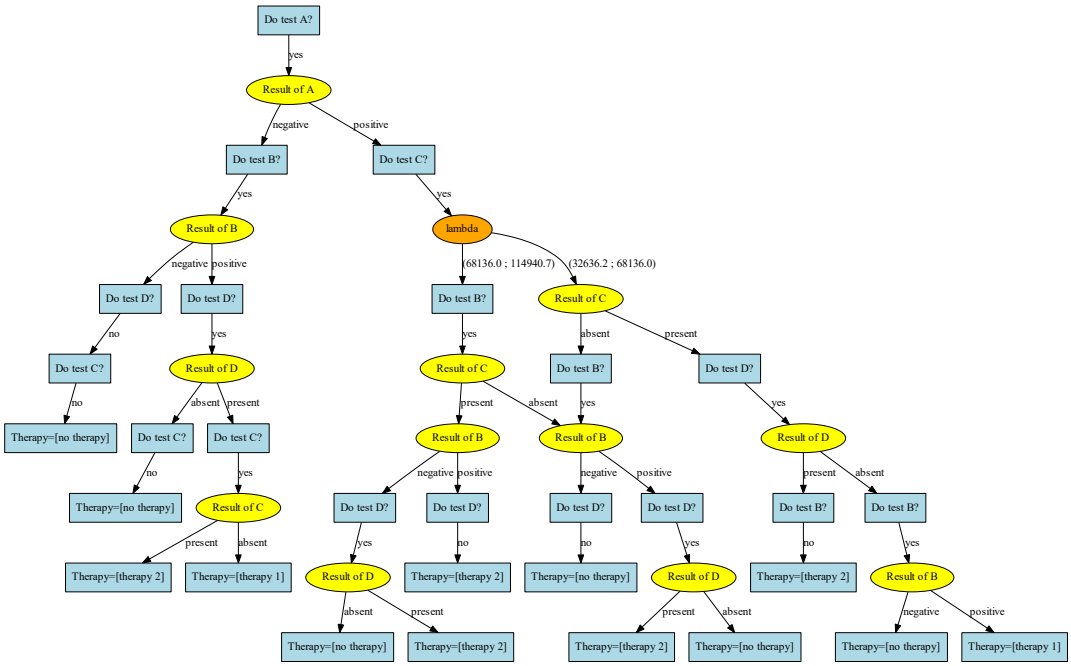

Figure 2.71: Nueva rama en el intervalo (326362-114940.7).

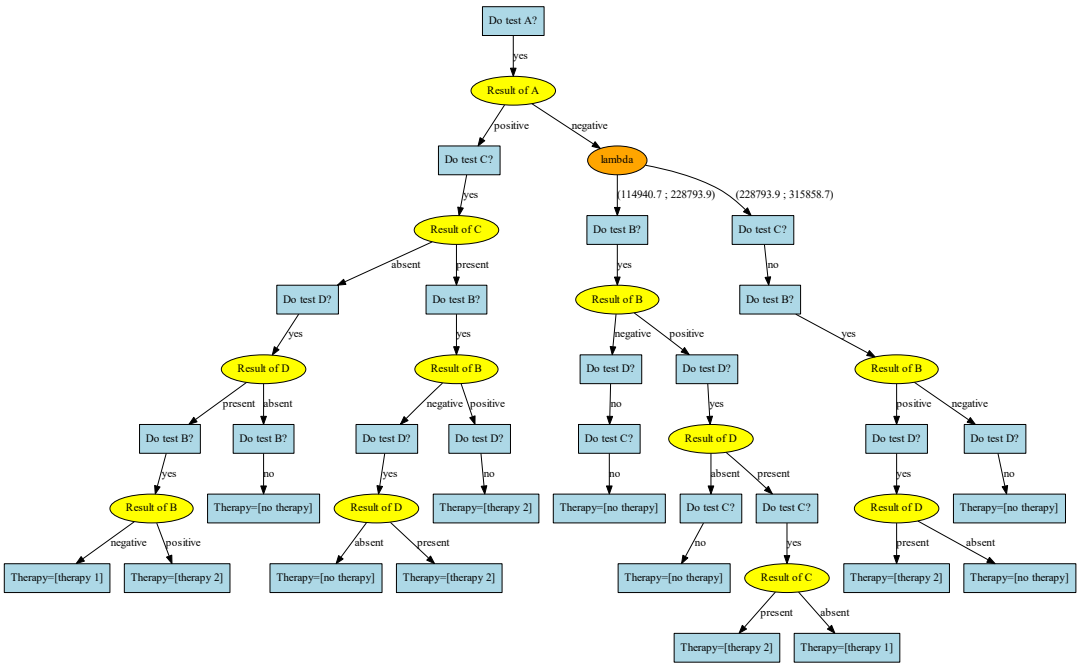

Figure 2.72: Nueva rama en el intervalo (1149407-315858.7).

Chapter 3

Conclusión y trabajos futuros

3.1 Conclusiones

La idea de este trabajo fin de máster surgió para poder simplificar árboles de decisión grandes a través de una nueva herramienta en OpenMarkov.

Algunas redes como Arthronet, mediastinet,... tienen como resultado árboles con muchas ramas, por lo que su simplificación puede ser muy útil en el momento de analizarlas.

Por este motivo se pensó en varios algoritmos que pudieran reducir el árbol original, pero sin perder sus características, es decir, manteniendo sus decisiones.

A través de técnicas de poda como el intercambio de variables o la eliminación de redundancia, se fueron generando un listado de posibles acciones que al ejecutarse sobre el árbol original daban lugar a un espacio de búsqueda, que según el algoritmo a utilizar se iría recorriendo de una forma u otra.

Los primeros algoritmos que se probaron, fueron los algoritmos de búsqueda no informada (anchura (BFS) y profundidad (DFS)). El mayor inconveniente era la necesidad de usar una copia en profundidad del árbol original y sobre esta copia ejecutar la acción correspondiente de la lista de acciones, para poder generar un nuevo estado. El resultado de esta operación son tiempos de ejecución más altos y mayor ocupación de memoria.

Para mejorar el proceso y evitar usar memoria se pensó en utilizar una heurística que fuera seleccionando en cada iteración el mejor de los estados (hijos) resultantes al aplicar las posibles acciones al estado padre, sin necesidad de hacer copias en profundidad del estado padre. El resultado mejoro bastante en tiempos de ejecución.

Todo este estudio de algoritmos de optimización esta pensado para ser aplicado al árbol que genera el análisis CEA determinista en función de los intervalos de λ.

Por este motivo a parte de las técnicas de intercambio de variables y eliminación de redundancias se añade una nueva acció donde poder ir agrupando nodos y ramas en función de λ.

De esta manera se consigue a través de un ID o DAN presentar un árbol optimizado del

análisis CEA determinista.

3.2 Trabajos futuros

En este trabajo se han analizado las técnicas descritas anteriormente para optimizar el árbol durante todo el proceso del algoritmo, pero habría que estudiar e investigar si otros posibles cambios se podrían aplicar para conseguir reducir aún más el árbol.

A nivel visual habría que:

- Realizar cambios en la función que utiliza la librería Graphviz, para que las ramas del árbol se muestren ordenadas por su intervalo λ.
- Evitar la coalescencia que genera OpenMarkov durante el proceso de optimización.

Anexo A

Pseudocódigo

Algoritmo 1 - applyAction		
	input:	action
1	if	$action = removeRedundancy$
2		$doAction()$
3	$else\ if$	$action = exchangeVariables$
4		$doAction()$
5	$else\ if$	$action = exchangeLambda$
6		$doAction()$

Algoritmo 2 - RemoveRedundancy		
	input:	$treeADDPotential \rightarrow Nodo$
		$probNet \rightarrow Datos\ red$
	output:	$treeADDPotential$
	Método: $isApplicable$	
1	if	$typeNode = CHANCE$
2	$for\ branch\ to\ branches$	
3		$CompareTrees\,(branches)$
4	$return\ isApplicable$	
	Método: $doAction$	
5	$branchPotential$	$Eliminar\ CHANCE \rightarrow Potential$
6	$for\ branch\ to\ branches$	
7	Eliminar una rama $\rightarrow$	$branchAfterRemoveRedundancy$
8	$Actualizar\ treeADDPotential$	

Algoritmo 3 - ExchangeVariables		
	input:	$treeADDPotential \rightarrow Nodo$ $probNet \rightarrow Datos\ red$
	output:	$treeADDPotential$
	Método: $isApplicable$	
1	if	$typeParent = CHANCE$ $and\ typeChild = DECISION$ $and\ isSameChildren$ $and\ isChildrenSameState$
2	$return\ isApplicable$	
	Método: $doAction$	
3	$for\ branch\ to\ branches$	
4	Nueva rama con el intercambio→	$Nodo\ root = DECISION$ $and\ Nodo\ child = CHANCE$
5	$Actualizar\ treeADDPotential$	

Algoritmo 4 - ExchangeLambda		
	input:	$strategyTree \rightarrow Arbol\ o\ rama$ $treeADDPotential \rightarrow Nodo$ $probNet \rightarrow Datos\ red$ $cep \rightarrow Particiones\ CEA$
	output:	$treeADDPotential$
	Método: $isApplicable$	
1	if	$typeChild = DECISION$ $and\ nodos\ iguales$
2	$else\ if$	$typeChild = CHANCE$ $and\ isCommonBranch$
3	$return\ isApplicable$	
	Método: $doAction$	
4	if	$typeChild = DECISION$
5	Nueva rama con el intercambio$\rightarrow$	$Nodo\ root = DECISION$ $and\ Nodo\ child = lambda$
6	$else\ if$	$typeChild = CHANCE$
7	Nueva rama$\rightarrow$	$Nodo\ root = CHANCE$ $Añadimos\ rama\ común\ al\ nodo$ $Creamos\ rama\ con\ nodo\ lambda$ $y\ con\ las\ ramas\ no\ comunes\ de$ $cada\ nodo\ CHANCE$
8	$Actualizar\ treeADDPotential$	

Bibliography

M. Arias. *Carmen: Una Herramienta de Software Libre para Modelos Gráficos Probabilistas*. PhD thesis, UNED, Madrid, Spain, 2009. In Spanish.

M. Arias and F. J. Díez. Cost-effectiveness analysis with sequential decisions. Technical Report CISIAD-11-01, UNED, Madrid, Spain, 2011.

M. Arias and F. J. Díez. The problem of embedded decision nodes in cost-effectiveness decision trees. *Pharmacoeconomics*, 32:1141–1145, 2014. doi: 10.1007/s40273-014-0195-1.

M. Arias and F. J. Díez. Cost-effectiveness analysis with influence diagrams. *Methods of Information in Medicine*, 54:353–358, 2015. doi: 10.3414/ME13-01-0121.

M. Arias, M. A. Artaso, I. Bermejo, F. J. Díez, M. Luque, and J. Pérez-Martín. Advanced algorithms for medical decision analysis. Implementation in OpenMarkov. In *Proceedings of the 16th Conference on Artificial Intelligence in Medicine (AIME 2017)*, Vienna, Austria, 2017a. doi: 10.1007/978-3-319-59758-4_43.

M. Arias, M. Luque, J. Pérez-Martín, and F. J. Díez. Cost-effectiveness analysis with decision analysis networks. In *Annual Meeting of the Society for Medical Decision Making*, Pittsburgh, PA, 2017b.

M. Arias, J. Pérez-Martín, M. Luque, and F. J. Díez. OpenMarkov, an open-source tool for probabilistic graphical models. In S. Kraus, editor, *Proceedings of the Twenty-eighth International Joint Conference on Artificial Intelligence (IJCAI'19)*, page 6485–6487, Macao, China, 2019. Morgan Kaufmann. doi: 10.24963/ijcai.2019/931.

B. G. Buchanan and E. A. Feigenbaum. DENDRAL and Meta-DENDRAL: Their applications dimension. *Artificial Intelligence*, 11:5–24, 1978.

F. J. Díez, M. Luque, and I. Bermejo. Decision analysis networks. *International Journal of Approximate Reasoning*, 96:1–17, 2018. doi: 10.1016/j.ijar.2018.02.007.

M. F. Drummond, M. J. Sculpher, K. Claxton, G. L. Stoddart, and G. W. Torrance. *Methods for the Economic Evaluation of Health Care Programmes*. Oxford University Press, Oxford, UK, 4th edition, 2015.

R. A. Howard and J. E. Matheson. Influence diagrams. In R. A. Howard and J. E. Matheson, editors, *Readings on the Principles and Applications of Decision Analysis*, page 719–762. Strategic Decisions Group, Menlo Park, CA, 1984.

M. Hunink, P. Glasziou, J. Siegel, J. Weeks, J. Pliskin, A. Elstein, and M. Weinstein. *Decision making in health and medicine*. Cambridge University Press, Cambridge, UK, 3rd edition, 2004.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, Cambridge, MA, 2009.

K. M. Kuntz and M. C. Weinstein. Modelling in economic evaluation. In M. F. Drummond and A. McGuire, editors, *Economic Evaluation in Health Care*, chapter 7, page 141–171. Oxford University Press, New York, 2001.

M. Luque. *Probabilistic Graphical Models for Decision Making in Medicine*. PhD thesis, UNED, Madrid, 2009.

M. Luque and F. J. Díez. Variable elimination for influence diagrams with super-value nodes. *International Journal of Approximate Reasoning*, 51:615–631, 2010. doi: 10.1016/j.ijar.2009.11.004.

M. Luque, F. J. Díez, and C. Disdier. Optimal sequence of tests for the mediastinal staging of non-small cell lung cancer. *BMC Medical Informatics and Decision Making*, 16:1–14, 2016. doi: 10.1186/s12911-016-0246-y.

M. Luque, M. Arias, and F. J. Díez. Synthesis of strategies in influence diagrams. In *Proceedings of the Thirty-third Conference on Uncertainty in Artificial Intelligence (UAI'17)*, page 1–9, Corvallis, OR, 2017. AUAI Press. URL `http://auai.org/uai2017/proceedings/papers/134.pdf`.

T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

S. Pauker and J. Wong. The influence of influence diagrams in medicine. *Decision Analysis*, 2:238–244, 2005.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

H. Raiffa. *Decision Analysis. Introductory Lectures on Choices under Uncertainty*. Addison-Wesley, Reading, MA, 1968.

H. Raiffa and R. Schlaifer. *Applied Statistical Decision Theory*. John Wiley & Sons, Cambridge, MA, 1961.