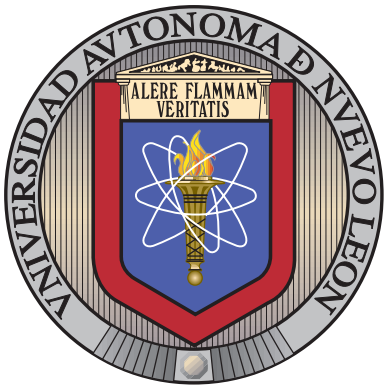
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO



DISEÑO Y EVALUACIÓN DE MODELOS DE PLANIFICACIÓN DE INTELIGENCIA ARTIFICIAL Y PROGRAMACIÓN MATEMÁTICA PARA GENERAR RUTAS DE APRENDIZAJE

POR

CRISTINA MAYA PADRÓN

EN OPCIÓN AL GRADO DE

DOCTOR EN INGENIERÍA

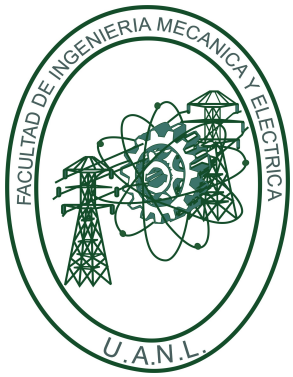
CON ESPECIALIDAD EN INGENIERÍA DE SISTEMAS

SAN NICOLÁS DE LOS GARZA, NUEVO LEÓN ENERO 2017

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO



DISEÑO Y EVALUACIÓN DE MODELOS DE PLANIFICACIÓN DE INTELIGENCIA ARTIFICIAL Y PROGRAMACIÓN MATEMÁTICA PARA GENERAR RUTAS DE APRENDIZAJE

POR

CRISTINA MAYA PADRÓN

EN OPCIÓN AL GRADO DE

DOCTOR EN INGENIERÍA

CON ESPECIALIDAD EN INGENIERÍA DE SISTEMAS

SAN NICOLÁS DE LOS GARZA, NUEVO LEÓN ENERO 2017

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la Tesis «Diseño y evaluación de modelos de planificación de Inteligencia Artificial y programación matemática para generar rutas de aprendizaje», realizada por el alumno Cristina Maya Padrón, con número de matrícula 0937602, sea aceptada para su defensa como opción al grado de Doctor en Ingeniería con especialidad en Ingeniería de Sistemas.

El Comité de Tesis

Dr. Romeo Sánchez Nigenda
Asesor

Dra. Iris Abril Martínez Salazar
Revisor

Dr. Francisco Torres Guerrero
Revisor

Dra. Lluvia Carolina Morales Reynaga
Revisor

Dra. Carolina Franco Espinosa
Revisor

Vo. Bo.

Dr. Simón Martínez Martínez
Subdirección de Estudios de Posgrado

San Nicolás de los Garza, Nuevo León, enero 2017

A mis cada vez más grandes y hermosas hijas:

Shannon Cristina y Karen Naomi.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

AGRADECIMIENTOS

Agradezco a la Facultad de Ingeniería Mecánica y Eléctrica de la Universidad Autónoma de Nuevo León, especialmente al Posgrado en Ingeniería de Sistemas.

Gracias al Consejo Nacional de Ciencia y Tecnología (CONACyT) por la beca otorgada para la realización de este trabajo de tesis.

RESUMEN

Cristina Maya Padrón.

Candidato para el grado de Doctor en Ingeniería
con especialidad en Ingeniería de Sistemas.

Universidad Autónoma de Nuevo León.
Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio: Diseño y evaluación de modelos de planificación de Inteligencia Artificial y programación matemática para generar rutas de aprendizaje

DISEÑO Y EVALUACIÓN DE MODELOS DE PLANIFICACIÓN DE INTELIGENCIA ARTIFICIAL Y PROGRAMACIÓN MATEMÁTICA PARA GENERAR RUTAS DE APRENDIZAJE

Número de páginas: 119.

OBJETIVOS Y MÉTODO DE ESTUDIO: En este proyecto de investigación se plantea y desarrolla un problema de generación de rutas de aprendizaje para estudiantes y se analizan los métodos de solución. Se realiza el *análisis de complejidad* de nuestro problema, demostrando que el *problema de planificación educativa* es NP-completo. Para generar las rutas de aprendizaje representamos el plan de estudios, así como la información del estudiante, en modelos de *planificación* de *inteligencia artificial* que utilizan para su solución, algoritmos de propósito especial llamados *planificadores* así como, la realización

de un modelo de programación matemática para conocer las soluciones óptimas de las instancias generadas. Los planificadores seleccionados para dar solución a los modelos de planificación son SGPLAN y LPG. El solver utilizado para resolver los modelos de programación matemática es GAMS utilizando CPLEX. Se realiza una metodología híbrida la cual combina planificación de IA y programación matemática para obtener mejores soluciones.

CONTRIBUCIONES Y CONCLUSIONES: Diseño de un modelo de planificación en PDDL y un modelo de programación matemática para representar el problema de planificación educativa (generación de rutas de aprendizaje). Realización de un análisis de complejidad de nuestro problema, demostrando que el *problema de planificación educativa* es NP-completo. Análisis de dos algoritmos de planificación (LPG y SGPLAN) para la resolución del problema de planificación educativa. Combinación de dos metodologías de solución: planificación de IA y programación matemática para la generación de rutas de aprendizaje. Como conclusión podemos decir que: la realización de comprobación de la complejidad de nuestro problema nos permite inferir en la importancia y relevancia de resolución del mismo. El diseño del modelo de planificación en PDDL representa correctamente el problema de generación de rutas de aprendizaje, sin embargo, los algoritmos de planificación presentan ciertas limitantes en la calidad de solución de los planes, presentando un GAP muy grande. Por su parte, el diseño del modelo de programación matemática nos permite conocer las soluciones óptimas de las instancias generadas en cuanto a la selección de actividades de aprendizaje que minimicen el tiempo total (función objetivo establecida en la experimentación), olvidando la ordenación de dichas actividades. Combinando ambas metodologías (planificación de IA y modelación matemática) nos permite obtener mejores soluciones.

Firma del asesor: _____

Dr. Romeo Sánchez Nigenda

CAPÍTULO 1

INTRODUCCIÓN

1.1 INTRODUCCIÓN

Las trayectorias de aprendizaje son lo que la IMS (Instructional Management System Proyect)[39] define como Diseño de aprendizaje (Learning Design) que son básicamente un plan de actividades educativas que se llevan a cabo con cierto orden con la finalidad de cumplir ciertos objetivos de aprendizaje.

En este proyecto de investigación se plantea un problema de generación de trayectorias de aprendizaje para estudiantes y se analizan los métodos de solución.

Se realiza el *análisis de complejidad* de nuestro problema, demostrando que el *problema de planificación educativa* es NP-completo.

Para generar las trayectorias aprendizaje representamos el plan de estudios, así como la información del estudiante, en modelos de *planificación* de *inteligencia artificial* que utilizan para su solución, algoritmos de propósito especial llamados *planificadores*.

En experimentaciones previas hemos observado que estos *algoritmos de planificación* tienen dificultad para calcular las trayectorias de aprendizaje (serie de actividades de aprendizaje o planes educativos) automatizadas para estudiantes [29]. Así que realizamos experimentaciones adicionales (vea la sección de experimentación) para encontrar aquellas propiedades que dificultan la resolución de los modelos de planificación. En esta experimentación se señalan tres propiedades importantes: el tamaño de las instancias,

la presencia de relaciones habilitantes (representan la precedencia) y la consideración de acumulación de métricas en los modelos de planificación.

Con la finalidad de conocer la calidad de las soluciones obtenidas por los algoritmos de planificación se desarrolló un modelo matemático. Dicho modelo no contempla el órden de las *actividades de aprendizaje*, sólo la selección de aquellas que minimizen el tiempo total (consideramos esta función objetivo para la experimentación). Además de lo anterior se omitió la estructura jerárquica, considerando que la acumulación de calidad sólo es en el nivel de los subtemas de una materia y se modela una sola materia (veremos a detalle en la sección de metodología). Los resultados del modelo matemático bajo estas condiciones fueron muy buenos.

Tomando en cuenta que este modelo matemático nos permite conocer las soluciones óptimas de las instancias, en cuanto a la selección de aquellas actividades de aprendizaje que minimizan el tiempo total que el estudiante puede dedicar a ellas, consideramos estas soluciones como entrada a los modelos de planificación. La intención es ayudar a los algoritmos de planificación para que obtengan mejores soluciones.

En las siguientes secciones se describe el problema, su relevancia y objetivos. En los siguientes capítulos se verá el estado del arte, marco teórico, el planteamiento de ambos modelos desarrollados (modelo de planificación de IA y modelo matemático), complejidad y la experimentación realizada.

1.2 DESCRIPCIÓN DEL PROBLEMA

Considerando que se tiene un programa académico, queremos generar una secuencia ordenada de actividades de aprendizaje para el estudiante que le permitan minimizar el tiempo total que puede dedicar a dichas actividades. Es decir, queremos generar planes de estudio que el estudiante pueda realizar de manera que lo guíen durante su trayectoria escolar teniendo una medida de desempeño mínima-máxima y considerando optimizar el tiempo total. A este problema le llamamos *Problema de Planificación Educativa*(PPE)

El objetivo general de nuestros modelos es representar el plan de estudios y los objetivos de aprendizaje que los estudiantes deben cumplir tal que los algoritmos de planificación de dominio independiente puedan generar planes educativos personalizados que, en la cantidad mínima de tiempo, permita a los estudiantes alcanzar con éxito los objetivos de aprendizaje curricular. El diseño de nuestros modelos nos permite incluir en el objetivo las preferencias del usuario. Por ejemplo, podemos requerir calcular un plan que alcanza la nota mínima aprobatoria en la cantidad mínima de tiempo, donde el nivel de logro es definida por el usuario. Tal grado de libertad nos permite codificar las preferencias del usuario en términos de requisitos de objetivos, por tanto la personalización de las trayectorias de aprendizaje.

1.3 RELEVANCIA DEL PROBLEMA

Investigaciones recientes reportan la importancia de personalizar el entorno de aprendizaje de los estudiantes[13], de administrar su propio estudio [38] y de utilizar las aplicaciones tecnológicas sosteniendo que el alumno tiende aprender mejor cuando éstas están directamente relacionadas al curriculum que tienen que cubrir [45].

Se ha observado que los problemas de aprendizaje no son situaciones que surgen de forma inesperada ([35]), por lo que creemos que es importante analizar las trayectorias de aprendizaje de los estudiantes para implementar acciones preventivas y promover hábitos de estudio. De hecho, el Instituto de Ciencias de la Educación (Institute of Education Sciences (IES)) de Estados Unidos recomienda personalizar el entorno académico y el proceso de instrucción de los estudiantes ([13]).

La tecnología podría desempeñar un papel crucial para lograr tales recomendaciones. Tecnologías que otorgan a los estudiantes un mayor acceso al plan de estudios, que les permiten establecer sus propios objetivos de aprendizaje teniendo en cuenta sus preferencias de aprendizaje y las limitaciones de tiempo, y que consideran los recursos del sistema escolar ([45]) podría facilitar el proceso de aprendizaje a través del tiempo.

Para promover el progreso de tal tecnología, nuestro trabajo presenta un formalismo de Planificación de Inteligencia Artificial (IA) flexible que define los conceptos necesarios para construir y organizar las trayectorias de aprendizaje personalizadas (es decir, los planes de aprendizaje). La Planificación de IA es, en general, el proceso de sintetizar un curso de acciones que, si se ejecutan desde un estado inicial específico, se obtendrán un conjunto de objetivos ([42, 21]. Tenga en cuenta que el diseño de trayectorias de aprendizaje podría ser visto como un problema de planificación de IA, donde los objetivos de aprendizaje se correlacionan con los objetivos de planificación y las actividades de aprendizaje a las acciones. El problema es entonces generar planes (trayectorias de aprendizaje) que permitan alcanzar los objetivos de aprendizaje que los estudiantes deben completar en el cumplimiento de las restricciones del sistema y las preferencias de los usuarios.

Codificamos nuestros modelos utilizando un formalismo estándar de representación del conocimiento de IA, el *lenguaje de definición de dominios de Planificación* (PDDL - [15]), esto significa que podemos aprovechar los algoritmos de planificación disponibles y el apoyo fuerte y activo de la comunidad de investigación en el área. Obviamente, necesitamos herramientas que pueden facilitar la generación de nuestros modelos para el usuario final. Afortunadamente, podemos aprovechar el trabajo sobre la validación y verificación de los dominios de planificación (Orlandini et al. 2013), e ingeniería del conocimiento de la planificación (Vaquero et al. 2013) para apoyar tal tarea.

En este trabajo se presentan los conceptos teóricos que apoyan nuestros modelos de planificación. Desarrollamos un ejemplo de cómo es que nuestro formalismo genera las trayectorias de aprendizaje. Realizamos un estudio para demostrar la complejidad de nuestro problema. Desarrollamos un modelo de programación matemática para evaluar la optimalidad de las trayectorias de aprendizaje generadas por los modelos de planificación. Analizamos las propiedades de nuestros modelos para identificar aquellas que aumentan la complejidad de solución. Llevamos a cabo un análisis del comportamiento de los algoritmos de planificación para dar solución a los modelos. Y con el objeto de mejorar las solciones dadas por los algoritmos de planificación, utilizamos las soluciones del modelo matemático en el modelo de planificación.

1.4 OBJETIVOS

El objetivo de esta investigación es el estudio de un problema de *generación de trayectorias de aprendizaje* para estudiantes así como sus métodos de solución. Desarrollando dos modelos: un modelo de planificación de inteligencia artificial y un modelo matemático. Presentando un análisis de dos algoritmos de planificación que se utilizan para dar solución a los modelos de planificación. Así como una metodología híbrida que utiliza las soluciones del modelo matématico para incluirlas en el modelo de planificación con la finalidad de obtener mejores soluciones.

1.4.1 OBJETIVOS ESPECÍFICOS

- Desarrollo de un *modelo de planificación* de Inteligencia Artificial en el lenguaje PDDL que representa el *problema de planificación educativa* con la finalidad de obtener las trayectorias de aprendizaje.
- Realización de un estudio para demostrar la complejidad de nuestro problema.
- Experimentación del comportamiento de dos algoritmos de planificación seleccionados para dar solución al modelo de planificación.
- Análisis de las propiedades de nuestros modelos que aumentan la complejidad de solución.

  Analizamos las propiedades de nuestros modelos para identificar aquellas que aumentan la complejidad de solución.
- Desarrollo de un *modelo matemático* de programación lineal entera mixta para la selección de actividades de aprendizaje que minimicen el tiempo total. Se utiliza para conocer la calidad de las soluciones de los modelos de planificación en cuanto a la selección de aquellas actividades que cumplan con la función objetivo (minimizar el tiempo total).

- Realizar una metodología híbrida en la cual se considere la utilización de la programación matemática y la planificación de inteligencia artificial. Dicha metodología explotaría las bondades de ambas ramas de la investigación. En donde se puede utilizar la programación matemática para la selección de actividades y la planificación para la secuenciación. Esto con la finalidad de mejorar las soluciones.

CAPÍTULO 2

ESTADO DEL ARTE

2.1 TRABAJO RELACIONADO

La generación de trayectorias de aprendizaje (es decir, secuencias de actividades de aprendizaje) a través de materiales de aprendizaje se ha estudiado de manera diferente por varios campos de investigación y aplicación, tales como los Sistemas Tutores Inteligentes (ITS - Intelligent Tutoring Systems) ([47],[48], [49]), Sistemas Adaptativos Educativos Hipermedia (AEHS-Adaptive Educational Hypermedia Systems) ([25], [4]), Inteligencia Artificial (IA) ([12],[14], [6, 7], [19, 17], [32]), Sistemas Gestores de Aprendizaje (LMS) ([26, 27], [5]), y los Entornos Virtuales de Aprendizaje (VLE-Virtual Learning Environments) ([33]), entre otros. Aunque estas áreas de aplicación son diferentes, tienen un objetivo en común: personalizar la generación de trayectorias de aprendizaje para estudiantes. Podemos utilizar las características homogéneas de estas áreas para diseñar modelos que generen rutas de aprendizaje o secuencias cada vez mejores y más personalizadas.

Para poner nuestro trabajo en contexto, no pretendemos vincular nuestro trabajo a un ambiente de aprendizaje específico, como las áreas mencionadas anteriormente. De hecho, presentamos un formalismo de un modelo flexible que podría soportar la generación de trayectorias de aprendizaje en diferentes áreas de aplicación. Utilizamos PDDL como un medio para modelar el conocimiento pedagógico necesario para generar planes de aprendizaje. Aunque *planificación de IA* ha sido útil para modelar problemas complejos a partir de un estado inicial, una representación objetivo, y un conjunto de transiciones de acción (([42, 21]); es sólo recientemente que el problema de generación de trayectorias de apren-

dizaje es visto como un problema de *planificación* [12]. La principal motivación detrás de la utilización de modelos de planificación para representar trayectorias de aprendizaje radica en que permite la personalización y optimización del proceso de aprendizaje.

Los trabajos relacionados encontrados que utilizan la planificación para generar trayectorias de aprendizaje están estrechamente vinculados a la realización de modelos de planificación a través de la recopilación de la información contenida en los Sistemas Gestores de Aprendizaje (LMS), los cuales hacen uso de estándares de E-learning para representar la información del curso ([1]. Estos modelos al igual que nuestro trabajo consideran propiedades que deben existir para generar correctamente trayectorias de aprendizaje, tales como el tiempo ([31, 32], [7, 6]), métricas de optimización y restricciones en los recursos del sistema ([19, 17, 18], utilidades en las actividades de aprendizaje (score) ([14]), y jerarquía de tareas ([12]). Por otra parte, muchas de las propiedades de nuestros modelos también son consideradas por autores de otras áreas de aplicación, por ejemplo: estructuras jerárquicas para el aprendizaje ([47, 46]), composición de cursos adaptados ([49, 48, 47]), relaciones entre los conceptos de aprendizaje ([49, 5]), y la posibilidad de utilizar diferentes algoritmos para la generación de planes ([46]).

Asimismo, las áreas de investigación fuera de Planificación de IA han hecho grandes avances en las implicaciones de generar trayectorias de aprendizaje. Por ejemplo, en la replanificación dinámica de cursos de acuerdo al progreso del estudiante ([49, 48], [46], [33]), la generación de planes parciales con diferentes estados iniciales para estudiantes ([46], [5]), y la total integración de las técnicas de planificación a LMS como Moodle ([5]). Tenga en cuenta que nuestro formalismo también podría utilizarse para hacer frente a esas implicaciones. Podemos intercalar la planificación y ejecución de nuestros modelos para adaptarse al progreso del estudiante. Además, mediante el uso de PDDL, podemos construir fácilmente modelos con diferentes estados iniciales de los estudiantes representando posibles situaciones académicas.

Por otro lado, nuestro enfoque encapsula un valor de utilidad/recompensa a través del uso de las Funciones de Acumulación de Calidad (QAFs -[11]). Hemos diseñado tres clases diferentes de QAFs en nuestros modelos para representar la satisfacción del obje-

tivo total y la satisfacción métrica parcial de conceptos de aprendizaje. Estas QAFs nos permiten asignar un valor métrico a cada componente del modelo, que luego se pueden utilizar para especificar las restricciones del usuario, preferencias, funciones de acumulación múltiples y relaciones de órden entre los componentes del modelo para producir trayectorias de aprendizaje más personalizadas. Sin embargo, estamos de acuerdo en que algunos conceptos pedagógicos podrían ser difíciles de cuantificar numéricamente.

Con respecto a la modelación matemática, los trabajos relacionados a entornos educativos han orientado sus esfuerzos hacia la estimación de la gestión de recursos en las instituciones educativas ([44, 34, 3]), la asignación de profesores a los cursos en las universidades ([**?**, 28]), la asignación de personal a las actividades académicas diarias ([36]), así como salas y los horarios a eventos de enseñanza (clases/ cursos que se ofrecen, tutoriales, seminarios ([2]). Algunos estudios consideran las restricciones presupuestales para la planificación y la toma de decisiones ([50, 44]), y la vinculación de las reglas específicas a las trayectorias de aprendizaje en primaria en el área de matemáticas ([9]).

El trabajo más cercano al modelo matemático que nosotros desarrollamos es presentado en [14]. Ellos dirigen el diseño de cada curso como una variación del problema de sobresuscripción y utilizan la programación lineal para asistir a un planificador en la generación del diseño que se adapte mejor a los estudiantes. Nuestro enfoque en cambio, considera el conjunto total de los datos de aprendizaje en el modelo de planificación para generar los planes, pero utiliza las QAFs para decidir qué objetivos de aprendizaje vale la pena probar. En este sentido, nuestro problema también puede verse como un problema de sobresuscripción, ya que no es necesario llevar a cabo todas las actividades de aprendizaje en el modelo para satisfacer requerimientos de la solución.

2.2 ÁREAS DE OPORTUNIDAD

Los trabajos encontrados a la fecha para generar trayectorias de aprendizaje consideran la personalización de un solo curso/materia para el estudiante, dejando de lado que un estudiante quisiera obtener la planeación de todos los cursos o materias que cursa. Por

esta razón, nosotros consideramos la modelación de varias materias. Aunado a lo anterior queremos analizar el desempeño de los algoritmos de planificación para dar solución a estos modelos cuando tienen que resolver.

Hasta donde se tiene conocimiento, el estudio de trayectorias de aprendizaje para estudiantes tomando en cuenta la optimización del tiempo total (métrica a optimizar en la experimentación, sin embargo hay otras que pueden considerarse), la acumulación de utilidad mínima-máxima en la unidad de aprendizaje, relaciones de precedencia entre las actividades (lo que llamamos relaciones habilitantes), así como la utilización de un método de solución efectivo para dicho problema que se propone en este trabajo, no se ha abordado como tal en la literatura especializada. Por lo tanto, el desarrollo de este trabajo presenta una contribución relevante para el estado del arte en el campo de aplicación de la Planificación de Inteligencia Artificial, así como en el de Investigación de Operaciones (programación matemática). Es decir, el trabajo como tal, no ha sido abordado en la literatura, así como la metodología propuesta para dar solución.

CAPÍTULO 3

MARCO TEÓRICO

_

3.1 PLANIFICACIÓN DE INTELIGENCIA ARTIFICIAL

Partiendo de la necesidad de generar trayectorias de aprendizaje personalizadas que nos permitan cubrir un conjunto de objetivos de aprendizaje emplearemos técnicas de ***Planificación de Inteligencia Artificial***.

Russell y Norvig en [42] definen la **Inteligencia Artificial (IA)** como el "estudio de la acción racional", lo que significa que la **Planificación** - la elaboración de un plan de acción para alcanzar un objetivo - es una parte crítica y racional para la consecución de un conjunto de objetivos.

La **Planificación en Inteligencia Artificial (IA)** es la disciplina que centra su proceso de solución de problemas en *"la búsqueda y articulación de una secuencia de acciones que permiten alcanzar un objetivo"* [41]; es decir, permite al agente utilizar creencias sobre acciones y sus consecuencias para buscar una *solución* dentro del más abstracto espacio de los planes, no en el de situaciones [40]. Así, los *algoritmos de planificación* se pueden considerar como demostradores de teoremas de propósito especial capaces de razonar de manera eficiente con los axiomas que describen las acciones.

La **planificación de IA clásica** tiene en cuenta entornos que son completamente observables, deterministas, finitos, estáticos (los cambios suceden sólo cuando los agentes

actúan) y discretos (en tiempo, acciones, objetos y efectos).

3.1.1 SOLUCIÓN

Al resultado del proceso de Planificación se llama "solución". El término *solución* según [40] es un *plan* que un agente puede ejecutar y garantiza el logro de la meta. Es decir, una secuencia de acciones que, ejecutada en el estado inicial, da como resultado un estado final que satisface el objetivo.

Decimos que una **solución** es un plan *completo* y *consistente*. Un *plan completo* es aquel en el que cada condición previa de todas las acciones se logra mediante otra acción. Una acción logra una condición si ésta es uno de los efectos producidos por otra acción, y si ningún otra acción tiene posibilidad de eliminar la condición. Un *plan consistente* es aquel en el que no hay contradicciones en las restricciones de ordenamiento o enlazamiento.

3.2 LENGUAJE DE DEFINICIÓN DE DOMINIOS DE PLANIFICACIÓN (PDDL)

La *Planificación en Inteligencia Artificial* utiliza un lenguaje de modelación el cuál es llamado **PDDL** (acrónimo de *Planning Domain Definition Language*), el cual es el lenguaje estándar para la codificación de los dominios de planificación dado a conocer por Drew McDermott en la **International Planning Competition (IPC)** de 1998 [30] (IPC se describe en una sección posterior).

PDDL es un lenguaje centrado en las *acciones* inspirado en las formulaciones "Strips" de problemas de planificación. Éste es una estandarización de la sintaxis para expresar acciones utilizando precondiciones y post-condiciones para describir la aplicabilidad y efectos de las acciones [15]. La sintaxis está inspirada en Lisp [51] (acrónimo de LISt Processing), así que gran parte de la estructura de la descripción del dominio es una lista como Lisp de las expresiones entre paréntesis.

3.2.1 ESTRUCTURA DE LOS MODELOS DE PLANIFICACIÓN EN PDDL

Un *modelo de planificación en PDDL* está organizado en dos partes principales: el *dominio de planificación* y el *problema de planificación*. Siendo el *dominio* la construcción del modelo como tal y el *problema* una instancia a resolver de ese *dominio*. Sin embargo, es común utilizar el término *dominio de planificación* para referirse en sí al *modelo de planificación*.

A continuación se describe cómo está organizado el modelo de planificación en PDDL:

1. ***Dominio de planificación*.**- El cual describe las reglas de actuación y está compuesto por:
   - **Predicados**: Estos representan relaciones entre los objetos. Nos ayudan a describir un problema del mundo real tratando de representar los conceptos de nuestro problema a través de relaciones entre los objetos que lo conforman.
   - **Fluents**: Funciones que nos permiten manejar valores numéricos.
   - **Acciones / Operadores**: Maneras de cambiar el estado del mundo.
2. ***Problema de planificación***.- Describe el estado del mundo circundante y las metas. Está compuesto por:
   - **Objetos**: Las cosas en el mundo que nos interesan.
   - **Estado inicial**: El estado del mundo en el que se empieza. Es decir, el punto de partida de la búsqueda.
   - **Especificación Objetivo**: Comprueba si el estado actual corresponde a una solución del problema.

ELEMENTOS DE LAS ACCIONES

Las **acciones** forman parte del *dominio de planificación en PDDL* y son las maneras de cambiar el estado del mundo. A continuación se mencionan sus componentes:

- **Especificación de la acción**: es lo que de hecho devuelve el agente al ambiente para así proceder a hacer algo. Cuando está dentro del planificador sirve como nombre de la acción.

- **La condición previa**: es una conjunción de átomos (literales positivas) o funciones (fluents) que dice *qué debe existir* (ser verdad) antes de poder aplicar el operador.

- **El efecto de un operador**: es una conjunción de literales (positivas o negativas) o funciones (fluents) que dice *de qué manera cambia la situación* al aplicar el operador.

Es decir, todo lo que hay en la condición previa implícitamente se refiere a la situación inmediatamente previa a la acción así como todo lo que hay en el efecto implícitamente se refiere a la situación resultante de la acción.

Las ***acciones*** pueden ser de dos tipos: acciones que tienen *duración*, esto es, que involucra la consideración del tiempo en la ejecución de la misma y las que no. A las acciones que tienen en cuenta la duración de la misma son llamadas ***acciones durativas***. Estas *acciones durativas* además de tener los componentes clásicos de las acciones, descrito anteriormente, incluyen el término de *duración de la acción*. Nota: En las acciones durativas la *condición previa* es llamada *precondición*.

3.2.2 EJEMPLO DEL LENGUAJE PDDL

A continuación se muestra el ejemplo típico de **planificación**, el **mundo de los bloques** [41]. Este dominio consiste en un conjunto de bloques con forma de cubo situados en una mesa. Los bloques pueden ser amontonados, pero únicamente podemos situar uno sobre otro. Un brazo mecánico puede cambiar bloques de sitio, tanto sobre la mesa como sobre otros bloques. El brazo puede tomar sólo un bloque por cada instante de tiempo, de modo que no puede tomar uno si aún no ha soltado otro anterior. El objetivo será construir una o más configuraciones de bloques, que quedarán especificados en términos de cuántos bloques están sobre cuántos otros bloques.

En el ejemplo del mundo de los bloques supóngase que el *estado inicial* es como sigue: tenemos cuatro objetos que son los bloques A, B, C, D y todos los bloques están sobre la mesa, figura3.1.

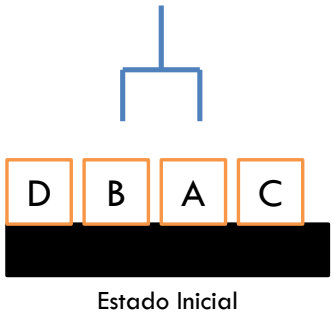


Figura 3.1: Ejemplo del mundo de los bloques- Estado inicial

El objetivo es que los bloques estén apilados de la siguiente manera: el bloque A es el único que está sobre la mesa, luego el bloque B está apilado sobre el bloque A, el C sobre el B, y el D sobre C respectivamente, figura 3.2, esto es lo que conocemos como el *estado final*.

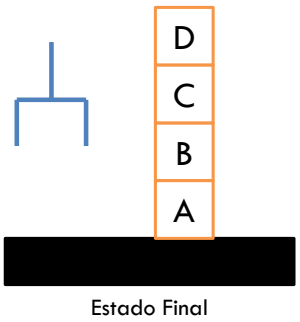


Figura 3.2: Ejemplo del mundo de los bloques- Estado final

DOMINIO

Empezamos definiendo el ***dominio***. Éste consta de un nombre, en la figura 3.3 se muestra la sintaxis. *Los requerimientos* en este ejemplo sencillo es STRIPS, figura 3.4. Los predicados, figura 3.5, representan los conceptos de nuestro problema a través de relaciones entre los objetos que lo conforman, por ejemplo, en el caso del predicado *on(a,b)*

estoy estableciendo una relación entre *a* y *b*, en este caso es que el bloque *a* se encuentra encima del bloque *b*.

En el ejemplo tenemos 5 predicados:

(on ?x ?y) hace referencia a que el objeto *x* se encuentra sobre el objeto *y*,

(ontable ?x) indica que el objeto *x* está sobre la mesa,

(clear ?x) indica que el objeto *x* esta "limpio", es decir que no tiene ningún objeto apilado,

(handempty) indica que el brazo mecánico está disponible,

(holding ?x) contrario al predicado anterior, indica que el brazo mecánico tiene sostenido un objeto.

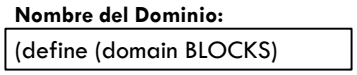


Figura 3.3: Ejemplo del mundo de los bloques- Nombre de Dominio

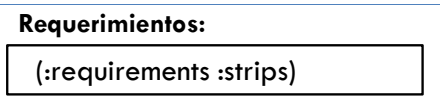


Figura 3.4: Ejemplo del mundo de los bloques- Requerimientos del Dominio

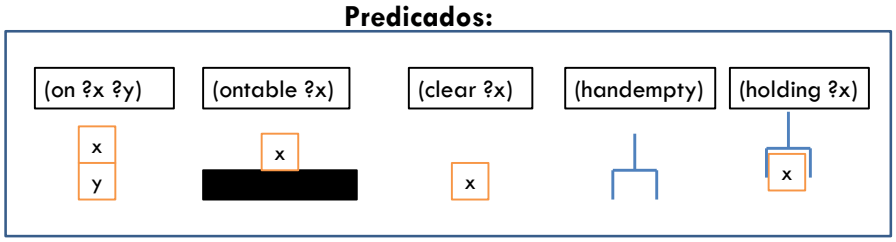


Figura 3.5: Ejemplo del mundo de los bloques- Predicados del Dominio

Una vez definido lo anterior, se procede a modelar las *acciones* posibles a ejecutar. En el ejemplo tenemos cuatro acciones que son: "pick up", "put down", "stack" y "unstack" que describimos a continuación.

Acción: "pick-up". Esta acción hace referencia a "levantar un bloque de la mesa". En la figura 3.6 se observa cómo se representa esta acción, tanto en sintaxis como en imagen para ilustrar. Nótese que la acción debe tener un nombre. Los componentes de esta acción son:

- *parámetros*: aquí solo necesita un bloque.
- *precondición*: es para indicar los requisitos que se deben cubrir para poder aplicar esta acción, los cuales se definen con los *predicados*: "clear", que dice que el objeto no tiene nada encima, "ontable" que el objeto está sobre la mesa y "handempty" que indica que el brazo mecánico esta libre.
- *efectos*: son las consecuencias de aplicar esta acción, y al igual que en las precondiciones se utilizan los predicados que definimos anteriormente, así como sus negaciones. En este ejemplo sencillo los predicados utilizados son "holding" que indica que el objeto esta siendo tomado, y las negaciones de los predicados "clear", "ontable", "handempty".

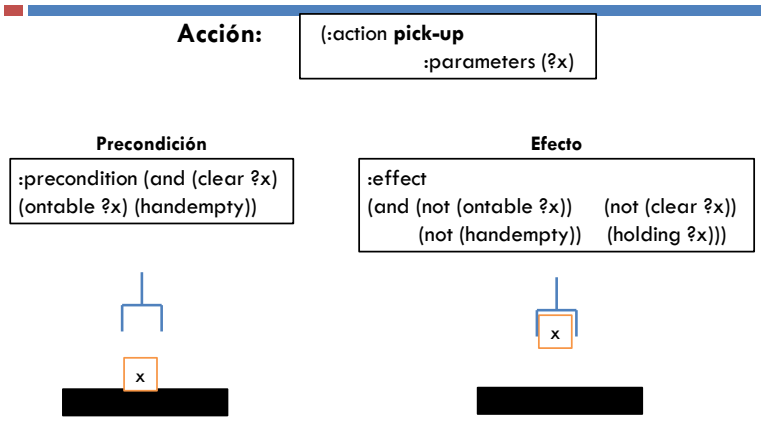


Figura 3.6: Ejemplo del mundo de los bloques- Acción pick up y su componentes.

Acción: "put-down". Esta acción hace referencia a "dejar un bloque sobre la mesa", en la figura 3.7 se observa cómo se representa esta acción, tanto en sintaxis como en imagen para ilustrar. Los componentes de esta acción son:

- *parámetros*: aquí solo necesita un bloque.

- *precondición*: es estar sosteniendo el bloque *x*, los predicados para indicar esto son: "holding".

- *efectos*: como la precondición era estar sosteniendo el bloque *x*, en efectos se hace la negación de que el bloque está sostenido "holding", y ahora decimos que el brazo mecánico esta disponible con el predicado "handempty", decimos además que el bloque *x* está "clear" y "ontable" .

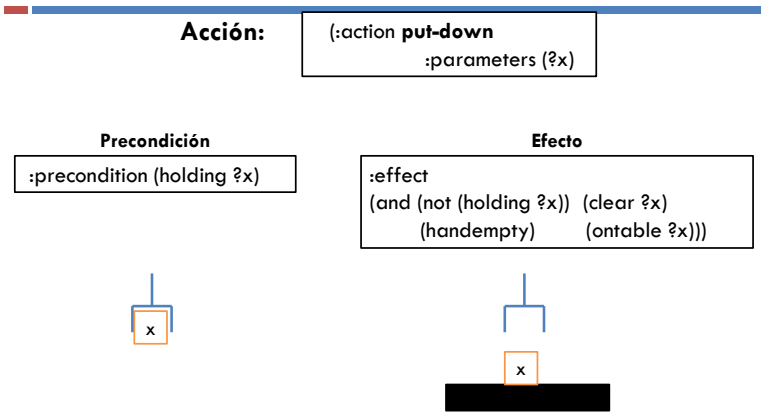


Figura 3.7: Ejemplo del mundo de los bloques- Acción Put-down y su componentes

Acción: "stack" que hace referencia a "apilar un bloque", en la figura 3.8 se observa cómo se representa esta acción, tanto en sintaxis como en imagen para ilustrar. Los componentes de esta acción son:

- *parámetros*: aquí necesita dos bloques, el que será apilado y sobre el cual se apilará.

- *precondición*: que el bloque *y* sobre e cual se quiere apilar, esté sin nada encima, es decir, "clear", y el bloque *x* esté siendo sostenido por el brazo mecánico "holding".

- *efectos*: la negación de toda la precondición, es decir, que el bloque *x* ya no esta sostenido la negación de "holding", y ya que éste se apiló en el bloque *y*, *y* ya no está "clear". Ahora el bloque que está "clear" es el apilado, el bloque *x*, también están los predicados del brazo vacío "handempty" y el predicado "on" que indica que un bloque esta en otro.

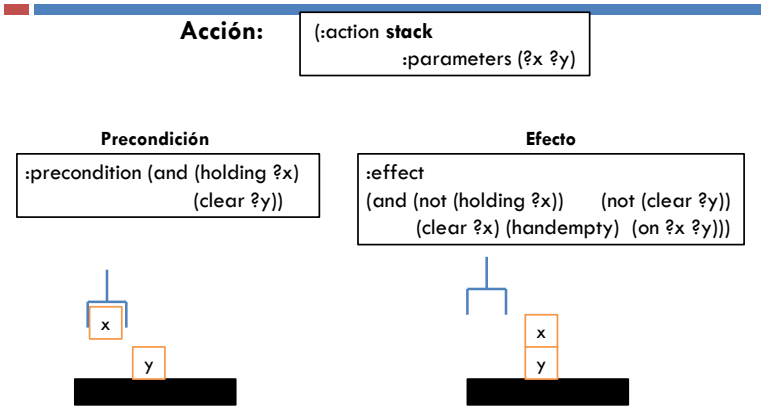


Figura 3.8: Ejemplo del mundo de los bloques- Acción stack y su componentes

Acción: "unstack" esta es la acción contraria a "stack". Se intenta quitar un bloque que esta apilado en otro, en la figura 3.9 se observa cómo se representa esta acción, tanto en sintaxis como en imagen para ilustrar. Los componentes de esta acción son:

- *parámetros*: aquí necesita dos bloques: "x" y "y".
- *precondición*: como la acción es "desapilar" un bloque de otro, necesitamos tener algo "apilado", así que la precondición es que el bloque "x" esté en "y" (on), que el bloque "x" no tenga nada encima (clear) y que el brazo mecánico esté disponible (handempty).
- *efectos*: el brazo mecánico sostiene al bloque "x", "y" no tiene nada encima, se agregan la negación de que el bloque "x" no tiene nada encima (clear), la negación de que el brazo esté disponible (handempty) y la negación de que esté el bloque "x" en "y".

PROBLEMA

Una vez que tenemos el *dominio*, procedemos a definir el *problema*, cabe aclarar que una vez que tenemos el dominio definido, podemos modelar varios problemas, es decir, diferentes instancias o escenarios que utilizan el mismo conjunto de operadores para solucionarlos. En el ejemplo el problema tiene cuatro objetos, y el objetivo es que

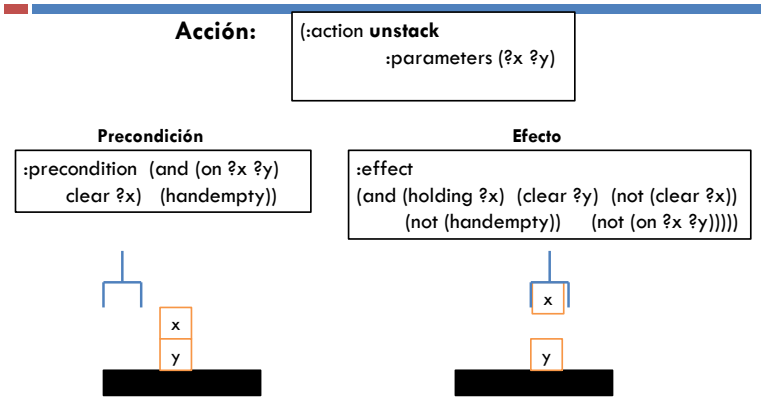


Figura 3.9: Ejemplo del mundo de los bloques- Acción unstack y su componentes

estén apilados uno sobre el otro siguiendo el orden A, B, C, D. Esto sería un escenario, pero podríamos tener más ó menos bloques, pudiera ser que en lugar de estar todos los bloques apilados uno encima del otro, solo se quieran dos, o en un orden distinto, etc.

Al igual que en el *dominio*, el *problema* tiene un "nombre" como se ve en la figura 3.10, es importante definir además, a que dominio pertenece, figura 3.11 y los objetos a modelar, en el ejemplo son cuatro, véase figura 3.12.

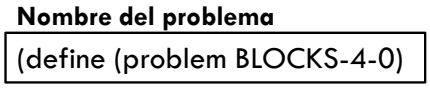


Figura 3.10: Ejemplo del mundo de los bloques- Nombre de Problema

Figura 3.11: Ejemplo del mundo de los bloques- A que dominio pertenece el Problema

Es en el problema en donde definimos el*estado inicial* y el *estado final* (objetivo). En el ejemplo de los bloques tenemos que el estado inicial es que los cuatro bloques están sobre la mesa (ontable), estos mismos bloques no tienen nada apilado (clear) y el brazo mecánico esta disponible. La figura 3.13 muestra como se define este estado inicial en pddl.

Asimismo el estado final que queremos modelar es que los bloques estén apilados

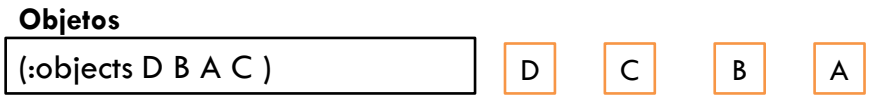


Figura 3.12: Ejemplo del mundo de los bloques- Objetos del problema

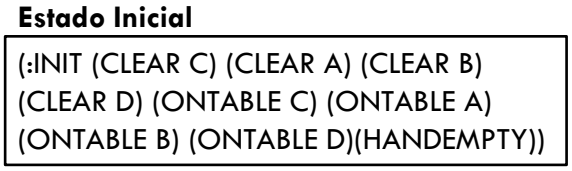


Figura 3.13: Ejemplo del mundo de los bloques- Estado Inicial definido en pddl

en el siguiente orden, D sobre C, C sobre B, B sobre A. Esto lo mostramos en la figura 3.14

SOLUCIÓN

Una vez que tenemos el modelo de planificación (dominio y problema) del mundo de los bloques, se procede a obtener soluciones del mismo. Esto se logra utilizando cualquier *planificador* que soporte STRIPS.

Una solución para este problema sería el plan mostrado en la tabla 3.1.

3.2.3 PDDL 2.1

Para **PDDL** se han desarrollado varias versiones en donde se han ido agregando mejoras. Después de la versión original por McDermott, Maria Fox and Derek Long extendieron la versión para permitir variables numéricas y la ejecución concurrente de acciones durativas, la cual es **PDDL 2.1** [15]. Esta versión se utilizó como el lenguaje de la competencia IPC del 2002.

Se mencionan las siguientes garantías de compatibilidad con versiones anteriores:

Estado Final

```
(:goal (AND (ON D C) (ON C B)
(ON B A)))
)
```

Figura 3.14: Ejemplo del mundo de los bloques- Estado Final definido en pddl

Paso 0:	Pick-up D
Paso 1:	Stack D C
Paso 2:	Pick-up B
Paso 3:	Stack B A
Paso 4:	Unstack D C
Paso 5:	Put-down D
Paso 6:	Pick-up C
Paso 7:	Stack C B
Paso 8:	Pick-up D
Paso 9:	Stack D C

Tabla 3.1: Plan del modelo del "mundo de los bloques"

- Todos los dominios existentes PDDL (de uso común) son válidos en dominios PDDL 2.1. Esto es importante para permitir que bibliotecas existentes de problemas de referencia sigan siendo válidas.
- Los planes válidos en PDDL son válidos en planes PDDL 2.1.

Además de la compatibilidad con las versiones anteriores, PDDL 2.1 tiene tres características importantes en relación con las versión original. Las características principales de PDDL 2.1 son:

- Tratamiento de (un conjunto finito de) *fluents* con valores numéricos.

- Representación explícita de tiempo y duración.
- Especificación de la *métrica del plan* como parte de la instancia del problema de planificación.

CARACTERÍSTICAS DE PDDL 2.1

A continuación detallamos y desglozamos las características de PDDL 2.1.

Numéricas En PDDL 2.1 se propuso una sintaxis definitiva para las expresiones de *fluentes numéricos* (fluents). Las expresiones numéricas se construyen utilizando operadores aritméticos de expresiones numéricas primitivas, los cuales son valores asociados con la tupla de los objetos de dominio por las funciones de dominio. Cabe mencionar que la sintaxis que se utiliza es prefija para todos los operadores aritméticos, incluyendo los predicados de comparación, con el fin de simplificar el análisis.

Condiciones Las condiciones en las expresiones numéricas son siempre las comparaciones entre pares de expresiones numéricas. Los efectos pueden hacer uso de una selección de las operaciones de asignación con el fin de actualizar los valores de las expresiones numéricas primitivas. Estos incluyen la asignación directa y asignaciones relativas (tales como aumento y disminución). Los números no se distinguen en sus posibles funciones, por lo que los valores pueden representar, por ejemplo, las cantidades de recursos, acumulación de servicios públicos, índices o contadores. En esta versión se decidió sólo permitir funciones con valores numéricos.

Métricas del Plan Las métricas del plan especificarán la base sobre la cual se evaluará un plan para un problema particular. Los mismos estados iniciales y finales pueden producir planes óptimos totalmente diferentes dadas diferentes métricas del plan. Por supuesto, un planificador podría no optar por utilizar la métrica para orientar su desarrollo de una solución, sólo para evaluar una solución post hoc. Este enfoque podría dar lugar a planes sub-óptimos, e incluso de mala calidad, pero es un enfoque pragmático para el manejo de métricas, las cuales han sido ampliamente utilizadas en las competiciones.

Una de las métricas más utilizadas es el valor total de tiempo (total-time) que puede ser usado para referirse a la duración temporal de todo el plan. Otros valores deben ser construido a partir de expresiones numéricas primitivas definidas dentro de un dominio y manipulados por las acciones del dominio. Como consecuencia de ello, los indicadores del plan sólo puede expresar métricas no-temporales en PDDL2.1 utilizando expresiones numéricas en el dominio.

Cualquier expresión aritmética puede ser utilizada en la especificación de una métrica, no hay ningún requisito de que la expresión sea lineal. Es responsabilidad del diseñador de dominio asegurarse de que la métrica del plan esté bien definida (por ejemplo, no realizar una división por cero). Se muestra a continuación un ejemplo de la sintaxis de una métrica: el ejemplo de gasto de gasolina. *(: metric minimize (+ (* 2 (fuel-used car)) (fuel-used truck)))*

.

Acciones durativas Para manejar los trabajos recientes de planificación temporal se han desarrollado dos formas de acción durativa que permiten la especificación de formas restringidas de condiciones de tiempo y efectos en su descripción. Las dos formas son acciones durativas discretas y acciones durativas continuas.

El modelado de las relaciones temporales en una acción durativa discreta se realiza por medio de condiciones y efectos anotados temporalmente. Todas las condiciones y los efectos de las acciones durativas debe ser anotadas temporalmente. La anotación de una condición se hace explícita si la proposición asociada se debe mantener al *inicio* del intervalo (el punto en el que se aplica la acción), el *final* del intervalo (el punto en el que los efectos finales de la acción se afirman) o en el intervalo del *inicio* y el *fin* (invariable durante la duración de la acción). La anotación de un efecto se hace explícito si el efecto es inmediato (lo que sucede en el inicio del intervalo) o tardío (si sucede al final del intervalo). Ningún otro punto en el tiempo es accesible, por lo que toda la actividad discreta se realiza en los puntos identificados *inicio* (start) y *final* (end) de las acciones del plan. Si se requiere mantener un intervalo que esté abierto en ambos extremos (*start* y *end*) se utiliza la construcción *over all*.

Por otro lado, si uno desea especificar que un hecho *p* se mantiene en el intervalo cerrado durante la duración de una acción durativa, entonces son requeridas las tres condiciones: *(at start p)*, *(over all p)* y *(at end p)*.

SEMÁNTICA DE PDDL2.1

En PDDL 2.1 se añadieron cuatro extensiones significativas con respecto a la planificación clásica y la semántica Lifschitz desarrollado para STRIPS. Estos son:

- La introducción del tiempo, de manera que los planes describen el comportamiento con respecto a una línea de tiempo real;
- Relacionada con la primera extensión, el tratamiento de la concurrencia, acciones que se pueden ejecutar en paralelo, que pueden conducir a planes que contienen procesos concurrentes que interactúan (aunque estos procesos se encapsulan en acciones durativas en PDDL2.1);
- Una extensión para manejar valores numéricos *fluents*;
- El uso de efectos condicionales, tanto solos como en combinación con todos las extensiones anteriores.

La semántica se basa en un conocido modelo de *transición de estados*. Los requisitos de la semántica pueden reducirse a cuatro elementos esenciales:

- 1. Para definir lo que es un *estado*. La introducción de tiempo y valores numéricos complican la definición usual de un *estado* como un conjunto de átomos.
- 2. Para definir cuando un *estado* satisface una fórmula proposicional que representa una condición o requisito objetivo de una acción. Una extensión de la interpretación habitual de un estado como una valoración en la que un átomo es verdadero si y sólo si el átomo está en el estado (la suposición del mundo cerrado) es necesario para controlar los valores numéricos en el estado.

- 3. Para definir la transición de estado inducido por la aplicación de una acción. La regla de actualización para el estado lógico debe complementarse con una explicación de las consecuencias para la parte numérica del estado.

- 4. Para definir cuándo dos acciones se pueden aplicar simultáneamente y cómo su aplicación *concurrente* afecta a la aplicación de dichas acciones de forma individual.

3.3 COMPETENCIAS INTERNACIONALES DE PLANIFICACIÓN

En la actualidad existen organizaciones que se encargan de fomentar el campo de la *planificación y la programación automática*. Tal es el caso de **ICAPS** (The International Conference on Automated Planning and Scheduling).

La Conferencia Internacional sobre Planificación y Programación Automática (ICAPS) es el principal foro para investigadores y profesionales en la *planificación y programación* - dos tecnologías que son fundamentales para áreas como la manufactura, sistemas de espacio, ingeniería de software, robótica, educación y entretenimiento. La conferencia ICAPS es resultado de la fusión de dos conferencias bianuales, la Conferencia Internacional sobre Planificación y Programación de Inteligencia Artificial (AIPS por sus siglas en inglés de International Conference on Artificial Intelligence Planning and Scheduling) y la Conferencia Europea de Planificación (ECP por sus siglas en inglés de European Conference on Planning) [24].

Los objetivos principales de ICAPS son fomentar el campo de la *planificación y la programación automática* mediante la organización de reuniones técnicas, incluida la **conferencia anual ICAPS**, a través de la organización de escuelas de verano, cursos y actividades de capacitación en varios eventos, a través de la organización de competencias de *planificación y la programación*, la evaluación comparativa y otros medios de promoción y evaluación del estado del arte en el campo, mediante la promoción de la participación

de jóvenes científicos en el campo a través de becas y otros medios, y mediante la promoción y difusión de publicaciones, los sistemas de *planificación y programación*, *dominios*, simuladores, herramientas de software y material técnico.

3.3.1 COMPETENCIA INTERNACIONAL DE PLANIFICACIÓN (IPC)

La **Competencia Internacional de Planificación** (IPC, por sus siglas en inglés de International Planning Competition) es un evento bi-anual organizado en el marco de ICAPS, el cual tiene varios objetivos, incluyendo el análisis y el avance del estado del arte en los sistemas de planificación automatizada; proporcionando nuevos conjuntos de datos para ser utilizados por la comunidad científica como puntos de referencia para la evaluación de diferentes enfoques para la planificación automatizada; haciendo hincapié en los nuevos temas de investigación en planificación; promoviendo la aceptación y aplicabilidad de la tecnología de planificación.

La IPC se divide en tres partes:

- La parte **determinista**, que tiene en cuenta la planificación totalmente determinista y observable (anteriormente llamada también la planificación çlásica"),
- La parte de **incertidumbre** que considera acciones no deterministas y probabilísticos en dominios totalmente observables, parcialmente observables o no observables,
- La parte de **aprendizaje**, donde los planificadores explotan el conocimiento dependiente del dominio que ha sido extraido de forma automática durante un periodo de entrenamiento offline.

3.4 ALGORITMOS DE PLANIFICACIÓN (PLANIFICADORES)

Para dar solución a los *modelos de planificación* se utiliza un *planificador de IA*, el cual es un algoritmo de propósito especial que utiliza un lenguaje de planificación formal, como PDDL, con una sintaxis, semántica y teoría de la demostración bien definidas [40].

La teoría de la demostración especifica qué es lo que se puede inferir de los resultados de las secuencias de acción y, por lo tanto, cuáles son los planes legales.

Estos *planificadores* a menudo son llamados *planificadores independientes del dominio*, esto es porque independientemente del modelo de planificación de que se trate, sea un modelo educativo, de scheduling, compras, manufactura, etc., pueden obtener una solución (un plan) a dicho modelo.

El desarrollo más sobresaliente de planificadores es presentado en el ICAPS, específicamente en las Competencias Internacionales de planificación (IPC), en donde compiten para ver el desempeño de estos planificadores en dominios de planificación (modelos de planificación) que presenta la IPC.

No pretendemos detallar todos los planificadores que han participado en las Competencias Internacionales de Planificación (IPC), sino aquellos que nosotros consideramos para la resolución de nuestro modelo de planificación. El criterio de selección, además que fueron de los más sobresalientes en la IPC, es que soportan los requerimientos de nuestro modelo de planificación, tales como acciones durativas, tipos (types), métricas numéricas (fluents), igualdades y constantes.

3.4.1 SGPLAN

Uno de los *planificadores* seleccionados para la resolución de nuestro modelo de planificación es SGPLAN que quedó en el primer lugar en la IPC del 2006 en la parte determinística de la competencia [23]. SGPlan particiona un problema de planificación grande en subproblemas, cada uno con sus propios sub-objetivos.

La partición de sub-objetivos es eficaz porque cada subproblema particionado implica un espacio de búsqueda sustancialmente menor que el del problema original. SGPlan desarrolló métodos para la detección de ordenamiento razonables entre sub-objetivos, un análisis de agenda de objetivos intermedios para descomponer jerárquicamente cada subproblema, un algoritmo de reducción de espacios de búsqueda para eliminar acciones irrelevantes en subproblemas, y una estrategia para llamar al mejor planificador para resolver

cada subproblema del nivel inferior [8].

3.4.2 LPG

El otro planificador seleccionado es LPG [20] que ha participado en la IPC y fue premiado como mejor planificador automatizado en el 2003 y posteriormente en la IPC 2004 fue premiado como mejor desempeño en dominios que implican *Timed Initial Literals* y en la calidad del plan (satisficing planning track).

Es un planificador estocástico basado en un algoritmo de *mejor primero* similar al que usa FF [22]. FF es el planificador independiente del dominio que se basa en búsquedas en el espacio de estados hacia adelante. LPG es recomendado para dominios que tienen cantidades numéricas y duraciones [10].

LPG produce planes de calidad multicriterio. El núcleo del sistema se basa en un método de búsqueda local estocástica y una representación basada en grafos llamada "Temporal Action Graphs" (TA-graphs). Dentro de las características de LPG es que soporta el lenguaje PDDL2.1 que soporta "acciones durativas" y "cantidades numéricas".

La búsqueda local está emergiendo como un método poderoso para tratar la planificación totalmente automatizada, aunque en principio este enfoque no garantiza la generación de planes óptimos. El esquema de búsqueda general de este planificador es Walkplan, un procedimiento de búsqueda local estocástica similar a la conocida Walk-sat. Dos de las más importantes extensiones en este planificador son el uso de *grafos de acción temporal* (TA-graphs) y algunas nuevas técnicas para guiar el proceso de búsqueda local. En un grafo-TA los nodos de las acciones están marcados con valores temporales que estiman el tiempo más corto cuando la acción correspondiente termina, mientras que los nodos hechos son marcados con valores temporales estimando el tiempo mas corto cuando el hecho correspondiente se convierte en verdadero. Un conjunto de restricciones de orden se mantienen durante la búsqueda para manejar las acciones mutuamente excluyentes y representar las restricciones temporales implícitas en las relaciones "causales" entre las acciones y el plan actual.

La nueva heurística explota alguna información de accesibilidad para sopesar los elementos (grafo-TA) en la búsqueda del vecindario que resuelven inconsistencias seleccionadas de un grafo-TA actual. La evaluación de estos grafos-TA se basa en la estimación del número de pasos de búsqueda necesarios para alcanzar una solución (un plan válido), el makespan estimado, y la estimación del costo de ejecución. El **LPG** es un planificador incremental, en el sentido de que produce una secuencia de planes válidos cada uno de lo que mejora la calidad de los anteriores. La calidad del plan se modela mediante la ejecución y costos temporales de una manera flexible (el usuario puede determinar la importancia relativa del criterio de la calidad del plan).

3.5 INVESTIGACIÓN DE OPERACIONES Y LOS MODELOS DE PROGRAMACIÓN MATEMÁTICA

La **Investigación de Operaciones** (IO) (conocida también como teoría de la toma de decisiones o programación matemática) es una rama de las matemáticas que consiste en el uso de *modelos matemáticos, estadística y algoritmos* con objeto de realizar un proceso de *toma de decisiones*.

Entre los métodos utilizados por la IO, los administradores utilizan las matemáticas y las computadoras para tomar decisiones racionales en la resolución de problemas.

En el enfoque científico de toma de decisiones, se requiere el uso de uno o más **modelos matemáticos**. Éstos son representaciones matemáticas de situaciones reales que se podrían usar para tomar mejores decisiones, o bien, simplemente para entender mejor la situación actual.

Un modelo de este tipo dicta el comportamiento para una organización que le permitirá alcanzar mejor su(s) meta(s).

Entre los elementos de un modelo están:

- **Funcion(es) objetivo**: En la mayoría de los modelos hay una función que deseamos maximizar o minimizar.
- **Variables de decisión**: Son variables cuyos valores están bajo nuestro control e influyen en el desempeño del sistema.
- **Restricciones**: En la mayor parte de las situaciones, sólo son posibles ciertos valores de las variables de decisión.

Existen diferentes modelos matemáticos que pueden realizarse, esto es según la naturaleza de las variables de decisión y cómo se defina la función objetivo y restricciones. Entre los diferentes tipos de modelos matemáticos están:

- **Modelos lineales y no lineales**: Supóngase que siempre que las variables de decisión aparecen en la función objetivo y en las restricciones de un modelo de optimización, están multiplicadas por constantes y acomodadas en forma de suma. Un modelo de esta forma es un *modelo lineal*. Si un modelo de optimización no es lineal, entonces es un *modelo no lineal*.
- **Modelos enteros y no enteros**: Si una o más variables de decisión deben ser enteros, entonces se dice que un modelo de optimización es un modelo entero. Si todas las variables de decisión son libres para asumir valores fraccionarios, entonces el modelo de optimización es un modelo no entero.
- **Modelos binario y entero mixto**: Si las variables de decisión sólo pueden tomar valores 0 ó 1, entonces el correspondiente modelo lineal entero es llamado modelo lineal binario. Si en un modelo lineal sólo algunas variables son restringidas a enteros, entonces tenemos un modelo lineal entero mixto.

- **Modelos determinísticos y estocásticos**: Supóngase que para cualquier valor de las variables de decisión, se conoce con certeza el valor de la función objetivo y si las restricciones se cumplen o no. Entonces se tiene modelo determinístico; de no ser así, se tiene un modelo estocástico. Considere como ejemplo la incertidumbre respecto a la demanda futura de un producto, quiere decir que para un programa de producción dado, no se sabe si se cumplirá a tiempo con la demanda. Esto lleva a pensar que es necesario un modelo estocástico para modelar la situación.

- **Modelos estáticos y dinámicos**: *Un modelo estático* es uno en el cual las variables de decisión no requieren sucesiones de decisiones para periodos múltiples. *Un modelo dinámico* es uno en el cual las variables de decisión sí requieren sucesiones de decisiones para periodos múltiples. En esencia, en el modelo estático se resuelve un problema luego de un solo intento, cuyas soluciones dictan valores óptimos de las variables de decisión en todos los puntos del tiempo.

  Una vez que se han descrito los diferentes modelos de programación matemática, como conclusión de esta sección, mencionamos que nosotros desarrollamos un modelo lineal entero mixto, determínistico y estático para la modelación de nuestro problema.

CAPÍTULO 4

METODOLOGÍA

4.1 METODOLOGÍA

La metodología utilizada para la generación de trayectorias de aprendizaje es la siguiente:

1. Se propone una metodología que considera la utilización de *planificación* de *inteligencia artificial* para obtener *trayectorias de aprendizaje*. Esta metodología consiste en:
   - El desarrollo de un *modelo de planificación* que represente el problema de la generación de trayectorias de aprendizaje.
   - Selección de algoritmos de planificación para dar solución a dicho modelo.
   - Utilización de los dos algoritmos de planificación seleccionados, en este caso son SGPLAN y LPG, para dar solución (generar planes) de los modelos de planificación. El análisis del comportamiento de estos dos algoritmos para dar solución a los modelos de planificación se ve en la sección de experimentación.
2. Se utiliza la programación matemática para ayudar a conocer la calidad de las soluciones arrojadas por los planificadores. Para esto se desarrolla un modelo matemático de programación lineal entera mixta que realiza la selección de actividades que minimicen el tiempo total (métrica utilizada en la experimentación).

3. Posteriormente, se realiza una metodología híbrida para la obtención de mejores soluciones. Esto es, se incluyen las soluciones del modelo matemático a los modelos de planificación.

4.2 PLANIFICACIÓN DE INTELIGENCIA ARTIFICIAL

La ***Planificación*** se define como el proceso de búsqueda y articulación de una secuencia de acciones que permitan alcanzar un objetivo [41].

Aunque las técnicas de planificación inteligente han sido útiles para sintetizar problemas complejos dado un estado inicial, una representación de objetivos y un conjunto de transiciones de estados posibles [12], es hasta muy recientemente, que se ha visto la generación de trayectorias de aprendizaje para estudiantes como un problema de Planificación.

El resultado de la aplicación de la *planificación* en los modelos educativos son *planes* (secuencia de acciones) que comprenden una programación de actividades a realizar en donde se han seleccionando aquellas que vayan de acuerdo a la métrica a optimizar y considerando su objetivo.

En la figura 4.1 mostramos el proceso de planificación que utilizamos para obtener las trayectorias de aprendizaje usando la*planificación de inteligencia artificial*. En primer lugar, se desarrolla un modelo de planificación en el lenguaje PDDL, se utilizan algoritmos de planificación para obtener soluciones de los modelos de planificación, y por último, como resultado del proceso de planificación se obtienen los planes, que son las trayectorias de aprendizaje.

4.2.1 BENEFICIOS DE UTILIZAR PLANIFICACIÓN DE IA

La Planificación de IA puede satisfacer de una manera mas robusta las necesidades de generación de ***trayectorias de aprendizaje***, ya que:

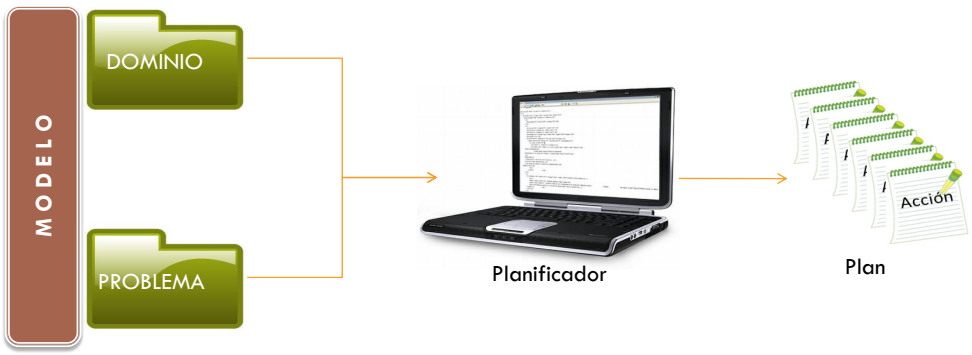


Figura 4.1: Proceso de planificación de inteligencia artificial.

- Recaba una gran cantidad de información acerca del mundo circundante para definir un estado inicial del entorno del problema con el cual habrá de enfrentarse.
- Define un conjunto de reglas de actuación en base a determinadas condiciones, precondiciones que una vez que se cumplen, permiten llevar a cabo una o varias acciones que generan efectos sobre el estado del mundo, es decir, definir el dominio por medio del cuál será posible cambiar el estado inicial de un problema para llegar hasta un estado que lo solucione.
- Establece una meta que suele representarse como una acción o conjunto de acciones que se quieran llevar a cabo, además de una serie de condiciones que debe cumplir el mundo circundante una vez que se hayan ejecutado dichas acciones meta.

4.3 PROBLEMA DE PLANIFICACIÓN EDUCATIVA

El problema de planificación educativa (PPE) puede definirse como la geneneración de una secuencia ordenada de actividades de aprendizaje para el estudiante (que llamamos trayectoria de aprendizaje) que permiten optimizar una métrica.

Necesitamos diseñar y formalizar un *modelo de planificación* con el fín de generar trayectorias de aprendizaje o planes educativos. Para esto consideramos formalizar

nuestros modelos educativos directamente con PDDL [15]. La motivación para utilizar PDDL es doble: En primer lugar para explorar un conjunto más amplio de las propiedades del dominio de planificación (modelo) con el fin de identificar aquellas que podrian incrementar la complejidad de encontrar una solución, y en segundo lugar para aprovechar los algoritmos de planificación de IA disponibles para generar las trayectorias de aprendizaje.

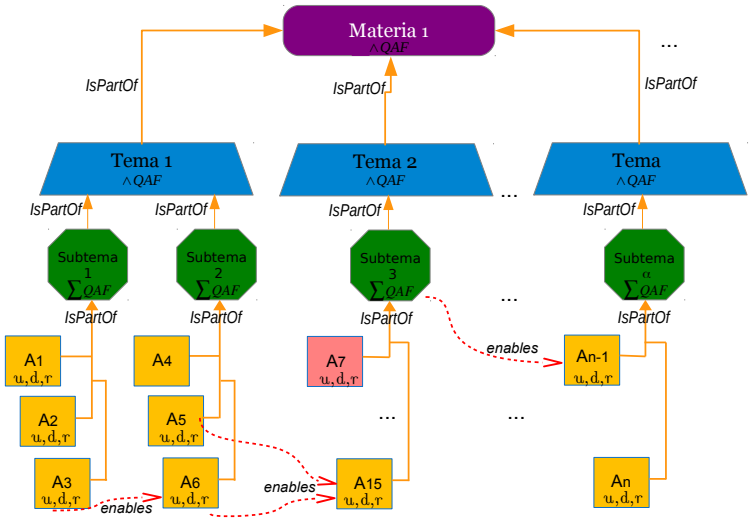


Figura 4.2: Una plantilla del modelo de tarea de aprendizaje para el problema de planificación educativa (PPE).

Uno de los principales objetivos del diseño de nuestro modelo de planificación es que sea simple. Queremos que el modelo sea fácil de codificar, pero al mismo tiempo lo suficientemente general para expresar un amplio conjunto de propiedades y diferentes tipos de plantillas de aprendizaje para el problema de planificación educativa (*PPE*). Una plantilla de aprendizaje es una descripción general estructurada de cómo se interrelacionan los objetivos de aprendizaje para capturar el plan de estudios que los estudiantes deben completar.

Representamos una plantilla de aprendizaje para el *problema de planificación educativa* como un grafo de descomposición de tareas, en donde las tareas pueden ser descompuestas en *subtareas* y *nodos hoja*. Podemos ver en la figura 4.2 un ejemplo gráfico de una plantilla general para representar el plan de estudios con nuestro mo-

delo. Un nodo hoja de la red (es decir, un nodo que no tiene hijos) corresponde a una *actividad de aprendizaje*; esto es, una actividad que el estudiante puede realizar para lograr los objetivos de aprendizaje. Por otro lado un nodo que no tiene padre corresponde a una tarea raiz de la estructura, es decir, el concepto de aprendizaje del nivel superior de la jerarquía de tareas. Esto podría representar a la materia o asignatura del plan de estudios del estudiante.

En la figura 4.2 tenemos un grafo de cuatro niveles que representan la *materia*(unidad de aprendizaje), *temas* (objetivos generales de la materia), *subtemas* (objetivos específico) y *actividades* a realizar para cumplir esos objetivos. Sin embargo, no es la única estructura que puede modelarse. Esto se logra mediante la aplicación de relaciones *IsPartOf* entre las subtareas. La idea es facilitar la construcción de plantillas de aprendizaje para tener en cuenta diferentes escenarios.

En el modelo cada *nodo tarea* está asociado con una *función de acumulación de calidad (QAF)* esto es, una función de evaluación que evalúa la recompensa (es decir, valor de calidad) del nodo. Utilizamos las relaciones *IsPartOf* para propagar la recompensa de un nodo padre desde la calidad de sus hijos. El concepto de calidad aquí es abstracto, podría relacionarse al score de los estudiantes, satisfacción, motivación, preferencias, o incluso connotaciones negativas (por ejemplo sanciones) como costo, tiempo o el uso de recursos. Esto debe ser definido por el diseñador del contenido del modelo. Estamos conscientes de que algunos conceptos pedagógicos no pueden ser cuantificados numéricamente. Sin embargo, el uso de QAFs permite aislar el proceso de calcular la información de recompensa, por lo que los educadores podrían codificar la estimación del valor. Para ejemplificar lo anterior, en una plantilla del problema EPP consideramos que cada actividad tiene una utilidad y que deseamos que la suma de estas utilidades esté entre un rango mínimo satisfactorio (que podría ser la calificación mínima aprobatoria) y un rango mayor, para no realizar actividades de más, por así decirlo. Ó como en trabajos relacionados que consideran la suma del tiempo de cada actividad como su QAF de manera que no exceda un tiempo límite determinado.

Como no es posible representar todos los escenarios posibles, nos enfocamos en uno para representar nuestro modelo de planificación que representa el problema *PPE*. En éste tenemos una jerarquía de cuatro niveles, como ya vimos en la figura 4.2. Aquí representamos tres clases de QAFs que nosotros diseñamos: la suma QAF_{Σ} y las conjunciones $QAF_{\bigwedge}$ y $QAF_{\bigwedge_{Kmin}}$. La suma QAF es uitlizada por los *subtemas* en el ejemplo gráfico, aquí agrega la recompensa al subtema por medio de las actividades (nodos hijos). Por otro lado, las conjunciones QAFs evalúan la recompensa a una decisión binaria utilizando una clásula lógica. Siguiendo el ejemplo, nuestro modelo puede agregar recompensa de las actividades a los subtemas, a partir de ahí para los temas y materia considerar la satisfacción total del objetivo dada por la acumulación de recompensa de los nodos hijos. Esto significa que el modelo requiere satisfacer una cláusula condición para considerar que el tema ha sido aprobado; es decir, que todos sus subtemas han acumulado suficiente recompensa por encima de una calificación o nota satisfactoria mínima. De manera que podemos ver que el modelo proporciona flexibilidad, ya que los QAFs se pueden combinar en cualquier nivel de la jerarquía de la red.

Existen relaciones adicionales que habilitan a otros nodos de la red. Estas relaciones las llamamos *enables*, la cual se especifican por medio de links direccionales entre un nodo fuente y una tarea objetivo. Representan el conjunto mínimo de requerimientos que un nodo fuente tiene que satisfacer con el objeto de considerar el nodo objetivo para la planificación. Estas relaciones pueden ser utilizadas para definir un conjunto parcial o total de precondiciones entre los nodos, y puedan ocurrir en cualquier nivel de la jerarquía. Por ejemplo se pueden utilizar para especificar que una $materia_x$ determinada no está disponible para un estudiante hasta que la $materia_y$ ha sido llevada por el estudiante y aprobada; para definir que un subtema dado requiere conocimientos previos de un tema; o para declarar que una actividad de aprendizaje específica requiere que un estudiante complete una o múltiples actividades de aprendizaje. Estas relaciones *enables* pueden ser utilizadas para especificar las restricciones de precedencia entre los nodos en la jerarquía para generar trayectorias de aprendizaje personalizadas que cumplan con los planes de estudio.

Aunque PDDL no soporta como tal las QAFs y la jerarquía de tareas, hemos sido capaces de modelarlas calculando explícitamente las relaciones entre las tareas a través de *predicados* relacionales entre las *acciones* y *objetos* en el modelo de planificación; y mediante el uso de *funciones* en las acciones, calculamos la calidad de una manera similar al *TAEMS* (Task Analysis, Environment Modeling, and Simulation) que se utiliza para modelar sistemas basados en agentes complejos ([11]). Las relaciones *enables* son modeladas mediante *predicados* ó el uso de precondiciones temporales métricas en las acciones ([15]).

El objetivo general de nuestros modelos es representar el plan de estudios y objetivos de aprendizaje que los estudiantes deben cumplir, tal que los algoritmos de planificación independientes del dominio (que llamamos planificadores ó simplemente algoritmos de planificación) puedan generar planes educativos personalizados que, en la cantidad mínima de tiempo permita a los estudiantes completar satisfactoriamente su plan de estudios. El diseño del modelo nos permite incluir las preferencias del usuario, por ejemplo, podemos pedir que se genere un plan que alcance una nota mínima aprobatoria en la menor cantidad de tiempo, donde el nivel de logro (nota mínima) es definida por el usuario. Este grado de libertad permite codificar preferencias del usuario en términos de requerimientos objetivo, por lo tanto la personalización de las trayectorias de aprendizaje. Aunque esto no es lo único que define la personalización, para esto tambien influye el estado inicial de un estudiante, es decir, si ya ha aprobado una materia determinada, ha realizado actividades anteriormente ó si desea realizar solo un conjunto de objetivos de aprendizaje.

4.4 FORMALIZACIÓN DEL MODELO

PPE es modelado como una plantilla de una red de tareas de aprendizaje L_T.

Definition 1 *Una plantilla de una red de tareas de aprendizaje L_T es una 6-tupla:*

$$L_T = \langle R, N, A, Q, E, Kmin \rangle, \textit{ donde}$$

R es un conjunto de recursos educativos. Para cada $r \in R$, $MAX(r)$ representa la cantidad de recursos r disponibles en el sistema educativo, como por ejemplo, número de computadoras, número de licencias de software, número de libros de texto de un tema en particular, etc.

N es un conjunto de nodos de tareas que representan los objetivos de aprendizaje del programa de estudio. $N = T \cup L_A$ es el conjunto de todos los nodos tarea, donde T es el conjunto de todas las tareas de aprendizaje que tienen subtareas en la jerarquía de la red, mientras L_A es el conjunto de actividades de aprendizaje (es decir, los nodos de hoja) en la red. Adicionalmente, $S \subset T$ especifica el subconjunto de nodos raíz para el modelo. Sea $C(n) \subset N$ el conjunto de hijos para el nodo n. Por lo tanto, $C(n) = \emptyset$ si y solo si $n \in L_A$. Además, si $n \notin C(n')_{\forall n' \in N}$ entonces $n \in S$; es decir, n es un nodo raíz. Como se mencionó anteriormente, las actividades de aprendizaje $l \in L_A$ representan las actividades que los estudiantes pueden realizar para acumular una utilidad o recompensa (por ejemplo, score) durante el proceso de aprendizaje. Cada actividad l tiene tres mapeos $\mathbf{u}, \mathbf{d}$, y $\mathbf{r}$, como se puede ver en la figura 4.2:

- $\mathbf{u}$ representa el mapeo $\mathbf{u} : L_A \longrightarrow \mathbb{N}^*$, donde $\mathbf{u}(l)$ representa el valor de utilidad dada a la actividad de aprendizaje l en el modelo. Miden la contribución de las actividades de aprendizaje en el score global de sus nodos padres. Tenga en cuenta que este mapeo es una función de utilidad o recompensa, pero como mencionamos anteriormente la recompensa puede relacionarse con diferentes valores de utilidad. En nuestro ejemplo relacionamos $\mathbf{u}$ con el score, por lo tanto $\{0 \leq \mathbf{u} \leq 100\}$.
- $\mathbf{d}$ representa el mapeo $\mathbf{d} : L_A \longrightarrow \mathbb{N}^*$, donde $\mathbf{d}(l)$ representa la duración que la actividad de aprendizaje l toma para ser completada.
- $\mathbf{r}$ representa el mapeo $\mathbf{r} : L_A \longrightarrow \mathbb{N}^*$, donde $\mathbf{r}(l)$ representa la cantidad de recurso $\mathbf{r}$ necesario para realizar con éxito la actividad l.

Nuestros modelos consideran la inclusión de actividades de aprendizaje obligatorias $M \subset L_A$ (vea por ejemplo la actividad A_7 de la figura 4.2), donde se

pueden considerar tareas (por ejemplo, examenes) que son requeridas durante el curso. Estas actividades pueden ser vistas como restricciones duras en los planes educativos generados.

A es un conjunto de relaciones jerárquicas ($IsPartOf$). Un arco $a_{i,j}$ entre dos nodos tarea i y j existe iff i *es parte de* la tarea de aprendizaje j en el plan de estudios, y $a_{j,i} \notin A$. Esto implica que i pertenece a la subjerarquía inmediata de j, así $i \in C(j)$.

Q es un conjunto de Funciones de Acumulación de Calidad (QAFs) ([11]), que definen cómo las utilidades (score) se agregan y evalúan a través de la red. Aunque cada actividad de aprendizaje $l \in L_A$ tiene un valor de *utilidad*, todavía tenemos que estimar la ganancia de utilidad al considerar este tipo de actividades en el resto de las tareas de aprendizaje en la jerarquía de la red. Nosotros presentamos tres QAFs en la versión actual de nuestro modelo:

$$QAF_{\Sigma}(n) = \begin{cases} \mathbf{u}(n), & \text{if } C(n) = \emptyset \wedge n \in PLAN \\ \sum_{n' \in C(n)} QAF_{\Sigma}(n'), & \text{else if } C(n) \neq \emptyset \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

$$QAF_{\bigwedge_{Kmin}}(n) = \begin{cases} 1, & \text{if } \forall_{n' \in C(n)} QAF_{\Sigma}(n') \geq Kmin \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

$$QAF_{\bigwedge}(n) = \begin{cases} 1, & \text{if } \forall_{n' \in C(n)} QAF_{\bigwedge_{Kmin}}(n') \geq 1 \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

La definición recursiva de la equación 4.1 para QAF_{Σ} establece que la utilidad de un nodo n se agrega a partir de las utilidades de sus hijos considerados para la ejecución del *PLAN* (Vea la definición 2). Las ecuaciones 4.2 y 4.3 son cláusulas de satisfacción de objetivo (es decir, QAFs conjuntivas). La primera garantiza que la utilidad para los nodos está por encima de un cierto valor de

preferencia mínimo, mientras que la segunda marca los nodos como *aprobados* o *no aprobados* dando valores binarios de sus hijos.

E es el conjunto de condiciones *habilitantes (enabling)* entre nodos tarea (actividades de aprendizaje) en nuestra red. Un enlace directo $e_{i,j}$ pertenece a E si el nodo tarea *i habilita* (proporciona un subconjunto de los requisitos necesarios por) la tarea j. En la versión actual de nuestro modelo, consideramos condiciones *Satisfacción de Objetivos* $({}_gS)$ y *Satisfacción de Métricas* $({}_mS)$:

1) ${}_gS(e_{i,j}) \Longleftrightarrow j \Rightarrow i : i, j \in L_A$, las condiciones de *satisfacción de objetivos* implican que para considerar la actividad de aprendizaje j para su ejecución, deberá ser el caso que la actividad i se ha completado con éxito. Una actividad de aprendizaje j puede necesitar más de una condicion *enabling*, en ese caso cada uno de sus *habilitadores* deberán ser completados antes de j. Esta condición obliga la satisfacción total de los *habilitadores* para modelar situaciones donde ciertas trayectorias de aprendizaje imponen restricciones educativas duras.

2) ${}_mS(e_{i,j}) \Longleftrightarrow j \Rightarrow QAF_{\bigwedge_{Kmin}}(i) : i, j \in N$, las condiciones de *satisfación de métricas* implica que para considerar la actividad de aprendizaje j para su ejecución, deberá ser el caso que un nodo en particular i en la jerarquía de la red, ha acumulado una cantidad mínima de utilidad, la cual es una preferencia del usuario proporcionada al modelo. Esta condición permite la satisfacción parcial de objetivos, ya que no requiere completar todas las actividades de aprendizaje en la subjerarquía de i para soportar j. De hecho, esta condición proporciona flexibilidad al modelo para soportar la generación de planes educativos bajo diferentes niveles requeridos de rendimiento.

Kmin representa la calificación aprobatoria mínima que los planes educativos tendrán que cumplir. Podemos ajustar este parámetro en el modelo para generar planes educativos con diferentes niveles de rendimiento.

Definition 2 *Una solución a una instancia del problema de planificación educativa*

es un $<PLAN>$, que es una secuencia de actividades de aprendizaje $\{l_1, l_2, ..., l_n\} \in L_A$ con la función objetivo de minimizar la duración total, esto es:

$$min \sum_{i=1}^{n} \mathbf{d}(l_i), l_i \in PLAN \tag{4.4}$$

tal que:

1. $\forall s \in S, \bigwedge QAF(s) \geq 1$*, el estudiante pasa cada materia (nodos raiz) del plan de estudios , y*
2. $\forall l_i \in PLAN, \forall l_j \in PLAN$ *si* $l_i < l_j$ *en la secuencia del PLAN, entonces* ${}_gS(e_{j,i}) \notin E$*, no viola ninguna de las condiciones de satisfacción objetivo en el modelo.*

Nuestra definición de solución implica una secuencia de actividades de aprendizaje, junto con los recursos necesarios para llevarlas a cabo, que a lo largo del tiempo permitan alcanzar un conjunto de objetivos de aprendizaje. En otras palabras, estamos tratando de encontrar el plan de duración más corta que garantice a los estudiantes avanzar en su aprendizaje dada una evaluación de calidad, un objetivo importante para estudiantes con horarios limitados. Nuestro enfoque no esta limitado a la optimización de trayectorias de aprendizaje con respecto al tiempo. Los modelos podrían maximizar o minimizar QAFs definidas por el usuario. Recordemos que las QAFs permiten encapsular y agregar utilidad, éstas definen como la utilidad es propagada en la jerarquía de red. En consecuencia, podemos directamente optimizarlas. Para el propósito de este estudio, evaluamos nuestro modelo con la definición 2 que presentamos anteriormente.

4.4.1 EJEMPLO DEL MODELO DE *Planificación* Y SU REPRESENTACIÓN EN PDDL

Podemos ver un ejemplo gráfico de la plantilla del modelo presentado en la figura 4.2, la cual contiene la información de una materia en la figura 4.3. En este ejemplo, el conjunto de tareas de aprendizaje es

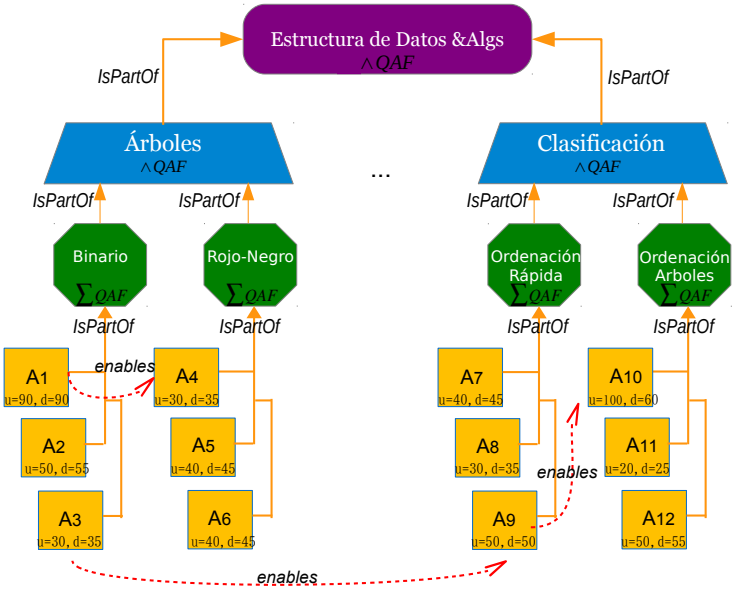


Figura 4.3: Plantilla de un modelo de tareas de aprendizaje instanciada con datos educativos.

$N = \{EstructuradeDatos\&Algs,\ Arboles,\ Clasificación,\ Binario,\ Rojo-Negro,\ OrdenaciónRápida,\ OrdenaciónArboles,\ A_1,\ ...,\ A_{12}\}$. El conjunto de relaciones jerárquicas $IsPartOf$ es $A = \{(A_1, Binario), (A_2, Binario), (A_3, Binario),\ (A_4, Rojo-Negro),\ (A_5, Rojo-Negro),\ (A_6, Rojo-Negro),\ (A_7, OrdenaciónRápida),\ (A_8, OrdenaciónRápida),\ (A_9, OrdenaciónRápida),\ (A_{10}, OrdenaciónArboles),\ (A_{11}, OrdenaciónArboles),\ (A_{12}, OrdenaciónArboles),\ (Binario, Arboles),\ (Rojo-Negro, Arboles),\ (OrdenaciónRápida, Clasificación),\ (OrdenaciónArboles, Clasificación),\ (Arboles, EstructuradeDatos\&Algs),\ (Clasificación, EstructuradeDatos\&Algs)\}$. Los QAFs del modelo son representados por el conjunto $Q = \{QAF_{\Sigma}(Binario),\ QAF_{\Sigma}(Rojo-Negro),\ QAF_{\Sigma}(OrdenaciónRápida),\ QAF_{\Sigma}(OrdenaciónArboles),\ QAF_{\wedge_{Kmin}}(Arboles),\ QAF_{\wedge_{Kmin}}(Clasificación),\ QAF_{\wedge}(EstructuradeDatos\&Algs)\}$. Finalmente, el conjunto de relaciones habilitantes (enabling) está compuesto de $E = \{{}_gS(A_1, A_4), {}_gS(A_3, A_9), {}_gS(A_9, A_{10})\}$. Por simplicidad asumimos que hay cuatro diferentes tipos de recursos educativos, esto es $R = \{r_0, r_1, r_2, r_3\}$, y la disponibilidad es ilimitada. Podemos ver en la figura 4.3 los valores de duración y

utilidad para cada actividad de aprendizaje. La calificación mínima para aprobar en este ejemplo es de $Kmin = 70$.

Utilizamos la plantilla del modelo instanciado, la cual representa nuestra estructura del modelo de tareas de aprendizaje, con los datos del plan de estudios por parte del usuario para codificar el *modelo de planificación* en PDDL. Como se mencionó anteriormente, PDDL es un lenguaje centrado en las acciones que representa la física de un problema a través de la definición de pre y post-condiciones que describen la aplicabilidad y efectos de las acciones. Por lo tanto, existe una correspondencia uno a uno entre las tareas de aprendizaje en nuestra estructura del modelo y acciones PDDL. Utilizamos los modelos de planificación en PDDL generados y los algoritmos de planificación (Planificadores) para generar las trayectorias de aprendizaje (es decir, planes) para *PPE*.

Se pueden distinguir dos clases diferentes de actividades de aprendizaje en nuestros modelos; actividades con y sin condiciones habilitantes. La plantilla PDDL para modelar las actividades sin condiciones habilitantes se puede ver en la figura 4.4. La plantilla de la acción tiene 4 parámetros de entrada. El estudiante *s* que realiza la actividad de aprendizaje *l* que *es parte de* (isPartOf) el subtema *o*, y requiere de recursos *r*. La duración de la actividad de aprendizaje *l* se especifica en la línea 2 de la descripción de la plantilla mediante el uso de *funciones* PDDL (por ejemplo, *laDuration* es una función que toma la actividad *l* como un parámetro de entrada y devuelve la *duración* de *l*). La cláusula de condición, que comprende las líneas 4 a 11, define las condiciones necesarias que deben ser verdaderas para que la acción sea aplicable. La línea 5 identifica a la actividad *l* que pertenece a la clase de actividades que no tienen habilitadores. Las líneas 6 y 7 validan que el estudiante *s* está libre (es decir, *s* no está ocupado realizando otra actividad), y que la actividad *l* no se ha realizado anteriormente. La línea 8 identifica la relación *isPartOf* de la actividad de aprendizaje, que es necesaria para calcular la suma de los valores score en la tarea padre correspondiente *o* (es decir, un subtema en nuestro ejemplo). Las líneas 9 y 10 determinan el tipo de recurso que la actividad requiere, y comprueba la disponibilidad de recursos. La última línea de la cláusula de condiciones establece

```
1:  (:durative-action EXEC-LA-noReqs
2:  :parameters (?s - student ?l - LA ?o - subtheme ?r - resource)
3:  :duration (= ?duration (laDuration ?l))
4:  :condition (and
5:          (at start (noReqs ?l))
6:          (at start (free ?s))
7:          (at start (not (done ?l ?s))
8:          (at start (isPartOfSubtheme ?l ?o))
9:          (at start (needsResource ?l ?r))
10:         (at start (> (quantity-resource ?r) 0))
11:         (at start (> (maxgrade-subtheme ?o)(valueLA ?l))))
12:     :effect (and
13:         (at start (not(free ?s)))
14:         (at start (decrease (quantity-resource ?r) 1))
15:         (at end (increase (quantity-resource ?r) 1))
16:         (at end (increase (score ?o ?s) (valueLA ?l)))
17:         (at end (decrease (maxgrade-subtheme ?o)(valueLA ?l)))
18:         (at end (done ?l ?s))
19:         (at end (free ?s)))
20:     )
```

Figura 4.4: Plantilla PDDL para las actividades de aprendizaje sin condiciones habilitantes.

que para que una actividad de aprendizaje sea aplicable su valor de utilidad debe ser capaz de contribuir en la utilidad agregada de su subtema. Para esta versión de nuestros modelos, ponemos un límite de 100 en la cantidad de utilidad que cada subtema puede adquirir. No hay ninguna razón para seguir trabajando en un subtema del plan de estudios, si el estudiante ha demostrado competencia en él.

Las líneas 12 a la 19 especifican los efectos de la realización de la actividad de aprendizaje. Puesto que, se trata de una actividad temporal, al inicio del intervalo de ejecución del estudiante se convierte en ocupado y disminuye de disponibilidad de recursos (véase [15] para una descripción completa de la semántica de acciones temporales en PDDL). Al final del intervalo de ejecución, aumenta la disponibilidad de recursos. Por otra parte, el valor de la utilidad de la actividad de aprendizaje se agrega al subtema (reward) (línea 16), y se utiliza para disminuir la cantidad de utilidad que todavía se puede adquirir en el subtema (línea 17). Observe que los valores quedan agregados a los subtemas porque en nuestro ejemplo tales nodos requieren funciones de acumulación QAF_{Σ}. Por último, la actividad de aprendizaje está marcado como completada, y el estudiante es establecido libre (disponible) para poder participar en otra actividad.

Para los casos en que las actividades de aprendizaje son habilitadas mediante la satisfacción objetivo o condiciones de satisfacción métrica, sus parámetros de entrada y cláusula de precondiciones cambian tomando en cuenta las tareas de aprendizaje habilitadoras. En nuestro ejemplo de la figura 4.3, hay una relación habilitante de la actividad de aprendizaje A_1 a la A_4. Para representar dicha relación, la plantilla PDDL de la figura 4.4 consideraría un parámetro de entrada adicional l' (es decir, el habilitador) para instanciar con el valor de A_1. Entonces, sería verificar si dicho habilitador se ha completado con éxito antes de comenzar l (instanciado con A_4). La verificación se lleva a cabo con una cláusula de condición previa adicional $(done\ ?l'\ ?s)$. Si hay más de un habilitador para una determinada actividad l, sólo tenemos que añadir las variables necesarias, y las cláusulas de condiciones previas para la verificación. Como se puede ver, los modelos de planificación propuestos son muy flexibles y pueden ser modificados fácilmente para capturar las restriccio-

```
1:  :durative-action PASS-Theme-Trees
2:  :parameters (?s - student)
3:  :duration (= ?duration 1)
4:  :condition (and
5:     (at start (>= (score Binary ?s)(Kmin DataStructuresAlgs)))
6:     (at start (>= (score Red-Black ?s)(Kmin DataStructuresAlgs))))
7:  :effect (and
8:     (at end (passed-Theme Trees DataStructuresAlgs ?s)))
9:)
```

Figura 4.5: Acción PDDL para evaluar si un tema en particular ha sido aprobado en nuestro ejemplo.

nes del plan de estudios.

Recordemos que las condiciones de satisfacción métrica de la forma ${}_mS(e_{i,j})$ se codifican utilizando funciones $QAF_{\bigwedge_{Kmin}}$. Modelamos esta condición en nuestro modelo de planificación con la instroducción de una cláusula adicional en la definición de precondiciones de una actividad de aprendizaje j. Específicamente, la siguiente cláusula es insertada $(at\ start\ (>\ (score\ ?i\ ?student)\ (Kmin)))$. La variable i, la cual representa una tarea, se pasa como un parámetro de entrada, y $Kmin$ es una constante (dada por el usuario) que establece el score límite que requiere una actividad de aprendizaje particular para ser ejecutada.

Podemos ver en la figura 4.5 la implementación PDDL de la función $QAF_{\bigwedge_{Kmin}}$ para una determinada tarea t. Siguiendo nuestro ejemplo, t toma el valor de *Arboles*, que es un tema de nuestro plan de estudios. La acción garantiza, a través de sus cláusulas conjuntivas de precondiciones, que cada subtarea en la jerarquía de *Arboles* ha acumulado un valor de utilidad mínimo de $Kmin$. Si ese es el caso, la tarea se marca como aprobada por el estudiante. La descripción PDDL de $QAF_{\bigwedge}$ para evaluar la satisfacción objetivo es muy similar a la representación $QAF_{\bigwedge_{Kmin}}$. Un ejemplo de la acción PDDL correspondiente se puede ver en la figura 4.6. La acción evalua si cada

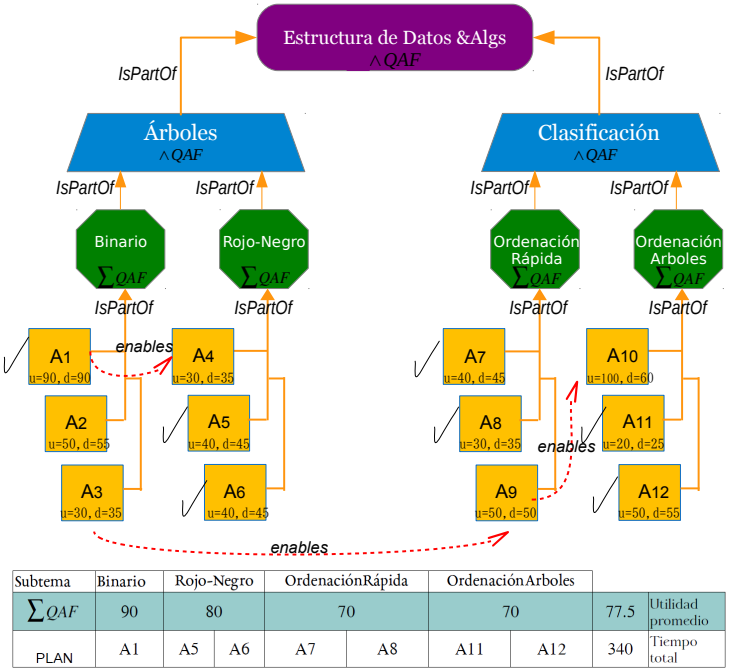
```
1:  :durative-action PASS-DataStructuresAlgs
2:  :parameters (?s - student)
3:  :duration (= ?duration 1)
4:  :condition (and
5:     (at start (passed-Theme Trees DataStructuresAlgs ?s))
6:     (at start (passed-Theme Sorting DataStructuresAlgs ?s))
7:  :effect (and
8:     (at end (passed-Subject DataStructuresAlgs ?s))
9:)
```

Figura 4.6: Acción PDDL para evaluar si una materia en particular ha sido aprobada.

subtarea (es decir, temas en nuestro ejemplo) en la jerarquía ha sido aprobada, si ese es el caso, el nodo padre (por ejemplo, una materia) es marcada como aprobada por el estudiante.

En la figura 4.7 se muestra un plan solución de nuestro ejemplo. El plan satisface la condición de satisfacción objetivo $QAF_{\bigwedge}$ de la definición 2, es decir, el nodo raíz (una materia) es aprobada porque sus subtareas son marcadas como aprobadas. Esta condición es representada con la acción PDDL de la figura 4.6. Observe que el apoyo a esta decisión se basa en los valores de las subtareas en la jerarquía. En nuestro ejemplo, los valores de las subtareas son calculados con funciones $QAF_{\bigwedge_{Kmin}}$ (vea la descripción PDDL en la figura 4.5), la cual garantiza que el plan acumula suficiente utilidad de recompensa (en este ejemplo en particular, mayor o igual a 70).

Podemos observar en este ejemplo lo flexible que es nuestro modelo. El diseñador puede combinar cualquiera de los QAFs soportados en la jerarquía de la red para evaluar el desempeño de los estudiantes, lo que facilita la representación de escenarios realistas. Para el ejemplo presentado, la solución se calcula utilizando el sistema de planificación SGPLAN ([**?**]), un planificador independiente de dominio greedy que considera la partición de sub-objetivos. Dada la naturaleza greddy de SGPLAN,



Subtema	Binario	Rojo-Negro		OrdenaciónRápida		OrdenaciónArboles			
$\sum QAF$	90	80		70		70		77.5	Utilidad promedio
PLAN	A1	A5	A6	A7	A8	A11	A12	340	Tiempo total

Figura 4.7: Ejemplo de una solución (plan) de la trayectoria de aprendizaje.

su solución no es óptima. La duración del plan óptimo es de 330 unidades de tiempo en este ejemplo. Para darnos cuenta de la optimalidad de los planes, desarrollamos un modelo de programación matemática para la solución de *PPE*, omitiendo la jerarquía y ordenación de actividades.

En la sección de evaluación, ofrecemos nuestro análisis y comparación del comportamiento de los algoritmos de planificación y nuestro modelo de programación matemática para la solución del problema.

4.5 MODELO MATEMÁTICO

Dado que estamos interesados en conocer la diferencia de optimalidad en las soluciones obtenidas por los algoritmos de planificación, desarrollamos un modelo de programación lineal entera mixta. En éste relajamos las restricciones de ordenamiento de las actividades. No considera la jerarquía de la red, únicamente realiza la selección de aquellas actividades que optimicen la métrica seleccionada, en nuestro caso es minimizar el tiempo total (utilizado en la experimentación). Además de

que considera las actividades que son obligatorias, y si existe una relación de precedencia entre actividades, se asegura que ambas actividades estén incluidas en el plan.

Presentamos a continuación el modelo matemático desarrollado.

4.5.1 SUPOSICIONES

- Las actividades se realizan una sola vez.
- De acuerdo con la métrica a optimizar habrá actividades que no sean seleccionadas.
- Todas las actividades cuentan con una duración y utilidad (score).
- No se considera la secuenciación (órden) de actividades.

4.5.2 PARÁMETROS

1. u_{ij}: Valor de utilidad dada a la actividad de aprendizaje i del subtema j.
2. d_{ij}: Duración de la actividad de aprendizaje i del subtema j.
3. $Kmin$: Calificación aprobatoria mínima que los planes educativos tendrán que cumplir en cada subtema.
4. $Kmax$: Calificación máxima que un estudiante puede obtener por subtema.

4.5.3 CONJUNTOS

1. M: Conjunto de actividades de aprendizaje obligatorias, tal que $m_{ij} \in M$ si la actividad de aprendizaje i del subtema j es obligatoria.
2. Ω: Matriz binaria de tamaño $n \times n$ actividades de aprendizaje, donde $\omega_{ii'}$=1 si la actividad de aprendizaje i habilita la actividad de aprendizaje i', and $\omega_{ii'}$=0 en otro caso.

4.5.4 VARIABLE DE DECISIÓN Y VARIABLE AUXILIAR

1. $x_{ij} = \begin{cases} 1 & \text{si la actividad } i \text{ del subtema } j \text{ esta completada} \\ 0 & \text{en otro caso} \end{cases}$

2. y_j: Variable auxiliar que representa el score acumulado obtenido en el subtema j.

$i \in \mathbb{N}, \quad i = 1, ..., n$ actividades de aprendizaje

$j \in \mathbb{N}, \quad j = 1, ..., \alpha$ subtemas

4.5.5 MODELO MATEMÁTICO

Función objetivo

$$\text{mín}\, z = \sum_{i=1}^{n} \sum_{j=1}^{\alpha} d_{ij} x_{ij} \tag{4.5}$$

Restricciones:

$$\sum_{i=1}^{n} u_{ij} x_{ij} \geq Kmin \qquad j = 1, 2, \ldots, \alpha \tag{4.6}$$

$$\sum_{i=1}^{n} u_{ij} x_{ij} \leq Kmax \qquad j = 1, 2, \ldots, \alpha \tag{4.7}$$

$$y_j = \sum_{i=1}^{n} u_{ij} x_{ij} \qquad j = 1, 2, \ldots, \alpha \tag{4.8}$$

$$x_{i'j'} \leq \omega_{ii'} x_{ij} \qquad i, i' = 1, 2, \ldots, n; j, j' = 1, 2, \ldots, \alpha \tag{4.9}$$

$$x_{ij} = 1 \qquad \forall m_{ij} \in M \tag{4.10}$$

$$x_{ij} \in \{0, 1\} \qquad i = 1, 2, \ldots, n; j = 1, 2, \ldots, \alpha \tag{4.11}$$

$$y_j \geq 0 \qquad j = 1, 2, \ldots, \alpha \tag{4.12}$$

La función objetivo (4.5) es minimizar el tiempo total de las actividades a realizar. El conjunto de restricciones (4.6) asegura que la suma del valor de utilidad de las actividades de aprendizaje seleccionadas son mayor o igual que una calificación aprobatoria mínima por subtema. Las restricciones de (4.7) son similares a las anteriores, pero para el límite superior de la calificación por cada subtema. Este conjunto de restricciones pudiera parecer trivial para la función objetivo z, ya que en ella consideramos el tiempo, pero es importante tener en cuenta que el conjunto de actividades seleccionadas, que minimicen el tiempo total, consideren que éstas aseguren un score o utilidad por subtema. El conjunto de restricciones (4.8) proporciona información sobre la calificación acumulada obtenida en cada subtema. Las restricciones (4.9) garantizan que cada relación habilitante entre las actividades de aprendizaje sean consideradas en la solución, es decir, si existe una relación de precedencia entre dos actividades, aunque no se considere cual se presenta primero y cual después, sí que si una es seleccionada para estar en el plan, la otra tambien. Por otro lado, las restricciones (4.10) aseguran que cada actividad de aprendizaje obligatoria sea incluida en el plan. Por último las restricciones (4.11, 4.12) establecen la naturaleza de las variables.

CAPÍTULO 5

COMPLEJIDAD DEL PROBLEMA

5.1 INTRODUCCIÓN

Un **problema** es un *conjunto de instancias* al cual corresponde un *conjunto de soluciones*, junto con una *relación* que asocia para cada instancia del problema un subconjunto de soluciones (posiblemente vacío) [43].

Los **problemas** pueden ser clasificados en:

- **Problemas de decisión**: La respuesta es "sí" o "no". La tarea es *decidir* **"sí" o "no"** la *relación* entre instancias y soluciones asigna un subconjunto vacío a una instancia dada. Por lo que si existen soluciones, la respuesta a la pregunta del problema es "si"; y si el subconjunto es vacio, la respuesta es "no".

  Es decir, un problema de decisión es aquel en el que las respuestas posibles son "si" o "no".

- **Problemas de optimización**: La pregunta es del tipo: "cuál es el mejor valor posible" o "con qué configuración se obtiene el mejor valor posible".

  Para problemas de optimización, la *instancia* está compuesta por un conjunto de configuraciones, un conjunto de restricciones, y una función objetivo que asigna un valor (real) a cada instancia. Si las configuraciones son discretas, el problema es combinatorial. La tarea es identificar cuál de las configuraciones factibles (las que cumplen con todas las restricciones) tiene el mejor valor de la función objetivo. Depende del problema si el mejor valor es el mayor (problema de maximización) o el menor (problema de minimización).

La configuración factible con el mejor valor se llama la solución óptima de la instancia.

Si deseamos conocer cual es la complejidad de un *problema*, debemos tener conocimiento acerca de las Máquinas de Turing (TM) y las clases de complejidad computacional que existen.

La Máquina Turing es un modelo formal de computación. Puede simular cualquier algoritmo con pérdida de eficiencia insignificante utilizando una sola estructura de datos. Esa estructura es una sucesión de símbolos escrita en una cinta (infinita) que permite borrar e imprimir símbolos.

Definición formal de una TM

$M = (K, \Sigma, \delta, s)$

- un conjunto finito de *estados* K, $s \in K$,
- un *alfabeto* finito de *símbolos* Σ así que $\sqcup, \triangleright \in \Sigma$ ($\sqcup$ es un espacio vacio y $\triangleright$ es el inicio de la cinta),
- una *función de transición*
  δ:$K \times \Sigma \to (K \cup \{\text{“alto”, “si”, “no”}\}) \times \Sigma \times \{ \to, \leftarrow, - \}$,
- los *estados* “alto”, “sí” y “no”
- direcciones del puntero: $\to$ (derecha), $\leftarrow$ (izquierda) y $-$ (sin mover).

Resolver un problema de decisión por una TM es *decidir* un *lenguaje* que consiste de representaciones de las instancias del problema que corresponden a la respuesta “sí”.

Una máquina Turing M *decide* el lenguaje L *si y sólo si* para toda sucesión $x \in (\Sigma \setminus \{ \sqcup \})$ aplica que si $x \in L$, $M(x)$ =“si” y si $x \notin L$, $M(x)$=“no”.

Debemos distinguir además entre:

- Máquina de Turing Determinista: Para cada par (estado, símbolo), existe como máximo una transición a otro estado.

- Máquina de Turing No Determinista: Existe al menos un par (estado, símbolo), con más de una transición a estados diferentes.

La **complejidad computacional** estudia el orden de complejidad de un algoritmo que resuelve un problema *decidible*. Para ello, considera 2 tipos de *recursos* requeridos durante el cómputo para resolver un problema:

- **Tiempo**: Número de pasos base de ejecución de un algoritmo para resolver un problema.
- **Espacio**: Cantidad de memoria utilizada para resolver un problema.

La complejidad de un algoritmo se expresa como función del tamaño de *entrada* del problema, *n*

En esta sección se presenta el análisis de complejidad de nuestro problema (PPE), es decir, se realiza una reducción de un problema del cual ya se conoce su complejidad y se comprueba que nuestro problema es al menos igual de difícil que el problema conocido.

5.2 ANÁLISIS DE COMPLEJIDAD

Dado un problema en NP, todo lo que necesitamos hacer es mostrar que algún problema NP-completo conocido puede ser transformado a él.

Entonces podemos decir que el *Problema de Planificación Educativa (PPE)* es al menos tan difícil como el problema Subset Sum (SS) si SS se reduce a PPE. Decimos que SS se reduce a PPE si hay una transformación R que, para cada entrada x de SS, produce una entrada equivalente R(x) de PPE.

En [16] y [37] menciona el proceso de elaboración de una prueba de NP-completitud para un problema de decisión. Estos pasos son los siguientes:

- Mostrar que nuestro problema (PPE) está en NP,

- Seleccionar un problema NP-completo conocido, nosotros seleccionamos el problema Subset Sum,
- Construir una reducción del problema SS al problema PPE, y
- Demostrar que la reducción se lleva a cabo en espacio logarítmico.

Una vez mencionados estos pasos, procedemos a realizar la prueba para demostrar que nuestro problema en su versión de decisión es NP-Completo, y por tanto el problema de optimización es NP-Duro.

5.2.1 DESCRIPCIÓN DE AMBOS PROBLEMAS EN SU VERSIÓN DE DECISIÓN

[*Problema Subset Sum (SS)*]
INSTANCIA: Tenemos un conjunto finito O, con tamaño e(o) $\in \mathscr{Z}^+$ para cada o $\in$ O y un número entero positivo b.

PREGUNTA: Hay un subconjunto O' $\subseteq$ O tal que la suma de sus tamaños de los elementos en O' es exactamente b?.

[*Problema de Planificación Educativa (PPE)*]

INSTANCIA: Tenemos un conjunto de Materias, un conjunto de temas, conjunto de subtemas y conjunto de actividades de aprendizaje que forman cada subtema. Cada materia está compuesta de una cierta cantidad de temas, cada tema está compuesto de una cierta cantidad de actividades. Cada actividad tiene un score $r(a)$ y una duracion $d(a)$. Sea U una cota superior y L una cota inferior que representan el score que se debe acumular por subtema.

PREGUNTA: ¿Hay una selección de actividades tal que la suma de los scores de las actividades seleccionadas sea mayor o igual a L y menor o igual a U para

cada subtema y en donde el tiempo total de la suma de la duración de todas las actividades seleccionadas sea menor o igual a c?.

5.2.2 *Theorem*

Problema de Planificación Educativa es NP-completo:

Prueba: Es fácil ver que el problema PPE está em NP, debido a que un algoritmo no determinista necesita solo "adivinar" una selección de actividades y checar en tiempo polinomial que la suma de los score de las actividades seleccionados de cada subtema están entre U y L y que la suma de la duración de todas las actividades seleccionadas es menor o igual que c.

Construimos una instancia del problema PPE en el cual consideramos una sola materia, la cual tiene un único tema y un único subtema. Dicho subtema está compuesto por un conjunto de actividades. Sea A un conjunto de actividades con score $r(a)$ y sea establecida la duración de cada actividad en cero $(d(a) = 0)$. Debido a que sólo se tiene un único subtema no hay relaciones de precedencia entre los subtemas y actividades de otros subtemas. Sea A' un subconjunto de A. La suma de los scores de las actividades del subconjunto A' es igual a $L = U$, donde se asume que esta cantidad es igual a b. 5.1

5.2.3 $\Rightarrow$ *"Si"* EN EL PROBLEMA SS ES UN *"Si"* EN EL PROBLEMA PPE

.

Dada una máquina de Turing TM la cual decide *si* a una instancia del problema SS significa que existe un subconjunto O' tal que la suma de los tamaños de los elementos en O' es igual a b entonces significa que hay un subconjunto A' para el

Instancia problema SS	**Instancia problema PPE**
O Conjunto finito de elementos,	A Conjunto finito de actividades de aprendizaje,
n elementos,	n actividades de aprendizaje,
e_i tamaño de elementos i=1,2,...,n,	r_i score de actividades i=1,2,...,n,
b suma de tamaños de elementos en $O' \in O$	b suma de score de las actividades en $A' \in A$ sujeto a que L=U

Tabla 5.1: Instancias de ambos problemas

cual la suma de los score de los elementos de A' es igual a b.

5.2.4 $\Leftarrow$ "NO" EN EL PROBLEMA PPE ES UN "NO" EN EL PROBLEMA SS

Dada una máquina de Turing TM la cual decide *no* a una instancia del problema PPE significa que no existe un subconjunto A' tal que la suma de sus score sea igual a b entonces significa que no existe un subconjunto O' para el cual la suma de los tamaños de los elementos de O' es igual a "b".

5.2.5 LA REDUCCIÓN DEL PROBLEMA SS AL PROBLEMA PPE ES EN ESPACIO LOGARÍTMICO

Nosotros necesitamos tener en memoria: La cantidad de elementos en $setO$, el número entero b, el elemento actual $o \in O$, el tamaño $e(o)$. Teniendo así $4logn$ espacio, $\mathcal{O}(logn)$.

5.2.6 CONCLUSIONES DE LA COMPLEJIDAD

Presentamos un análisis de complejidad que permite demostrar que el Problema de Planificación Educativa en su versión de decisión es NP-Completo y por lo tanto el problema de optimización es NP-Duro.

CAPÍTULO 6

EXPERIMENTACIÓN

Realizamos varios experimentos con el fin de observar el desempeño de los algoritmos de planificación, el modelo matemático y la conjunción de ambas soluciones. De manera que dividimos en tres secciones las experimentaciones realizadas. En la primer sección veremos los resultados obtenidos con únicamente el modelo de planificación, así como el desempeño de los algoritmos de planificación para obtener soluciones (planes). En la segunda sección veremos el GAP de las soluciones de los algoritmos de planificación vs las soluciones obtenidas por un modelo matemático que desarrollamos. En la tercer sección veremos los resultados de mezclar las soluciones del modelo matemático a los modelos de planificación.

6.1 SECCIÓN 1: EXPERIMENTACIÓN CON LOS MODELOS DE PLANIFICACIÓN

La experimentación se realizó en un servidor HP DL360P G8 con dos Procesadores Intel E5-2630 de 6 núcleos a 2.3Ghz y 16GB de RAM. Se generaron automáticamente modelos de planificación de Inteligencia Artificial por medio de un generador de instancias programado en Ansi C. Posteriormente se utilizaron algoritmos de propósito especial (planificadores) para dar solución a los modelos.

Las características de los modelos de planificación generados son: la función objetivo establecida en los modelos educativos propuestos es minimizar el tiempo total (makespan) de la solución (plan). Es decir, obtener la trayectoria de aprendizaje que,

en el menor tiempo posible, le garantice al estudiante la aprobación de los objetivos de aprendizaje contenidos en el modelo educativo. Se asumió para el proceso de modelación un único estudiante, y que los recursos asociados a las actividades de aprendizaje son ilimitados.

Se seleccionaron dos planificadores para dar solución a los modelos de planificación generados: Son LPG [20] y SGPLAN [23].

SGPLAN soporta todas las propiedades de nuestros modelos educativos, y fue el algoritmo que mejor se comportó en nuestras pruebas. SGPLAN particiona un problema grande de planificación (como el nuestro) en subproblemas. Cada subproblema contiene sus propios subobjetivos que tienen que satisfacerse, pero al particionar el espacio de búsqueda se reduce exponencialmente la complejidad de solución con respecto al problema original [8]. Esta propiedad del planificador es importante porque nuestros modelos están compuestos de materias independientes. Aunque existen relaciones entre actividades de una misma materia, estas son independientes con respecto a actividades de otras materias.

El otro planificador seleccionado es LPG, el cual es un planificador estocástico basado en un algoritmo de *mejor primero*. LPG es recomendado para dominios que tienen cantidades numéricas y duraciones [10], características que tienen nuestros modelos de planificación.

Nosotros utilizamos estos dos planificadores como un medio para validar nuestros modelos, y para obtener soluciones, y de esta manera tratar de analizar los límites que estos puedan tener en resolver nuestros tipos de modelos.

En esta sección se realizaron tres tipos de experimentación. El primer tipo está enfocado a evaluar el tamaño de las instancias (modelos) y la existencia o no de relaciones de precedencia (habilitadores entre las acciones). El segundo tipo de experimentación está orientada a identificar cual elemento del árbol jerárquico dificulta más la resolución de los modelos. El tercer tipo de experimentación se dirige a analizar si la existencia de las métricas de evaluación en el modelo de planificación influyen en la complejidad de resolución de los modelos. Considerándose solo las clases mediana

y grande de los dominios generados en el segundo tipo de experimentación.

En los primeros dos tipos de experimentación está implícita la acumulación de métricas ya que forman parte del modelo para permitir la progresión de actividades. En el último tipo, el modelo es modificado de manera que no tome en cuenta estas métricas, esto para observar el comportamiento del algoritmo de planificación cuando no están presentes.

6.1.1 TIPO 1 DE LA EXPERIMENTACIÓN

La experimentación está enfocada a evaluar dos de las características, que creemos más importantes, si deseamos que la tecnología escale a problemas del mundo real. La primera es la densidad de actividades de aprendizaje. La segunda es la densidad en las relaciones de precedencia entre las actividades de aprendizaje y los nodos de la red educativa.

Se definieron tres clases de problemas basadas en el tamaño de la red. La clase pequeña comprende de 1 a 3 temas, 1 a 3 subtemas y 1 a 3 actividades de aprendizaje. La clase mediana considera de 4 a 5 el número de hijos en la jerarquía de los nodos en la red, y la grande eleva el número de hijos de entre 6 a 7. Se modelaron 1, 2, 3, 4 y 5 materias para cada clase. Es decir, 1 materia de clase pequeña, 1 materia clase mediana, ... , 5 materias clase grande. Esto nos da como resultado, en un árbol jerárquico completo, con una sola materia, un total de 27 actividades de aprendizaje para la clase pequeña, 125 para la clase mediana y 343 para la grande, para el valor máximo de los rangos que comprenden las clases por materia. Teniendo la clase grande con 5 materias, 1715 actividades de aprendizaje.

Se generaron 30 instancias (modelos) por materia, y se consideraron hasta 5 materias por clase. Los modelos se generan con y sin requerimientos de precedencia (relaciones habilitantes) para evaluar las dos características principales de nuestro diseño de experimentos: densidad en la jerarquía de la red y densidad en las relaciones internodos. En total se generaron 900 modelos de prueba siguiendo esta metodología.

Clase	Materias	Temas	Subtemas	Actividades
Pequeña	1	1 a 3	1 a 3	1 a 3
Pequeña	2	1 a 3	1 a 3	1 a 3
Pequeña	3	1 a 3	1 a 3	1 a 3
Pequeña	4	1 a 3	1 a 3	1 a 3
Pequeña	5	1 a 3	1 a 3	1 a 3
Mediana	1	4 a 5	4 a 5	4 a 5
Mediana	2	4 a 5	4 a 5	4 a 5
Mediana	3	4 a 5	4 a 5	4 a 5
Mediana	4	4 a 5	4 a 5	4 a 5
Mediana	5	4 a 5	4 a 5	4 a 5
Grande	1	6 a 7	6 a 7	6 a 7
Grande	2	6 a 7	6 a 7	6 a 7
Grande	3	6 a 7	6 a 7	6 a 7
Grande	4	6 a 7	6 a 7	6 a 7
Grande	5	6 a 7	6 a 7	6 a 7

Tabla 6.1: Clasificación de instancias por clases

Las relaciones de precedencia están al 100 % por cada materia. Esto nos da árboles jerárquicos totalmente interconectados. Es decir, en una materia, todos los temas, (excepto el primero) tiene relación de precedencia a los anteriores. Las actividades de aprendizaje están completamente relacionadas ya sea a las actividades del subtema anterior o actividades anteriores.

En la figura 6.1 se muestran los porcentajes de instancias resueltas por ambos planificadores. Separamos las instancias en dos: con requerimientos, esto es, que tienen relaciones habilitantes en toda la red; y aquellas que no tienen ninguna dependencia entre los nodos de la red.

Podemos observar que a medida que los modelos crecen en número de materias, la

		Con requerimientos		Sin requerimientos	
Clase	Materias	%LPG	%SGPLAN	%LPG	%SGPLAN
Pequeña	1	77	97	77	97
	2	83	93	80	93
	3	83	90	73	90
	4	40	83	40	93
	5	17	70	17	83
Mediana	1	87	90	93	93
	2	73	83	67	87
	3	17	90	13	90
	4	0	3	3	93
	5	0	7	0	77
Grande	1	93	90	97	97
	2	10	67	23	87
	3	0	70	0	80
	4	0	7	0	57
	5	0	3	0	77

Figura 6.1: Concentrado del porcentaje de instancias resueltas por los planificadores SGPLAN y LPG. Modelando con o sin requerimientos

efectividad de los planificadores para resolver dichos modelos disminuye. En SGPLAN la técnica de particionamiento por objetivos que funciona relativamente bien en nuestros modelos sin requerimientos, donde en teoría cada subtema en el modelo pudiera ser resuelto independientemente, sufre dramáticamente cuando se introducen requerimientos a los modelos. Cuando esto sucede, los subproblemas que el modelo introduce ya no son independientes, debido a las interconexiones entre nodos. Podemos observar entonces que SGPLAN se desempeña muy mal en estos casos, sobre todo en instancias grandes, en donde para 5 materias solamente pudo resolver el 3 % de los modelos generados. A pesar de que SGPLAN resuelve un porcentaje mayor de instancias en comparación con LPG. La calidad de los planes generados por SGPLAN no son muy buenos, ya que con el fín de generar un plan se violan algunas restricciones de las acciones.

En el caso de LPG se observa que el tamaño de las instancias afecta en su desempeño ya que en instancias grandes con 3, 4 y 5 materias no pudo resolver nada. Y parece no afectarle tanto si las instancias tienen o no requerimientos, ya que el porcentaje de instancias resueltas sin requerimientos no es significativo.

Por lo anterior se concluye que el tamaño de las instancias afecta el desempeño de los planificadores para resolver una cantidad mayor de instancias, sobre todo en las instancias grandes que son las que más se asemejan a la realidad. Además de lo ante-

rior, el hecho de que los modelos tengan relaciones de precedencia (requerimientos) afecta a la resolución de los mismos en ambos planificadores, pero acentuándose más en el planificador SGPLAN.

6.1.2 TIPO 2 DE LA EXPERIMENTACIÓN

El diseño de este tipo 2 de experimentos está enfocado a analizar que elementos de la red jerárquica, es decir, temas, subtemas y actividades de aprendizaje, afectan la resolución de los problemas. Lo anterior es porque deseamos tener más claridad con respecto a las características que dificultan la resolución de los modelos.

Se definieron cinco clases de problemas basadas en el tamaño de la red, fijando entre cada catergoría la cantidad de elementos que lo conforman. En cada clase lo único que se varia es la cantidad de actividades de aprendizaje, esto para identificar de una manera más eficiente qué elementos pudieran afectar su resolución. La clase pequeña está comprendida de 1 materia, 3 temas y 3 subtemas. La clase mediana considera 3 materias, 5 temas y 5 subtemas y la grande considera 5 materias, 7 temas, 7 subtemas. Tenemos además dos clases que son una combinación de las anteriores, la clase combinada mediana tiene 1 materia, 5 temas y 5 subtemas y la combinada grande 1 materia, 5 temas, 7 subtemas. En todas las clases se varia la cantidad de actividades de aprendizaje (de 1 a 3, de 4 a 6 y de 7 a 9). Esto nos da como resultado, en un árbol jerárquico completo, un total de 81 actividades de aprendizaje para la clase pequeña, 675 para la clase mediana, 2205 para la grande, 225 para la combinada mediana y 315 para la combinada grande como el valor máximo de los rangos que comprenden las clases.

Se generaron 50 instancias (modelos) para cada una de las clases pequeña, mediana y grande variando en cada clase el tamaño de actividades de aprendizaje; además de 100 instancias para cada clase combinada (combinada-mediana, combinada-grande). Los modelos se generan con y sin requerimientos (de precedencia). En total se generaron 2100 modelos de prueba siguiendo esta metodología.

Como en la experimentación anterior el planificador SGPLAN resolvió una mayor

Clase	Materias	Temas	Subtemas	Actividades
Pequeña	1	3	3	1 a 3
Pequeña	1	3	3	4 a 6
Pequeña	1	3	3	7 a 9
Mediana	3	5	5	1 a 3
Mediana	3	5	5	4 a 6
Mediana	3	5	5	7 a 9
Grande	5	7	7	1 a 3
Grande	5	7	7	4 a 6
Grande	5	7	7	7 a 9
CombinadaMediana	1	5	5	1 a 3
CombinadaMediana	1	5	5	4 a 6
CombinadaMediana	1	5	5	7 a 9
CombinadaGrande	1	5	7	1 a 3
CombinadaGrande	1	5	7	4 a 6
CombinadaGrande	1	5	7	7 a 9

Tabla 6.2: Clasificación de instancias del tipo 2

cantidad de instancias, se decidió probar con solo este planificador los experimentos. En la figura 6.2 se muestran los porcentajes de instancias resueltas por SGPLAN.

Podemos observar que la efectividad de SGPLAN disminuye conforme aumenta la cantidad de nodos en las clases. Se observa además que, como en la experimentación anterior, los modelos sin requerimientos tienen un mayor porcentaje de resolución. Y dado el tamaño de la cantidad de elementos en la clase grande de los modelos con requerimientos, SGPLAN no pudo resolver ninguno.

Se puede observar además un patrón en las clases, en donde cuando éstas tienen de 4 a 6 actividades de aprendizaje existe un mayor porcentaje de modelos resueltos. Esto podría deberse a que se tiene una mejor opción de seleccionar las actividades

Clase	Número de Materias	Número de Temas (por materia)	Número de Subtemas (por tema)	Número de actividades (por subtema)	Con requerimientos	Sin requerimientos
Pequeña	1	3	3	1 – 3	90%	98%
Pequeña	1	3	3	4 – 6	100%	100%
Pequeña	1	3	3	7 – 9	96%	100%
Mediana	3	5	5	1 – 3	42%	96%
Mediana	3	5	5	4 – 6	84%	100%
Mediana	3	5	5	7 – 9	62%	100%
Grande	5	7	7	1 – 3	0%	100%
Grande	5	7	7	4 – 6	0%	100%
Grande	5	7	7	7 – 9	0%	0%
Combinada-Mediana	1	5	5	1 – 3	76%	99%
Combinada-Mediana	1	5	5	4 – 6	95%	100%
Combinada-Mediana	1	5	5	7 – 9	83%	100%
Combinada-Grande	1	5	7	1 – 3	66%	99%
Combinada-Grande	1	5	7	4 – 6	94%	100%
Combinada-Grande	1	5	7	7 – 9	83%	100%

Figura 6.2: Concentrado del porcentaje de instancias resueltas por el planificador SGPLAN. Modelando con o sin requerimientos

de aprendizaje que satisfagan las relaciones de precedencia y una mejor acumulación de las métricas. En relación a cuando se tiene menor número de actividades de aprendizaje pareciera que el problema se ha restringido mucho, por lo que podría darse el caso de que el planificador esté combinando todas las actividades para encontrar una potencial solución. El espacio de soluciones puede estar muy pequeño. Caso contrario cuando se tienen muchas actividades, ya que incrementa la combinatoria al considerar un mayor número de actividades.

Podemos concluir que cuando el tamaño de la red jerárquica es grande afecta la resolución de los modelos, aún cuando tienen de 1 a 3 actividades. Tambien que aún siendo de una misma clase, si la cantidad de actividades es muy pequeña y muy grande tiene dificultad al resolver, en comparación cuando las actividades varian de entre 4 y 6.

Creemos que el principal problema con modelos educativos densos, no es nada más el número de actividades de aprendizaje que estos contienen, y las relaciones de precedencia, sino el proceso de propagación de la calidad de las actividades en la red jerárquica del modelo educativo Υ. Teniendo en consideración esto realizamos el tercer tipo experimentación.

6.1.3 TIPO 3 DE LA EXPERIMENTACIÓN

Considerando el comportamiento de los modelos en las etapas anteriores en donde se observa que afecta el tamaño de las intancias y las relaciones de precedencia en la resolución de los modelos, se tenía la intuición de que el proceso de propagación de nuestras métricas vuelve más complejos lo modelos por el incremento en la combinatoria del problema.

Realizamos un ejemplo para poder visualizar la propagación de las métricas. Tenemos una red jerárquica de tareas la cual representa a la materia de Estructura de Datos, ésta tiene sólo 2 temas (conceptos básicos y algoritmos de ordenamiento) los cuáles a su vez tienen 2 subtemas cada uno, con 3 actividades de aprendizaje.

Esto lo podemos ver en la figura 6.3.

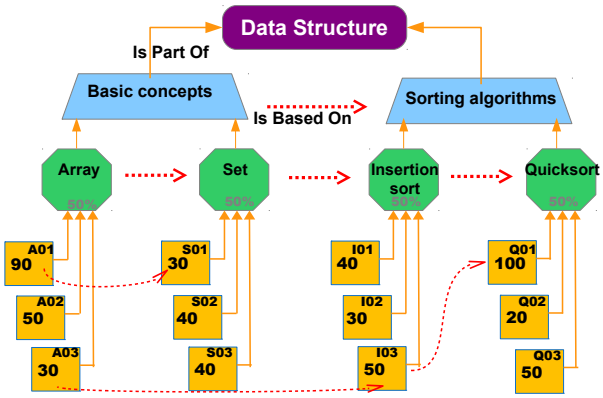


Figura 6.3: Ejemplo de una materia de Estructura de datos con dos temas y dos subtemas para cada tema.

Si el plan para cumplir el objetivo de aprobar esa materia es seleccionar las actividades de aprendizaje que se muestran en la figura 6.4. Podemos observar que al realizar la actividad A01 del subtema “array” que tiene un score de 90 satisfaría el score requerido de ese subtema, asumamos que el score requerido por subtema es 70. Posteriormente en el siguiente subtema podría realizar cualquier actividad, considerando que existen una relación de dependencia entre la actividad S01 con el score de 30 y la actividad anterior A01 con score de 90, así que puede realizar esa

actividad en combinación con cualquier otra de manera que cuando se acumule a nivel de subtema cumpla con el score requerido. Para seleccionar una actividad del subtema “Insertion sort” las únicas combinaciones posibles son las actividades I01 e I02 con el score de 40 y 30 respectivamente, ya que la actividades I03 tiene una relación de precedencia con la actividad A03 que no fué seleccionada. Por último en el subtema “Quicksort” también puede realizar sólo las actividades Q02 y Q03 ya que Q01 tiene una relación de precedencia con la actividad del subtema anterior I03. Con esta combinación de actividades se tiene que el estudiante obtendría un score total de 75.

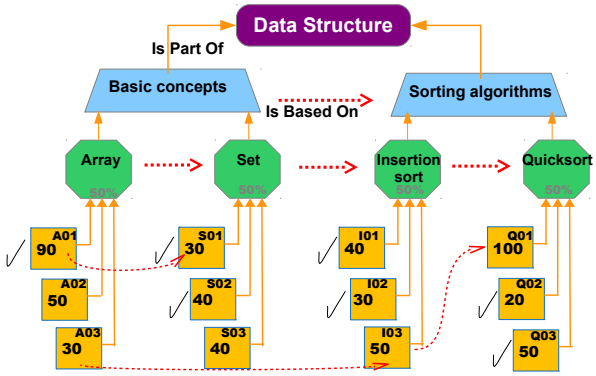


Figura 6.4: Ejemplo de selección de actividades.

Sin embargo se pudieron haber seleccionado las actividades que se muestran en siguiente figura 6.5. En la cual en el primer subtema son seleccionadas las actividades A02 y A03 que no son las de mayor score en el subtema. Posteriormente las únicas combinaciones que se pudieran seleccionar son las actividades S02 y S03 del subtema “Set” ya que la actividad S01 tiene una relación de precedencia con el subrema S01 el cual no fue seleccionado. En el siguiente subtema se seleccionan las actividades I01 y I03, en donde vemos que la actividad I03 tiene una relación con A03, pero esto no nos preocupa puesto que esa actividad ya fué seleccionada. En el último subtema podemos seleccionar cualquier actividad del subtema puesto que si queremos realizar la Q01 que tiene una relación con la I03 del subtema anterior ya está satisfecha en el subtema anterior. Esta combinación de actividades nos da un score total de 87.50, 12.5 puntos más que la combinación anterior. Con esto vemos

que no necesariamente el seleccionar las actividades con mayor score en un inicio nos da un score mayor total.

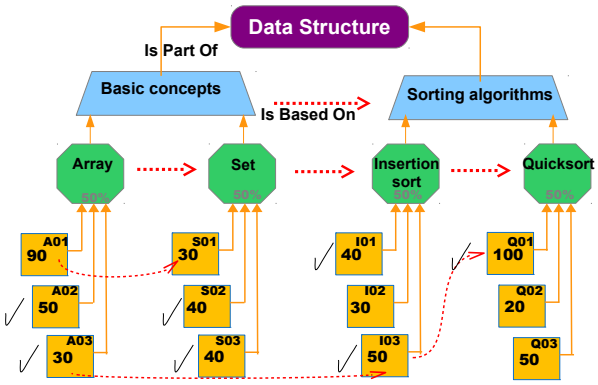


Figura 6.5: Ejemplo de selección de actividades (opción dos).

Esto es un problema para determinar el impacto que tiene un score de una actividad de aprendizaje en el resto de la red por parte de los algoritmos, es decir, cuáles son las actividades de aprendizaje que generan un mayor beneficio (impacto) a la calidad total de la trayectoria de aprendizaje.

Para esta etapa de experimentos se modifican los dominios de planificación de las clases mediana y grande generados en la etapa anterior de manera que no se considere la métrica de acumulación de calidad entre los nodos, ésta se sustituyó por un *predicado* PDDL. Lo anterior es para realizar una comparación entre los dominios y observar si realmente afecta ésto en el desempeño. Solo se consideraron los dominios que tienen requerimientos.

Se desarrolló un programa en lenguaje C el cual toma los dominios y los modifica, quitando la métrica de acumulación de score. Siendo 50 instancias de la clases mediana y grande en cada tamaño de actividades de aprendizaje con requerimientos de precedencia. En total se modificaron 300 modelos de prueba siguiendo esta metodología.

En la figura 6.6 se muestran los porcentajes de instancias resueltas por el planificador SGPLAN.

Podemos observar que aumenta la cantidad de modelos resueltos al ser quitada la

métrica de acumulación de score y que aunque la efectividad de SGPLAN disminuye conforme aumenta la cantidad de nodos en las clases, se tiene un mejor desempeño. En la clase mediana con 1 a 3 actividades de aprendizaje aumentó mas del 50 % los modelos resueltos al ser quitada esta métrica. Y aunque tuvo un muy bajo porcentaje de modelos resueltos en la clase grande se pudo observar que al menos resolvió algunas instancias en comparación con el 0 % de los modelos con la métrica de acumulación de score.

Clases	Número de Materias	Número de Temas (por materia)	Número de Subtemas (por tema)	Número de actividades (por subtema)	Con Requerimientos considerando acumulación de métricas	Con Requerimientos considerando aumulación de métricas
Mediana	3	5	5	1 – 3	42%	100%
Mediana	3	5	5	4 – 6	84%	90%
Mediana	3	5	5	7 – 9	62%	78%
Grande	5	7	7	1 – 3	0%	8%
Grande	5	7	7	4 – 6	0%	4%
Grande	5	7	7	7 – 9	0%	0%

Figura 6.6: Concentrado del porcentaje de instancias resueltas por el planificador SGPLAN. Modelando con restricciones y considerando acumulación de métrica y sin ella.

6.2 SECCIÓN 2: EXPERIMENTACIÓN CON EL MODELO MATEMÁTICO

En esta sección se presenta la experimentación realizada con los modelos de planificación y matemático. De manera que representen el problema de planificación educativa. Desarrollamos un generador en Ansi C que genera 450 instancias del modelo de planificación y del modelo matemático de la plantilla de aprendizaje que presentamos en el capítulo de metodología.

El diseño de las instancias considera tres clases de modelos diferentes, representados en la tabla 6.3. Las clases del modelo varían en el número de tareas de aprendizaje que representan. Para esto establecimos una sola materia y variamos la cantidad de temas, subtemas y actividades de aprendizaje. Consideramos que se modela un solo estudiante, los recursos son ilimitados y la función objetivo es minimizar el tiempo total.

	Materias	Temas	Subtemas	Actividades de Aprendizaje
Pequeña	1	1	2	5
Mediana	1	3	4	5
Grande	1	5	6	5

Tabla 6.3: Clases del modelo de evaluación

Cada clase considera los siguientes parámetros de entrada:

a) El primer parámetro es la densidad de las relaciones habilitantes (enabling) entre las tareas de aprendizaje. En base a la experimentación de la primer sección, estas relaciones aumentan la complejidad de encontrar solución de los algoritmos de planificación y creemos que reducen la calidad de solución. Recordando el ejemplo de solución que presentamos en la sección de metodología (Figura 4.7), la solución encontrada por SGPLAN tenía una utilidad promedio por subtema de 77.5 y el tiempo total de 340. Corrimos la misma instancia con LPG ([20]), que es otro planificador, el cual está basado en búsqueda local y grafos de planificación, y obtuvimos una solución similar (es decir, una utilidad promedio de 75 y un tiempo total de 340). Sin embargo, al ver el ejemplo, existe la siguiente solución $PLAN = \{A_2, A_3, A_5, A_6, A_8, A_9, A_{10}\}$, con una utilidad promedio de 85 y un tiempo total de 325. Incluso para este ejemplo sencillo, los algoritmos de planificación parecen tener problemas para estimar el impacto de las métricas (por ejemplo, la duración, utilidad, etc) en la cadena de condiciones habilitantes en la red de tareas. Consideramos tres valores de densidad 0 % (es decir, no existen relaciones), 20 %, y 80 % del total de numero de tareas.

b) El segundo criterio es el tipo de condiciones habilitantes (enabler). Para esta evaluación consideramos solo condiciones de *Satisfacción objetivo*, uno-uno (1:1) y 2-1 (2:1). Es decir, uno y dos habilitadores para las actividades de aprendizaje. Por lo tanto, de la cantidad total de condiciones habilitantes presentadas en el modelo, variamos los valores entre 20 % y 80 % para cada tipo de condición.

c) El tercer parámetro para el diseño de evaluación es el número de actividades de aprendizaje obligatorias. Las actividades de aprendizaje obligatorias son importantes porque restringen el conjunto soluciones posibles, actuando como restricciones duras durante la síntesis de la solución. Se consideran tres grupos de valores: 0 % (actividades no obligatorias), 10 %, y 20 % del número total de actividades de aprendizaje presentes en los modelos.

d) El último parámetro de entrada para la evaluación es el rango de valor de utilidad asignado a las actividades de aprendizaje. Estos valores son importantes porque nuestros QAFs los utilizan para evaluar el progreso en el plan de estudios. Consideramos dos rangos de valores: *Uniformes (UF)*, and *Aleatorios (RD)*. Los valores UF son calculados primero encontrando la media μ entre el valor de utilidad máximo (por ejemplo, en nuestra evaluación) y el número de actividades de aprendizaje n para un subtema en particular, esto es $\mu = \frac{100}{n}$. Entonces, cada una de las $n-1$ actividades de aprendizaje se les asigna el valor que varía aleatoriamente entre $+-5$ puntos de μ. El valor n^{th} se le asigna a diferencia entre la utilidad máxima y la suma de los valores $n-1$ asignados previamente. Por otro lado, los valores de utilidad RD se construyen toamndo valores aleatorios en el rango $[10, 70]$, generando actividades de aprendizaje con una diferencia de utilidad amplia.

Clases del Modelo	Parámetros de entrada				Número de casos de Prueba
	a) Densidad de enablers	b) Tipos de enablers.	c) Número de LAs obligatorias.	d)Rango de valores de utilidad	
[Pequeña, Mediana, Grande]	0 %	0 %	[0 %, 10 %, 20 %]	[UF, RD]	3 (param. c) x 2 (param. d) = 6
	20 %	1:1 (20 %), 2:1 (80 %)	[0 %, 10 %, 20 %]	[UF, RD]	3x 2 = 6
		1:1 (80 %), 2:1 (20 %)	[0 %, 10 %, 20 %]	[UF, RD]	6
	80 %	1:1 (20 %), 2:1 (80 %)	[0 %, 10 %, 20 %]	[UF, RD]	6
		1:1 (80 %), 2:1 (20 %)	[0 %, 10 %, 20 %]	[UF, RD]	6
Número total de casos de prueba por clase del modelo					30
Número total de casos de prueba					30x 3 (clases) = 90

Tabla 6.4: Configuracion de evaluación

Podemos ver la combinación completa de los parámetros que se utilizan para gene-

rar los casos de prueba para la evaluación en la Tabla 6.4. Hay 30 casos de prueba por clase. Por lo tanto, tenemos 90 casos de prueba en total porque tenemos tres clases del modelo. Generamos cinco diferentes instancias del modelo por caso de prueba, lo que nos da un total de 450 instancias del modelo para ser evaluados para planificación y programación matemática.

Todas las instancias del modelo evaluan la misma función objetivo, la cual considera la minimización del tiempo total de las trayectorias de aprendizaje (planes). En otras palabras, estamos buscando la trayectoria de aprendizaje que, en la mínima cantidad de tiempo, garantice al estudiante el cumplimiento de los objetivos de aprendizaje requeridos por el plan de estudios. Para la evaluación, tenemos sólo un estudiante, el cual decimos que éste cumple con el plan de estudios si acumula una utilidad mayor o igual a 70 por subtema.

Los algoritmos de planificación seleccionados para la evaluación son SGPLAN ([**?**]) y LPG ([20]). SGPLAN fue el ganador de la parte métrica temporal subóptima en el 2004 ([**?**]) y el ganador de la parte determinista de satisfacción de planificación en el 2006 [**?**] en la Competencia de Planificación Internacional (IPC) [1]. La razón por la cual seleccionamos SGPLAN no es sólo porque es compatible con todas las propiedades del dominio PDDL de nuestros modelos de planificación, sino porque es el algoritmos que mostró un mejor rendimiento en las pruebas preliminares. SGPLAN se basa en la idea de particionar los subobjetivos y la resolución de conflictos. Podemos ver que las estructuras del modelo que presentamos se ajustan bien con la descripción del algoritmo ya que los subtemas podrían tratarse como subproblemas independientes.Obviamente, en las redes de tareas con una gran densidad de condiciones habilitantes el algoritmo puede sufir debido a que los subobjetivos no son independientes.

En el caso del planificar LPG fue premiado como mejor planificador automatizado en el 2003 por la IPC y posteriormente en el 2004 como mejor desempeño en dominios que implican *Timed Initial Literals* y en la calidad del plan en la parte

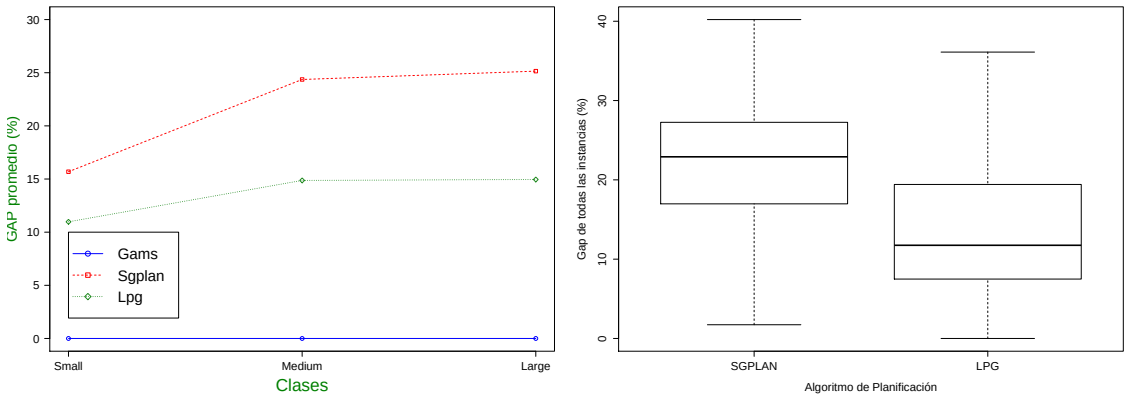
[1] see http://www.icaps-conference.org/index.php/Main/Competitions

de satisfacción de planificación. LPG es un planificador estocástico basado en un algoritmo de *mejor primero*, utilizando una búsqueda local estocástica que le permite encontrar planes con mejor calidad. LPG es recomendado para dominios que tienen cantidades numéricas y duración, como es el caso de nuestro modelo de planificación. Y aunque en experimentaciones previas ha tenido menor cantidad de problemas resueltos, en comparación con SGPLAN, los planes que arroja respetan las métricas y precondiciones establecidas.

Para resolver de manera óptima nuestros modelos de programación matemática se utiliza el Sistema de Modelación Algebraica General (GAMS, por sus siglas en inglés de General Algebraic Modeling System) ([**?**]) versión 24.2.3. GAMS es un sistema de modelado de alto nivel para programación matemática y optimización. Se compone de un compilador de lenguaje y un conjunto de solvers de alto rendimiento integrados. GAMS se adapta para aplicaciones de modelación compleja y a gran escala. El solver utilizado para nuestra evaluación fue GAMS/CPLEX. El optimizador CPLEX está diseñado para reolver problemas grandes y difíciles rápidamente con una intervención mínima del usuario. CPLEX es un software de alto rendimiento para problemas de Programación lineal y Programación lineal entera mixta (LP/MIP). La versión que utilizamos es CPLEX 12.6.0.0.

6.2.1 EVALUACIÓN DE LA CALIDAD DE LA SOLUCIÓN

La resolución de nuestros modelos de programación matemática nos proporcionan las soluciones óptimas en todas las instancias generadas. Con el fin de identificar la diferencia entre las soluciones de los algoritmos de planificación para los modelos de planificación de IA, a las soluciones óptimas del modelo matemático (en cuanto a la selección de las actividades de aprendizaje que minimicen el tiempo total - función objetivo establecida en la experimentación-) calculamos el GAP. En la Figura 6.7 se muestra el GAP encontrado organizado por clases del modelo y algoritmo de planificación (Figura 6.7a) y de todas las instancias del modelo (450) por planificador (Figura 6.7b).



(a) GAP de SGPLAN y LPG por clase

(b) GAP de todas las instancias

Figura 6.7: GAP de ambos planificadores

En la Figura 6.7a es fácil observar que en promedio, SGPLAN tiene un GAP más grande en comparación con LPG. SGPLAN tiene un GAP entre 15 % y 25 %, y LPG entre 10 % y 15 % en promedio por clase. Por clase el GAP de ambos algoritmos de planificación aumenta, teniendo SGPLAN en la clase Pequeña: 15.68 %, Mediana: 24.36 % y Grande: 25.14 % en promedio, por su parte LPG tiene 10.97 % en la clase pequeña, 14.86 % mediana y 14.95 % en promedio para la clase Grande.

Mostramos el GAP promedio para todas las instancias (450) en la Figura 6.7b. Podemos observar que SGPLAN presenta un GAP entre 16.98 % y 27.25 % y LPG entre 7.51 % y 19.41 %. Teniendo en promedio un GAP de 21.66 % para SGPLAN y 13.59 % para LPG.

SGPLAN pudo encontrar la solución óptima en alrededor de 4.89 % y LPG en alrededor de 8.22 % siendo, en ambos casos, instancias de la clase pequeña. Por otro lado, el porcentaje de instancias resueltas tanto por el modelo matemático como por los algoritmos de planificación es alto. Tenemos que GAMS resolvió el 100 % de las instancias, SGPLAN el 97.78 % y LPG 99.11 %; siendo todas las instancias no resueltas de las clases mediana y grande. Recordemos que solo estamos modelando una sola materia.

El tiempo de procesamiento para obtener las soluciones puede ser visto en la Figura

6.8, en donde mostramos el tiempo promedio por clase por tipo de solución. El tiempo que toman para resolver los modelos es despreciable, teniendo en cuenta que todas las soluciones se encuentran en microsegundos. Sin embargo, podemos observar que GAMS toma en promedio más tiempo para resolver las instancias, LPG es el segundo más tardado, considerando que en las intancias grandes el tiempo se dispara de menor a 20 a casi 100 ms. SGPLAN fué el que tomó menos tiempo para dar solución.

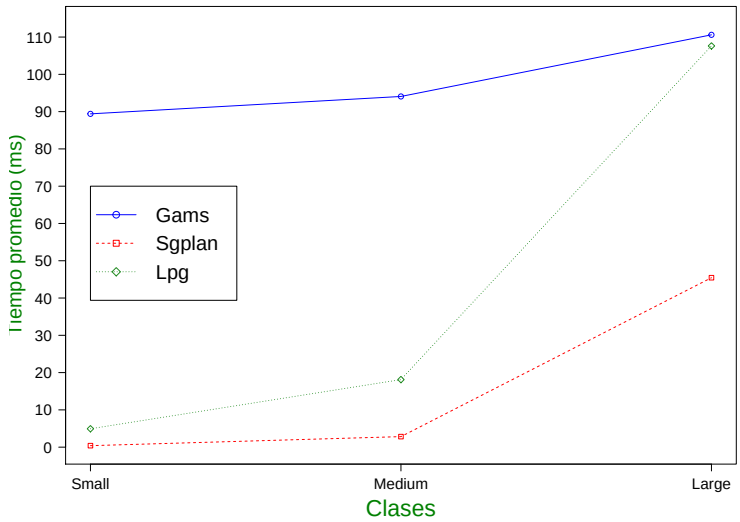


Figura 6.8: Tiempo de solución tomado en microsegundos para resolver modelos matemáticos vs modelos de planificación de IA

Es evidente que el modelo de programación matemática mientras minimiza el tiempo total de la solución (función objetivo), acumula menor cantidad de utilidad para satisfacer la calificación mínima aprobatoria en el modelo (es decir, 70 en nuestra experimentación). Es interesante ver en la Figura 6.9 la utilidad acumulada promedio por subtema obtenida por los algoritmos de planificación y el modelo matemático. Los datos, que están en orden ascendente por clase, muestran que aunque los algoritmos de planificación vienen con un GAP grande, sus trayectorias de aprendizaje garantizan mejores calificaciones para un estudiante. Se observa que SGPLAN da utilidades alrededor de 85, LPG alrededor de 80 y GAMS alrededor de 75.

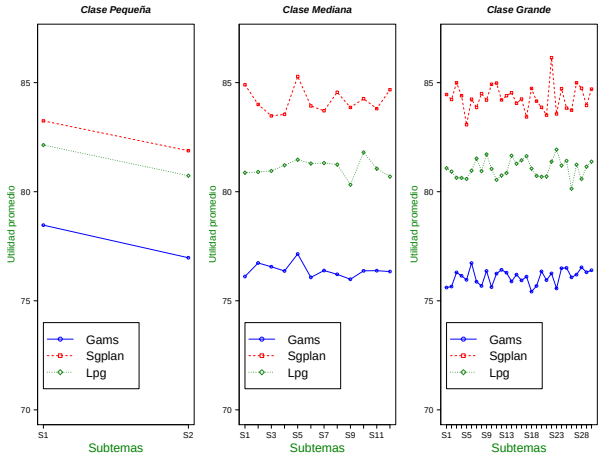


Figura 6.9: Utilidad acumulada promedio por subtema

6.2.1 EVALUACIÓN DE LOS PARÁMETROS

CONDICIONES HABILITANTES

De los parámetros de entrada vemos que las condiciones habilitantes -enablers- (parámetro a) parecen tener un ligero impacto en la calidad de los planes tanto en el modelo matemático, como en los algoritmos de planificación. Recordemos que con calidad del plan nos referimos a la función objetivo, que en este caso de la experimentación es la minimización del tiempo total (la duración del plan), de manera que los valores más bajos son mejores.

Consideramos las instancias con valor cero en el parámetro c (actividades obligatorias), para evitar que éstas interfieran en los otros parámetros, fijamos además el parámetro d en datos uniformes y variamos el nivel de condiciones habilitantes en 0, 20 y 80 %.

En la Figura fig:DT-PU se observa la clase pequeña evaluando los tres tipos de solución. Se observa que en el caso de GAMS a medida que aumentan las condiciones habilitantes crece la función objetivo, aunque en el caso de 20 y 80 % no parece haber mucha diferencia. Para SGPLAN parecería que le beneficia tener el 80 % de condiciones habilitantes, ya que su función objetivo es mejor. Incluso su función objetivo es peor cuando tiene 20 % de condiciones habilitantes, poco mayor

que cuando tiene el 0 %. En el caso de LPG la función objetivo mejora cuando no tiene condiciones habilitantes. Y no hay mucha diferencia cuando hay entre 20 y 80 %. Parece ser que se tienen peores soluciones en promedio cuando hay 20 % de condiciones habilitantes en los tres métodos de solución.

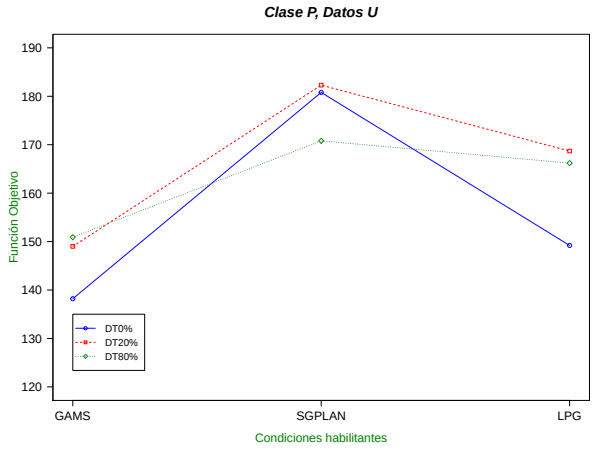


Figura 6.10: Evaluación de las Condiciones habilitantes en la calidad del plan - Clase Pequeña

En la clase mediana (véase Figura fig:DT-MU), los valores de la función objetivo con 20 % de condiciones habilitantes son peores en los tres métodos de solución. Sólo en el caso de GAMS la función objetivo es más pequeña cuando no existen condiciones habilitantes. SGPLAN sigue teniendo mejores resultados con el 80 % de condiciones habilitantes y en el caso de LPG los valores de la función objetivo cuando tiene 80 y 20 % son muy similares (diferencia de 3).

En la figura fig:DT-GU se muestra la clase grande. Podemos observar que otra vez, la función objetivo en promedio es peor cuando se tiene el 20 % de condiciones habilitantes. GAMS mantiene una pequeña diferencia cuando hay 80 y 20 % de condiciones habilitantes (diferencia de 6.5). SGPLAN tiene mejores soluciones con el 80 % de condiciones habilitantes y una diferencia de 5.9 cuando hay 20 y 0 % de condiciones habilitantes. LPG mantiene valores más pequeños cuando no hay condiciones habilitantes, pero no existe mucha diferencia cuando tiene 80 % y sus peores soluciones es cuando tiene 20 % de condiciones habilitantes.

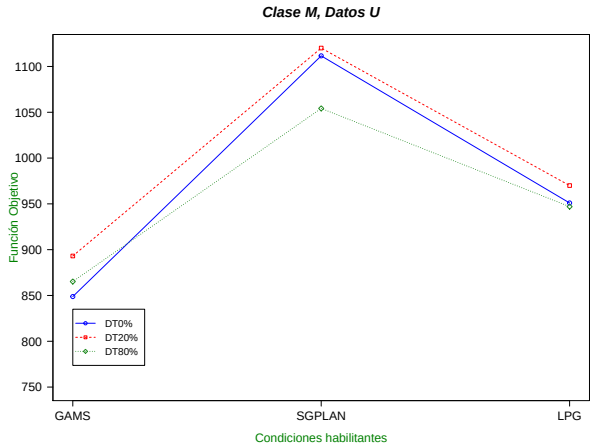


Figura 6.11: Evaluación de las Condiciones habilitantes en la calidad del plan- Clase Mediana

En conclusión podemos ver que las instancias con peores soluciones en los tres métodos de solución es en promedio cuando hay un 20 % de condiciones habilitantes, y los mejores es cuando existen el 80 % de condiciones habilitantes.

TIPOS DE CONDICIONES HABILITANTES 20-80,80-20

En la subsection anterior observamos que cuando hay un 20 % de condiciones habilitantes los valores de la función objetivo eran peores. De ese porcentaje, clasificamos en los tipos de condiciones habilitantes (parámetro b). En donde vemos que podemos tener dependencia a una actividad de aprendizaje o a dos, variando el porcentaje en 20-80 y 80-20 respectivamente.

Consideramos las instancias con valor cero en el parámetro c (actividades obligatorias), para evitar que éstas interfieran en los otros parámetros, fijamos el parámetro d en datos uniformes y en 20 % el total de las condiciones habilitantes, variando entre tener 20 % de DLA1 y 80 % DLA2 (20-80) y 80 % de DLA1 y 20 % de DLA2 (80-20).

En la Figura fig:DLA-ClasesU se muestran las tres clases. Podemos observar que cuando la clase es pequeña los valores de la función objetivo son mejores en la combinación 20-80 para los tres métodos de solución. En la clase mediana la diferencia

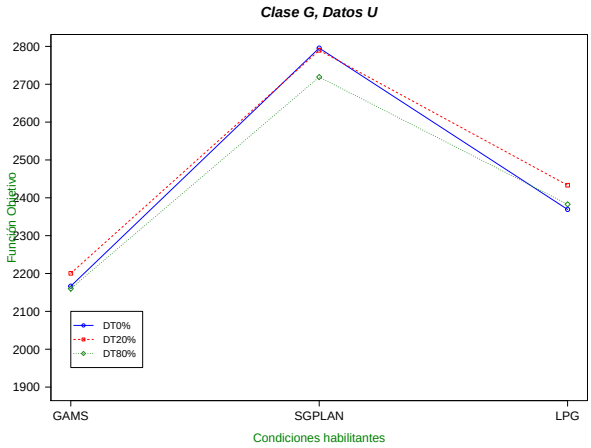


Figura 6.12: Evaluación de las Condiciones habilitantes en la calidad del plan-Clase Grande

entre ambas combinaciones es mínima para todos los métodos y en la clase grande se invierte en comparación con la clase pequeña, en donde los mejores valores son en la combinación 80-20.

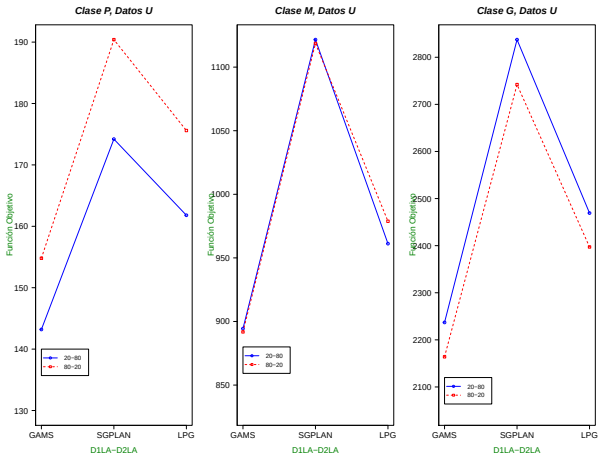


Figura 6.13: Evaluación de tipos de condiciones habilitantes en la calidad del plan

En conclusión podemos ver que, cuando hay un 20 % de condiciones habilitantes, a medida que aumentan las actividades de aprendizaje, esto es, crecen las clases, la combinación 80-20 (80 % dependencia a una actividad de aprendizaje y 20 % a dos actividades de aprendizaje) tiene mejor valor de función objetivo en los tres métodos de solución.

Para el caso en el que el total de las condiciones habilitantes sea de 80 %. En la Figura fig:DLA80-ClasesU se muestran todas las clases. Vemos que en el caso de GAMS y LPG se obtienen mejores soluciones con la combinación 80-20 en las tres clases. SGPLAN tiene igual, mejores soluciones en esta combinación en la clase mediana y grande, sólo en la clase pequeña los valores 20-80 son mejores.

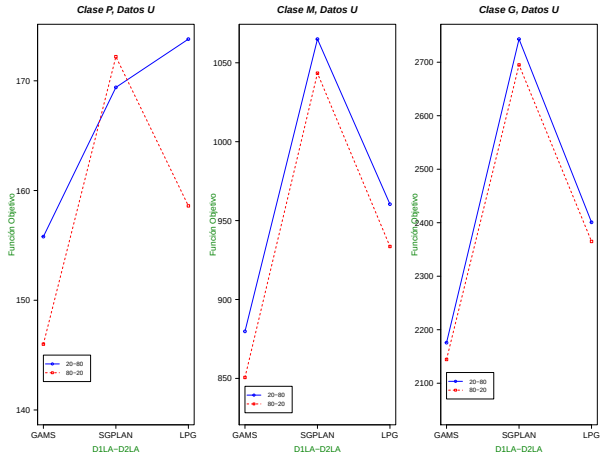


Figura 6.14: Evaluación de tipos de condiciones habilitantes en la calidad del plan

En conclusión podemos ver que en promedio cuando existe un 80 % de condiciones habilitantes los mejores valores de la función objetivo se encuentran en la combinación 80-20.

ACTIVIDADES OBLIGATORIAS

La cantidad de actividades obligatorias (parámetro c) tienen un ligero impacto en la calidad de los planes tanto en el modelo matemático, como en los algoritmos de planificación. Consideramos las instancias con valores cero en los parámetros a y b (condiciones habilitantes), fijamos el parámetro d en datos uniformes y variamos las actividades obligatorias en 0, 10 y 20 %.

En la Figura fig:ActOblig-PU se observa la clase pequeña evaluando los tres tipos de solución. Se observa que en GAMS y LPG es claro que a medida que aumentan las actividades obligatorias crece la función objetivo. En el caso de SGPLAN hay una diferencia de 4.2 entre los valores promedio de la función objetivo con 0 y 10 % de

actividades obligatorias. Sin embargo en todos los métodos de solución la función objetivo es mayor cuando se tienen el 20 % de actividades obligatorias.

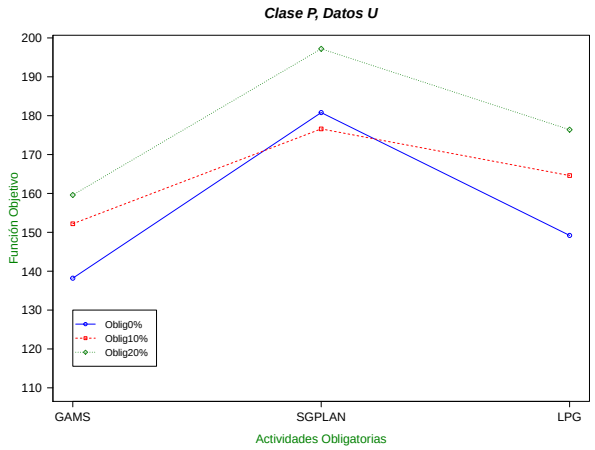


Figura 6.15: Evaluación de las Actividades Obligatorias en la calidad del plan - Clase Pequeña

En la clase mediana (véase Figura fig:ActOblig-MU), aunque siguen siendo mejor los valores de la función objetivo con cero porciento de actividades obligatorias en los tres métodos de solución, esta diferencia no es mayor cuando las actividades obligatorias están entre el 10 y 20 %. GAMS tiene un promedio de 911.4 con 10 % obligatorias y 908.2 con 20 % obligatorias (una diferencia de 3.2). Por su parte LPG tiene en promedio 994.0 con 10 % obligatorias y 997.8 con 20 % obligatorias (3.8 en promedio más cuando pasa de 10 a 20 %). En el caso de SGPLAN cuando hay un 10 % obligatorias el valor de la función objetivo es mayor en comparación con el 20 % (una diferencia de 24.8).

En la figura fig:ActOblig-GU mostramos la clase grande. Vemos que en GAMS y LPG a medida que la cantidad de actividades obligatorias aumenta, se incrementa el valor de la función objetivo. SGPLAN por su parte, en esta clase, parece tener problemas con y sin actividades obligatorias, ya que la diferencia entre tener 0, 10 ó 20 % es mínima, siendo incluso mayor el valor de la función objetivo cuando no tiene actividades obligatorias (sólo una diferencia de 11.6 con 20 % obligatorias). Sin embargo, en comparación con GAMS y LPG sus valores de la función objetivo

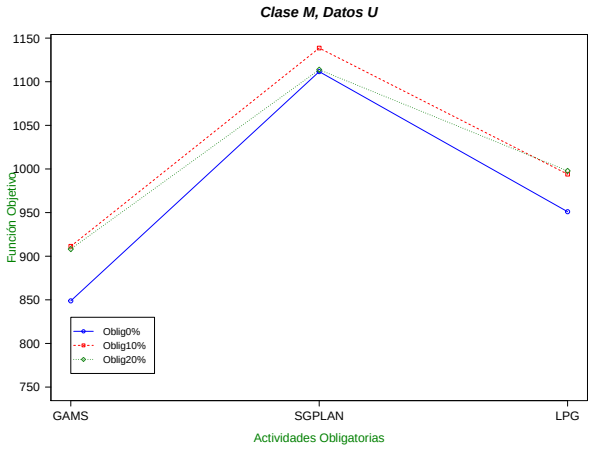


Figura 6.16: Evaluación de las Actividades Obligatorias en la calidad del plan- Clase Mediana

son muy grandes.

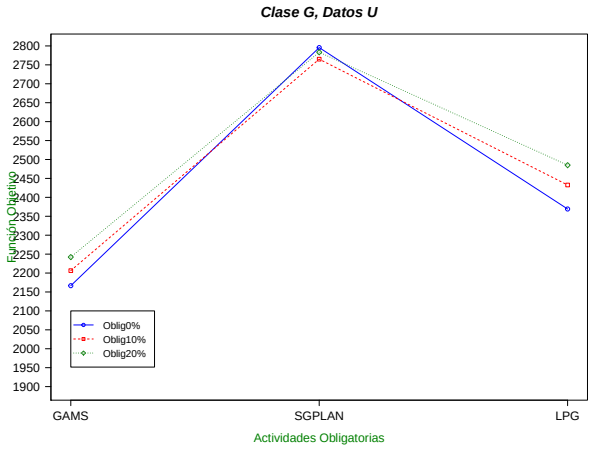


Figura 6.17: Evaluación de las Actividades Obligatorias en la calidad del plan-Clase Grande

Como conclusión decimos que en el caso de GAMS y LPG a medida que aumenta la cantidad de actividades obligatorias la función objetivo crece. Cuando no hay actividades obligatorias GAMS y LPG mantienen mejores soluciones. En el caso de SGPLAN tiene problema con o sin actividades obligatorias, ya que en promedio los valores de su función objetivo, son altos.

DATOS UNIFORMES(UF) Y ALEATORIOS (RD)

En la Figura fig:DatosUV mostramos el impacto en la función objetivo de los valores asignados a las actividades de aprendizaje. Estos valores son: uniformes (UF), es decir que los valores están alrededor del promedio, y aleatorios (RD) que los valores están en un rango de [10,70]. Se consideraron todas las instancias, solo se clasificó en datos UF y RD.

Podemos observar que independientemente de la clase, los valores UF tienen una función objetivo mayor que los datos con valores RD en GAMS y SGPLAN.En el caso de LPG sucede lo contrario, los valores con valores RD tienen un mayor valor en la función objetivo.

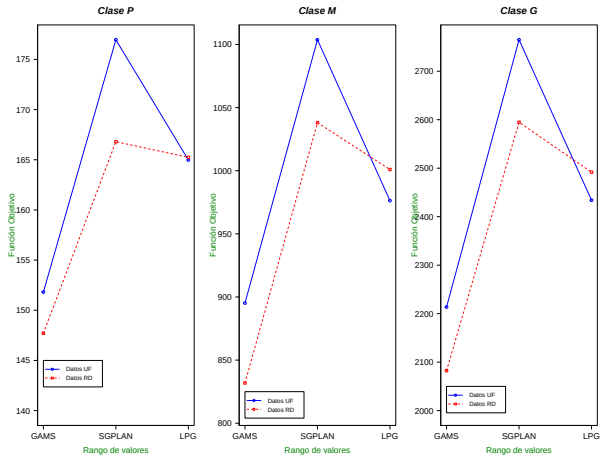


Figura 6.18: Evaluación de los tipos de valores en los datos

Como conclusión decimos que en promedio los datos con valores RD dan mejores valores de la función objetivo.

6.2.2 EXPERIMENTACIÓN CON MÁS DE 1 MATERIA

Anexo a la experimentación anterior, corrimos un conjunto de experimentos con redes de tarea más grandes. Consideramos la clase grande (5 temas por materia, 6 subtemas por tema y 5 actividades por subtema) variando la cantidad de materias a modelar (de 1 a 6 materias). Consideramos sólo tres tipos de instancias: las que

no tienen condiciones habilitantes, ni actividades obligatorias (les llamaremos sin requerimientos); las que tienen el 80 % de condiciones habilitantes con una combinación 20-80 y 10 % de actividades obligatorias (llamaremos grupo 20-80); y las que tienen el 80 % de condiciones habilitantes con una combinación 80-20 y 10 % de actividades obligatorias (grupo 80-20). Se consideraron los valores de las actividades de aprendizaje con datos uniformes (UF).

Los resultados mostraron que en estas redes los algoritmos de planificación resolvieron en promedio 62.22 % para el caso de SGPLAN y 88.88 % en el caso de LPG.

En la Figura 6.19 se muestra el GAP de ambos algoritmos de planificación. LPG mantiene un GAP entre 10.60 y 12.10. Por su parte, SGPLAN está entre 25.22 y 27.50. Por lo que podemos decir que claramente LPG tiene mejores soluciones que SGPLAN.

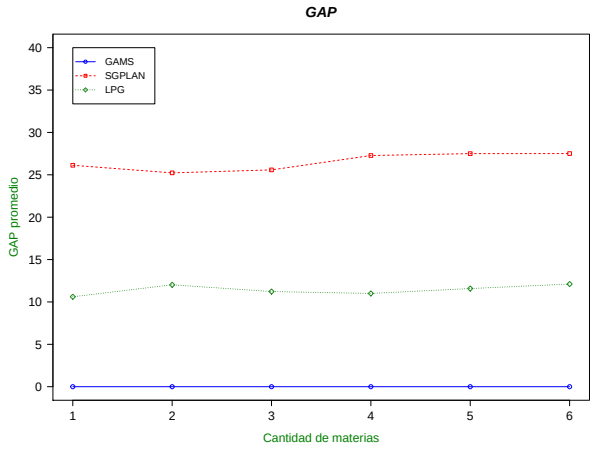


Figura 6.19: Presentación del GAP por algoritmo de planificación

Aunque vemos que LPG tiene un mayor porcentaje de instancias resueltas y mejor GAP, el tiempo de solución es en promedio mayor. En la Figura 6.20 se muestra el tiempo de solución en segundos. Se observa que LPG tiene mayor tiempo de solución a medida que aumenta la cantidad de materias, pero se dispara cuando tiene 5 materias. En el caso de GAMS y SGPLAN el tiempo es mucho menor.

Cómo en la figura anterior no se aprecia la diferencia de tiempos entre GAMS y SGPLAN, se muestra a continuación una gráfica con sólo estos dos métodos de

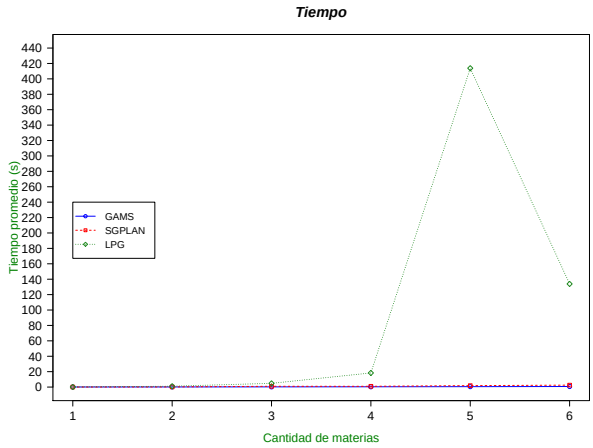


Figura 6.20: Tiempo promedio de solución para los tres métodos de solución

solución. En la Figura 6.21 se puede apreciar que a medida que aumenta la cantidad de materias, aumenta el tiempo de solución (se presenta en microsegundos).

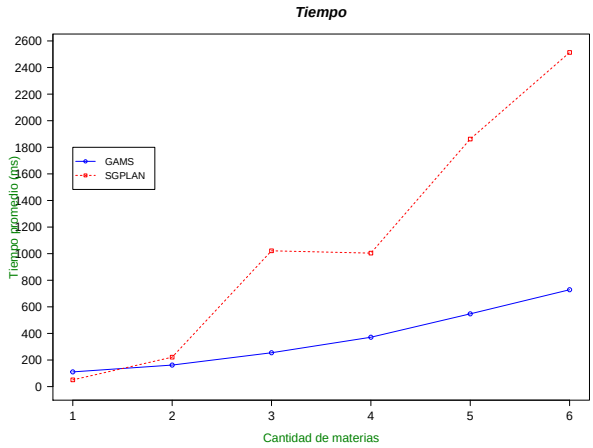


Figura 6.21: Tiempo promedio de solución para GAMS y SGPLAN

Como conclusión decimos que a medida que aumenta la cantidad de materias a modelar el tiempo de solución es mayor, pero en el caso de LPG, éste se dispara. Sin embargo, LPG parece ser más eficiente en cuanto a la cantidad de instancias resueltas y en cuanto al GAP en comparación con SGPLAN. En el caso de GAMS muestra un buen rendimiento, pero recordemos que sólo realiza la selección de actividades que minimicen el tiempo total, y no considera la secuencia de actividades.

6.3 SECCIÓN 3: EXPERIMENTACIÓN CONSIDERANDO SOLUCIONES DEL MODELO MATEMÁTICO COMO ENTRADA EN EL MODELO DE PLANIFICACIÓN.

El modelo matemático nos proporciona soluciones exactas con respecto a la selección de actividades de aprendizaje que minimicen el tiempo total (función objetivo), sin considerar la secuencia (ordenamiento) de las actividades de aprendizaje. El algoritmo de planificación busca realizar ambas cosas, pero arroja GAPs muy grandes. Por lo que consideramos las soluciones del modelo matemático y las incluimos en los modelos de planificación (para que éste no tenga que realizar la selección de aquellas actividades que optimicen la función objetivo) para que los algoritmos de planificación realicen el ordenamiento y obtenga mejores soluciones (En cuanto al GAP).

Para la experimentación consideramos las mismas instancias de la sección anterior (experimentación con el modelo matemático) modificando los modelos de planificación con la solución del modelo matemático. Pedimos resolver con los algoritmos de planificación SGPLAN y LPG.

La modificación en los modelos de planificación es dejar en dichos modelos sólo las actividades de aprendizaje seleccionadas por el modelo matemático. Además de lo anterior, existen dos formas en las que podemos modificar el *objetivo (goal)* en el modelo de planificación: la primera es poner como objetivo: *aprobar la materia*, dejando el cálculo de las métricas igual. La otra forma es poner como objetivo: que aprueba cada una de las actividades de aprendizaje seleccionadas por el modelo matemático, y, quitar el cálculo de la métrica del *maxgrade por subtema*, es decir, aquella que calcula la cantidad de actividades que se pueden realizar por subtema.

OBJETIVO: APROBAR LA MATERIA

En esta sección presentamos los resultados de la experimentación, modificando los modelos de planificación con las soluciones del modelo matemático, estableciendo como *objetivo* que se apruebe la materia. Permitiendo a los algoritmos realizar las acciones necesarias para cumplir con ese objetivo, ésto en teoría debe ser más fácil, porque ahora sólo tiene las actividades que optimizan el tiempo total.

Incluyendo únicamente las actividades de aprendizaje a los modelos de planificación, hacemos que que las soluciones reporten un GAP de cero, es decir, tenemos ahora soluciones exactas en cuanto a la función objetivo.

Con respecto al porcentaje de instancias resueltas tenemos que, SGPLAN y LPG encontrarón solución del 98.44 % de las instancias. Esto no es muy diferente al porcentaje de la experimentación anterior (sin soluciones exactas). Sin embargo, el tiempo de solución es menor en ambos algoritmos de planificación. En la Figura 6.22 se muestran los tiempos de solución de ambos planificadores en ms. Se muestran los resultados antes (leyendas SGPLAN, LPG) y después de mezclar ambas soluciones (leyendas SGPLAN-M, LPG-M). Se puede ver que los tiempos de solución mejoran con respecto a la experimentación anterior.

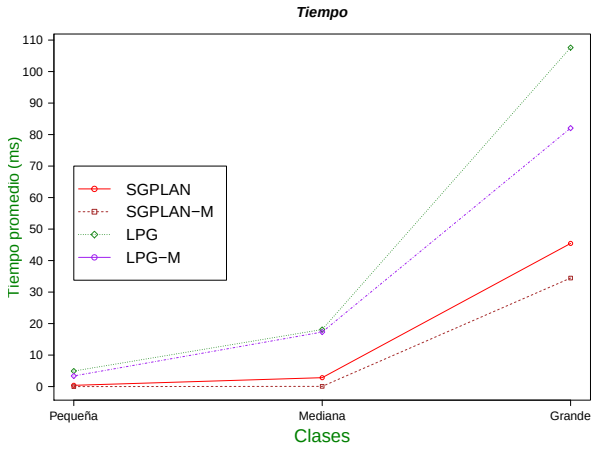


Figura 6.22: Tiempo promedio de solución de los algoritmos de planificación

OBJETIVO: REALIZAR CADA ACTIVIDAD SELECCIONADA

En esta sección presentamos los resultados de la experimentación, modificando los modelos de planificación con las soluciones del modelo matemático, estableciendo como *objetivo* que se realicen cada una de las actividades de aprendizaje seleccionadas por el modelo matemático. Quitando el cálculo de la métrica de *maxgrade por subtema*. Permitiendo a los algoritmos realizar las acciones necesarias para cumplir con ese objetivo, sin realizar cálculos de métricas adicionales.

Al igual que con las soluciones *del objetivo de aprobar la materia*, tenemos ahora soluciones exactas en cuanto a la función objetivo.

Con respecto al porcentaje de instancias resueltas tenemos que, SGPLAN y LPG encontraron solución del 100 % de las instancias. Presentando un mayor porcentaje de instancias resueltas tanto en la experimentacion sin soluciones exactas, como en la experimentación con el objetivo de *aprobar la materia*.

De la misma manera, se observa que el tiempo de solución disminuye en ambos algoritmos de planificación. En la Figura 6.23 se muestran los tiempos de solución de ambos planificadores en ms. Se muestran los resultados antes (leyendas SGPLAN, LPG) y después de mezclar ambas soluciones (leyendas SGPLAN-M, LPG-M). Se puede ver que los tiempos de solución mejoran con respecto a la experimentación anterior, incluso disminuyen con respecto a la experimentación con el objetivo *de aprobar la materia*.

SGPLAN tuvo una reducción de tiempo de solución del 38.70 % en la clase mediana y 49.54 % en la clase grande. Mientras que LPG tuvo una reducción del 33.78 % en la clase mediana y 39.66 % en la clase grande. Esto en comparación con los resultados sin soluciones exactas.

6.3.1 EXPERIMENTACIÓN CON MÁS DE 1 MATERIA

En esta parte de la experimentación consideramos las instancias de la sección anterior (sin soluciones exactas) con más de una materia (de 1 hasta 6 materias). Modifi-

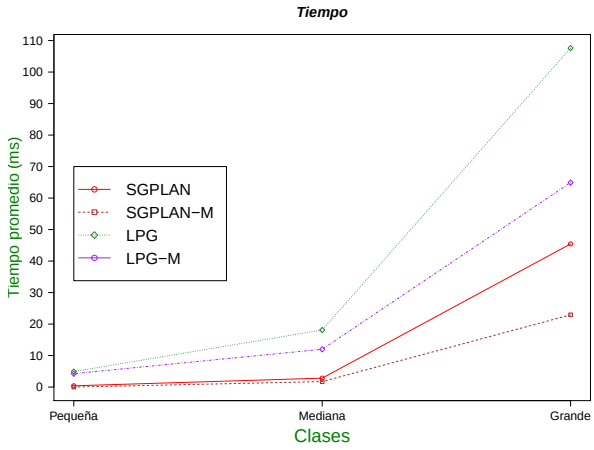


Figura 6.23: Tiempo promedio de solución de los algoritmos de planificación

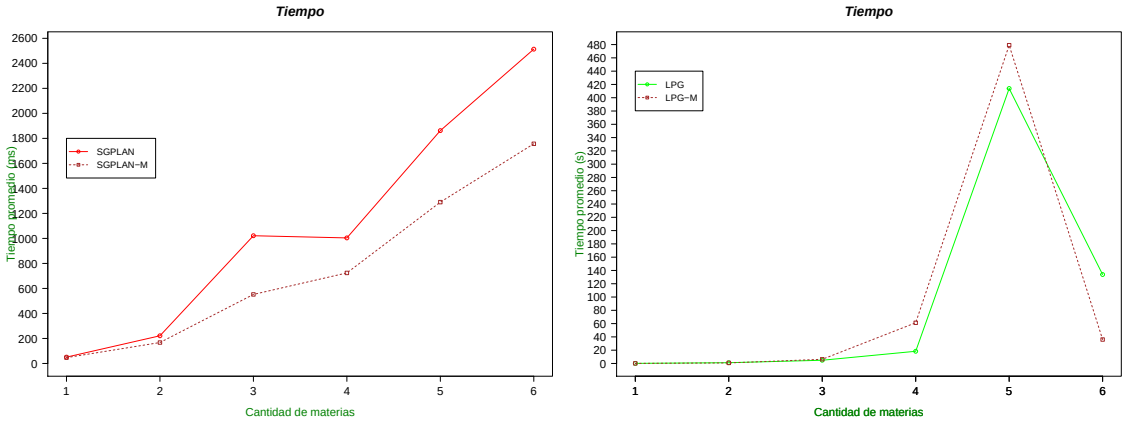
camos los modelos de planificación con las actividades de aprendizaje seleccionadas por el modelo matemático.

Al igual que en la sección anterior, realizamos modificaciones en el modelo de planificación considerando dos tipos de objetivos. En el primero es establecer como objetivo: *aprobar la materia*, dejando el cálculo de las métricas igual. La otra forma es poner como objetivo: que aprueba cada una de las actividades de aprendizaje seleccionadas por el modelo matemático, y, quitar el cálculo de la métrica del *max-grade por subtema*, es decir, aquella que calcula la cantidad de actividades que se pueden realizar por subtema.

OBJETIVO: APROBAR LA MATERIA

Con respecto al porcentaje de instancias resueltas SGPLAN pasó de 62.22 % (instancias sin soluciones exactas) a 64.44 %, por el contrario LPG pasó de 88.88 % a 82.22 % (una diferencia de 6 instancias no resueltas). Y aunque LPG ha tenido mejores resultados en general, vemos que en esta parte de la experimentación con más de una materia, el porcentaje de instancias resueltas disminuye con la inclusión de las soluciones del modelo matemático.

En la figura 6.24 se muestran los tiempos de solución en ms. En el caso de SGPLAN (Figura 6.24a el tiempo de solución disminuye cuando modificamos los modelos de



(a) Tiempo de solución de SGPLAN (ms)

(b) Tiempo de solución de LPG (s)

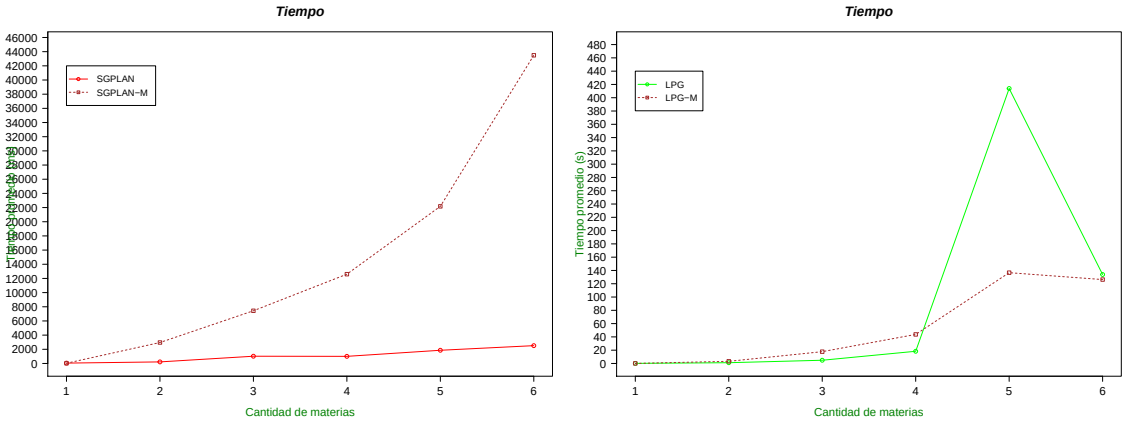
Figura 6.24: Tiempo promedio de ambos planificadores

planificación con la información del modelo matemático, teniendo un 27.84 % en promedio de disminución del tiempo. En el caso de LPG, no hay una condición clara al respecto, ya que el tiempo es muy similar en las primeras tres materias, crece en la cuarta y quinta materia y disminuye en la sexta materias.

OBJETIVO: REALIZAR CADA ACTIVIDAD SELECCIONADA

En esta parte de la experimentación, con respecto al porcentaje de instancias resueltas, SGPLAN pasó de 62.22 % a 64.44 % y LPG quedó igual en 88.89 %. Y aunque LPG ha tenido mejores resultados, vemos que con la inclusión de las soluciones del modelo matemático, no aumenta la cantidad de instancias resueltas. Sin embargo al menos se mantiene igual y no disminuye, en comparación con el objetivo de *aprobar la materia*.

En la figura 6.25 se muestran los tiempos de solución. En el caso de SGPLAN (Figura 6.25a el tiempo de solución aumenta cuando modificamos los modelos de planificación con la información del modelo matemático y el objetivo de realizar cada una de las actividades seleccionadas. A medida que aumenta la cantidad de materias a modelar, aumenta el tiempo de solución. Y aunque se esperaría que al tener únicamente las actividades de aprendizaje que seleccionó el modelo matemático



(a) Tiempo de solución de SGPLAN (ms) (b) Tiempo de solución de LPG (s)

Figura 6.25: Tiempo promedio de ambos planificadores

tardara menos tiempo, no fue así. Esto puede deberse a la naturaleza del algoritmo de solución, que al dividir sus objetivos, éstos no empaten sobre todo en actividades que tienen relación de precedencia.

En el caso de LPG (6.25b), parece que los tiempos son más estables en comparación con los resultados de la experimentación en donde no se consideraban las soluciones del modelo matemático. Y aunque el tiempo es ligeramente mayor en dos, tres y cuatro materias, el resto está por debajo.

Como conclusión podemos decir, que a diferencia de cuando sólo tenemos una materia, los tiempos en promedio de solución aumentan cuando se realiza la inclusión de las soluciones del modelo matemático a los modelos de planificación. Sin embargo, esto hace que tengamos soluciones óptimas en los planes arrojados por ambos algoritmos de planificación.

APÉNDICE A

MODELO EN PDDL

A.1 MODELO PDDL DE 1 MATERIA DE CLASE PEQUEÑA

A.1.1 DOMINIO

(define (domain degree)

(:requirements :durative-actions :typing :fluents :equality)

(:types student resource - object

subject Theme subtheme LA - LO)

(:constants

Materia1 - subject

Tema1 - Theme

Subtema1

Subtema2 - subtheme

LA1

LA2

LA3

LA4

LA5

LA6

LA7

LA8

LA9

LA10 - LA

rec0

rec1

rec2

rec3 - resource

)

(:predicates

(available-subject ?subj - subject ?s - student)

(free ?s - student)

(pass-degree ?subj - subject ?s - student)

(enrollment ?s - student ?subj - subject)

(done-Theme ?t - Theme ?subj -subject ?s - student)

(done-subject-LA ?subj - subject ?s - student)

(not-done-LA ?oa - LA ?subj - subject ?s - student)

(not-approved ?subj - subject ?s - student)

(isPartOfSubtheme ?oa - LA ?subt - subtheme)

(isPartOfTheme ?subt - subtheme ?t - Theme)

(isPartOfSubject ?t - Theme ?subj - subject)

(KindResourceLO ?oa - LA ?eq - resource)

(not-has-reqs ?oa - LA)

(has-reqs ?oa - LA ?req - LO)

```
(has-multiple-reqs ?oa - LA ?req - LO)
(done ?oa - LA)
)

(:functions
(credits-subject ?subj - subject)
(total-credits-subject-gain ?s - student)
(available-credits ?s - student)
(score ?req - LO ?s - student)
(quantity-resource ?eq - resource)
(valueLA ?oa - LA)
(mingrade ?subj - subject)
(DurationLA ?oa - LA)
(maxgrade-subtheme ?subt - subtheme)
(amount-in-subtheme ?oa - LA)
)

(:durative-action enroll-subject_Materia1
:parameters (?s - student)
:duration (= ?duration 0)
:condition (and
(at start (available-subject Materia1 ?s))
(at start (not-approved Materia1 ?s))
(at start (<(credits-subject Materia1)(available-credits ?s)))
)
:effect (and
(at end (enrollment ?s Materia1))
(at end (decrease (available-credits ?s)(credits-subject Materia1)))
(at end (not (available-subject Materia1 ?s)))
)
```

```
)

(:durative-action CHOOSE-LA-nothasreqs
:parameters (?s - student ?oa - LA ?subt - subtheme ?t - Theme ?subj - subject ?eq
- resource)
:duration (= ?duration (DurationLA ?oa))
:condition (and
(at start (free ?s))
(at start (enrollment ?s ?subj))
(at start (not-done-LA ?oa ?subj ?s))
(at start (isPartOfSubtheme ?oa ?subt))
(at start (isPartOfTheme ?subt ?t))
(at start (isPartOfSubject ?t ?subj))
(at start (KindResourceLO ?oa ?eq))
(at start (> (quantity-resource ?eq) 0))
(at start (not-has-reqs ?oa))
(at start (> (maxgrade-subtheme ?subt)(valueLA ?oa)))
)
:effect (and
(at start (not(free ?s)))
(at start (decrease (quantity-resource ?eq) 1))
(at end (increase (quantity-resource ?eq) 1))
(at end (not (not-done-LA ?oa ?subj ?s)))
(at end (increase (score ?subt ?s) (valueLA ?oa)))
(at end (free ?s))
(at end (decrease (maxgrade-subtheme ?subt)(valueLA ?oa)))
(at end (done ?oa))
)
)
```

```
(:durative-action CHOOSE-LA-hasreqsSubtheme
:parameters (?s - student ?oa - LA ?subt - subtheme ?t - Theme ?subj - subject ?eq
- resource ?req - LO)
:duration (= ?duration (DurationLA ?oa))
:condition (and
(at start (free ?s))
(at start (enrollment ?s ?subj))
(at start (not-done-LA ?oa ?subj ?s))
(at start (isPartOfSubtheme ?oa ?subt))
(at start (isPartOfTheme ?subt ?t))
(at start (isPartOfSubject ?t ?subj))
(at start (KindResourceLO ?oa ?eq))
(at start (> (quantity-resource ?eq) 0))
(at start (has-reqs ?oa ?req))
(at start (> (score ?req ?s) (amount-in-subtheme ?oa)))
(at start (> (maxgrade-subtheme ?subt)(valueLA ?oa)))
)
:effect (and
(at start (not(free ?s)))
(at start (decrease (quantity-resource ?eq) 1))
(at end (increase (quantity-resource ?eq) 1))
(at end (not (not-done-LA ?oa ?subj ?s)))
(at end (increase (score ?subt ?s) (valueLA ?oa)))
(at end (free ?s))
(at end (decrease (maxgrade-subtheme ?subt)(valueLA ?oa)))
(at end (done ?oa))
)
)


(:durative-action CHOOSE-LA-hasreqsLA
```

```
:parameters (?s - student ?oa - LA ?subt - subtheme ?t - Theme ?subj - subject ?eq
- resource ?req - LA)
:duration (= ?duration (DurationLA ?oa))
:condition (and
(at start (free ?s))
(at start (enrollment ?s ?subj))
(at start (not-done-LA ?oa ?subj ?s))
(at start (isPartOfSubtheme ?oa ?subt))
(at start (isPartOfTheme ?subt ?t))
(at start (isPartOfSubject ?t ?subj))
(at start (KindResourceLO ?oa ?eq))
(at start (> (quantity-resource ?eq) 0))
(at start (has-reqs ?oa ?req))
(at start (done ?req))
(at start (> (maxgrade-subtheme ?subt)(valueLA ?oa)))
)
:effect (and
(at start (not(free ?s)))
(at start (decrease (quantity-resource ?eq) 1))
(at end (increase (quantity-resource ?eq) 1))
(at end (not (not-done-LA ?oa ?subj ?s)))
(at end (increase (score ?subt ?s) (valueLA ?oa)))
(at end (free ?s))
(at end (decrease (maxgrade-subtheme ?subt)(valueLA ?oa)))
(at end (done ?oa))
)
)

(:durative-action CHOOSE-LA-hasreqs-multipleLA2
:parameters (?s - student ?oa - LA ?subt - subtheme ?t - Theme ?subj - subject ?eq
```

```
- resource ?req1 - LA ?req2 - LA)
:duration (= ?duration (DurationLA ?oa))
:condition (and
(at start (not(= ?req1 ?req2)))
(at start (free ?s))
(at start (enrollment ?s ?subj))
(at start (not-done-LA ?oa ?subj ?s))
(at start (isPartOfSubtheme ?oa ?subt))
(at start (isPartOfTheme ?subt ?t))
(at start (isPartOfSubject ?t ?subj))
(at start (KindResourceLO ?oa ?eq))
(at start (> (quantity-resource ?eq) 0))
(at start (done ?req1))
(at start (done ?req2))
(at start (has-multiple-reqs ?oa ?req1))
(at start (has-multiple-reqs ?oa ?req2))
(at start (> (maxgrade-subtheme ?subt)(valueLA ?oa)))
)
:effect (and
(at start (not(free ?s)))
(at start (decrease (quantity-resource ?eq) 1))
(at end (increase (quantity-resource ?eq) 1))
(at end (not (not-done-LA ?oa ?subj ?s)))
(at end (increase (score ?subt ?s) (valueLA ?oa)))
(at end (free ?s))
(at end (decrease (maxgrade-subtheme ?subt)(valueLA ?oa)))
(at end (done ?oa))
)
)
```

```
(:durative-action PASS-Theme-Tema1_Materia1
:parameters (?s - student)
:duration (= ?duration 0)
:condition (and
(at start (enrollment ?s Materia1))
(at start (>= (score Subtema1 ?s)(mingrade Materia1)))
(at start (>= (score Subtema2 ?s)(mingrade Materia1)))
)
:effect (and
(at end (done-Theme Tema1 Materia1 ?s))
)
)

(:durative-action PASS-Materia1
:parameters (?s - student)
:duration (= ?duration 0)
:condition (and
(at start (enrollment ?s Materia1))
(at start (done-Theme Tema1 Materia1 ?s))
)
:effect
(at end (done-subject-LA Materia1 ?s))
)

(:durative-action take-subject-pass
:parameters (?s - student ?subj - subject)
:duration (= ?duration 0)
:condition (and
(at start (enrollment ?s ?subj))
(at start (done-subject-LA ?subj ?s))
```

```
)
:effect (and
(at end (not (not-approved ?subj ?s)))
(at end (increase (total-credits-subject-gain ?s) (credits-subject ?subj)))
(at end (not (available-subject ?subj ?s)))
(at end (pass-degree ?subj ?s))
)
)
)
```

A.1.2 PROBLEMA

```
(define (problem degree-example)
(:domain degree)
(:objects
student1 - student
)

(:init
(free student1)
(= (total-credits-subject-gain student1) 0)
(= (available-credits student1) 48)

(available-subject Materia1 student1)
(= (credits-subject Materia1) 8)
(= (mingrade Materia1) 70)
(not-approved Materia1 student1)
(not-done-LA LA1 Materia1 student1)
(not-done-LA LA2 Materia1 student1)
```

(not-done-LA LA3 Materia1 student1)

(not-done-LA LA4 Materia1 student1)

(not-done-LA LA5 Materia1 student1)

(= (score Subtema1 student1) 0)

(not-done-LA LA6 Materia1 student1)

(not-done-LA LA7 Materia1 student1)

(not-done-LA LA8 Materia1 student1)

(not-done-LA LA9 Materia1 student1)

(not-done-LA LA10 Materia1 student1)

(= (score Subtema2 student1) 0)

(= (quantity-resource rec0)100000)

(= (quantity-resource rec1) 20)

(= (quantity-resource rec2) 30)

(= (quantity-resource rec3) 40)

(= (valueLA LA1) 8)

(= (valueLA LA2) 25)

(= (valueLA LA3) 38)

(= (valueLA LA4) 26)

(= (valueLA LA5) 57)

(= (valueLA LA6) 9)

(= (valueLA LA7) 34)

(= (valueLA LA8) 28)

(= (valueLA LA9) 39)

(= (valueLA LA10) 31)

(= (DurationLA LA1) 19)

(= (DurationLA LA2) 24)

(= (DurationLA LA3) 22)

(= (DurationLA LA4) 36)

(= (DurationLA LA5) 54)

(= (DurationLA LA6) 17)

(= (DurationLA LA7) 30)

(= (DurationLA LA8) 30)

(= (DurationLA LA9) 39)

(= (DurationLA LA10) 29)

(isPartOfSubtheme LA1 Subtema1)

(isPartOfSubtheme LA2 Subtema1)

(isPartOfSubtheme LA3 Subtema1)

(isPartOfSubtheme LA4 Subtema1)

(isPartOfSubtheme LA5 Subtema1)

(isPartOfSubtheme LA6 Subtema2)

(isPartOfSubtheme LA7 Subtema2)

(isPartOfSubtheme LA8 Subtema2)

(isPartOfSubtheme LA9 Subtema2)

(isPartOfSubtheme LA10 Subtema2)

(isPartOfTheme Subtema1 Tema1)

(isPartOfTheme Subtema2 Tema1)

(isPartOfSubject Tema1 Materia1)

(KindResourceLO LA1 rec1)

(KindResourceLO LA2 rec2)

(KindResourceLO LA3 rec1)

(KindResourceLO LA4 rec3)

```
(KindResourceLO LA5 rec3)
(KindResourceLO LA6 rec1)
(KindResourceLO LA7 rec1)
(KindResourceLO LA8 rec2)
(KindResourceLO LA9 rec1)
(KindResourceLO LA10 rec1)

(not-has-reqs LA1)
(not-has-reqs LA2)
(not-has-reqs LA3)
(not-has-reqs LA4)
(not-has-reqs LA5)
(not-has-reqs LA6)
(not-has-reqs LA7)
(not-has-reqs LA8)
(not-has-reqs LA9)
(not-has-reqs LA10)

(= (maxgrade-subtheme Subtema1) 100)
(= (maxgrade-subtheme Subtema2) 100)

)

;;objetivo: termine todas las materias
(:goal (and
(pass-degree Materia1 student1)
)
)
```

;;metrica a optimizar

(:metric minimize (total-time))

)

A.1.3 PLAN DADO POR EL ALGORITMO DE PLANIFICACIÓN LPG

; Version LPG-td-1.0

; Seed 119933079

; Command line: ./lpg -o domain1m-1t-2s-5a-0000V-1.pddl -f problem1m-1t-2s-5a-0000V-1.pddl -n 1 -cputime 1800

; Problem problem1m-1t-2s-5a-0000V-1.pddl

; Time 0.00

; Search time 0.00

; Parsing time 0.00

; Mutex time 0.00

; MakeSpan 151.00

0.0003: (ENROLL-SUBJECT_MATERIA1 STUDENT1) [0.0000]

0.0005: (CHOOSE-LA-NOTHASREQS STUDENT1 LA3 SUBTEMA1 TEMA1 MATERIA1 REC1) [22.0000]

22.0008: (CHOOSE-LA-NOTHASREQS STUDENT1 LA2 SUBTEMA1 TEMA1 MATERIA1 REC2) [24.0000]

46.0010: (CHOOSE-LA-NOTHASREQS STUDENT1 LA1 SUBTEMA1 TEMA1 MATERIA1 REC1) [19.0000] ;; InputAct

65.0012: (CHOOSE-LA-NOTHASREQS STUDENT1 LA9 SUBTEMA2 TEMA1 MATERIA1 REC1) [39.0000] ;; InputAct

104.0015: (CHOOSE-LA-NOTHASREQS STUDENT1 LA8 SUBTEMA2 TEMA1 MATERIA1 REC2) [30.0000]

134.0017: (CHOOSE-LA-NOTHASREQS STUDENT1 LA6 SUBTEMA2 TE-

MA1 MATERIA1 REC1) [17.0000]

151.0020: (PASS-THEME-TEMA1_MATERIA1 STUDENT1) [0.0000]

151.0023: (PASS-MATERIA1 STUDENT1) [0.0000]

151.0025: (TAKE-SUBJECT-PASS STUDENT1 MATERIA1) [0.0000]

A.1.4 PLAN DADO POR EL ALGORITMO DE PLANIFICACIÓN SGPLAN

; Time 0.02

; ParsingTime 0.02

; NrActions 8

; MakeSpan

; MetricValue 145.008

; PlanningTechnique Modified-FF(enforced hill-climbing search) as the subplanner

0.001: (ENROLL-SUBJECT_MATERIA1 STUDENT1) [0.0000]

0.002: (CHOOSE-LA-NOTHASREQS STUDENT1 LA9 SUBTEMA2 TEMA1 MATERIA1 REC1) [39.0000]

39.003: (CHOOSE-LA-NOTHASREQS STUDENT1 LA7 SUBTEMA2 TEMA1 MATERIA1 REC1) [30.0000]

69.004: (CHOOSE-LA-NOTHASREQS STUDENT1 LA5 SUBTEMA1 TEMA1 MATERIA1 REC3) [54.0000]

123.005: (CHOOSE-LA-NOTHASREQS STUDENT1 LA3 SUBTEMA1 TEMA1 MATERIA1 REC1) [22.0000]

145.006: (PASS-THEME-TEMA1_MATERIA1 STUDENT1) [0.0000]

145.007: (PASS-MATERIA1 STUDENT1) [0.0000]

145.008: (TAKE-SUBJECT-PASS STUDENT1 MATERIA1) [0.0000]

A.1.5 PROBLEMA, VERSIÓN CON RELACIONES HABILITANTES Y ACTIVIDADES OBLIGATORIAS

```
(define (problem degree-example)
(:domain degree)
(:objects
student1 - student
)

(:init
(free student1)
(= (total-credits-subject-gain student1) 0)
(= (available-credits student1) 48)

(available-subject Materia1 student1)
(= (credits-subject Materia1) 7)
(= (mingrade Materia1) 70)
(not-approved Materia1 student1)
(not-done-LA LA1 Materia1 student1)
(not-done-LA LA2 Materia1 student1)
(not-done-LA LA3 Materia1 student1)
(not-done-LA LA4 Materia1 student1)
(not-done-LA LA5 Materia1 student1)
(= (score Subtema1 student1) 0)
(not-done-LA LA6 Materia1 student1)
(not-done-LA LA7 Materia1 student1)
(not-done-LA LA8 Materia1 student1)
(not-done-LA LA9 Materia1 student1)
(not-done-LA LA10 Materia1 student1)
(= (score Subtema2 student1) 0)
```

(= (quantity-resource rec0)100000)

(= (quantity-resource rec1) 20)

(= (quantity-resource rec2) 30)

(= (quantity-resource rec3) 40)

(= (valueLA LA1) 8)

(= (valueLA LA2) 34)

(= (valueLA LA3) 30)

(= (valueLA LA4) 38)

(= (valueLA LA5) 45)

(= (valueLA LA6) 24)

(= (valueLA LA7) 28)

(= (valueLA LA8) 19)

(= (valueLA LA9) 29)

(= (valueLA LA10) 64)

(= (DurationLA LA1) 11)

(= (DurationLA LA2) 28)

(= (DurationLA LA3) 37)

(= (DurationLA LA4) 31)

(= (DurationLA LA5) 51)

(= (DurationLA LA6) 22)

(= (DurationLA LA7) 35)

(= (DurationLA LA8) 9)

(= (DurationLA LA9) 30)

(= (DurationLA LA10) 70)

(isPartOfSubtheme LA1 Subtema1)

(isPartOfSubtheme LA2 Subtema1)

(isPartOfSubtheme LA3 Subtema1)

(isPartOfSubtheme LA4 Subtema1)

(isPartOfSubtheme LA5 Subtema1)

(isPartOfSubtheme LA6 Subtema2)

(isPartOfSubtheme LA7 Subtema2)

(isPartOfSubtheme LA8 Subtema2)

(isPartOfSubtheme LA9 Subtema2)

(isPartOfSubtheme LA10 Subtema2)

(isPartOfTheme Subtema1 Tema1)

(isPartOfTheme Subtema2 Tema1)

(isPartOfSubject Tema1 Materia1)

(KindResourceLO LA1 rec3)

(KindResourceLO LA2 rec1)

(KindResourceLO LA3 rec2)

(KindResourceLO LA4 rec2)

(KindResourceLO LA5 rec3)

(KindResourceLO LA6 rec3)

(KindResourceLO LA7 rec1)

(KindResourceLO LA8 rec1)

(KindResourceLO LA9 rec3)

(KindResourceLO LA10 rec1)

(not-has-reqs LA1)

(not-has-reqs LA2)

(not-has-reqs LA3)

```
(not-has-reqs LA4)
(not-has-reqs LA5)

(has-reqs LA6 LA1)
(has-reqs LA7 LA4)
(has-reqs LA8 LA3)
(has-reqs LA9 LA5)

(has-multiple-reqs LA10 LA1)
(has-multiple-reqs LA10 LA3)

(= (maxgrade-subtheme Subtema1) 100)
(= (maxgrade-subtheme Subtema2) 100)

)

;;objetivo: termine todas las materias
(:goal (and
(pass-degree Materia1 student1)
)
)

;; metrica a optimizar
(:metric minimize (total-time))

)
```

BIBLIOGRAFÍA

[1] BARCHINO, R., J. M. GUTIÉRREZ y S. OTÓN, «An example of Learning Management System», *IADIS / International Association for Development of the Information Society*, 2005.

[2] BEYROUTHY, C., *Models, solution methods and threshold behaviour for the teaching space allocation problem*, Tesis Doctoral, University of Nottingham, 2008.

[3] BLEAU, B. L., «Planning models in higher education: Historical review and survey of currently available models», *Higher Education*, **10**(1), págs. 153–168, 1981.

[4] BRUSILOVSKY, P., «Adaptive navigation support: From adaptive hypermedia to the adaptive web and beyond», *PsychNology Journal*, **2**, págs. 7–23, 2004.

[5] CAPUTI, V. y A. GARRIDO, «Student-oriented planning of e-learning contents for Moodle», *Journal of Network and Computer Applications*, **53**(0), págs. 115 – 127, 2015, URL `http://www.sciencedirect.com/science/article/pii/S108480451500065X`.

[6] CASTILLO, L., L. MORALES, A. GONZÁLEZ-FERRER, J. FDEZ-OLIVARES, D. BORRAJO y E. ONAINDÍA, «Automatic generation of temporal planning domains for e-learning problems», *J. of Scheduling*, **13**, págs. 347–362, 2010.

[7] CASTILLO, L., L. MORALES, A. GONZÁLEZ-FERRER, J. FERNÁNDEZ-OLIVARES y O. GARCÍA-PÉREZ, «Current Topics in Artificial Intelligence», en *Knowledge Engineering and Planning for the Automated Synthesis of Customized Learning Designs*.

[8] CHEN, Y., C.-W. HSU y B. W.WAH, «SGPlan: Subgoal Partitioning and Resolution in Planning», *American Association for Artificial Intelligence*, 2004.

[9] CONFREY, J., A. P. MALONEY y A. K. CORLEY, «Learning trajectories: a framework for connecting standards with curriculum», *ZDM*, **46**(5), págs. 719–733, 2014.

[10] DE MADRID-ESCUELA POLITÉCNICA SUPERIOR, U. C. I., «Manual de usuario PLTool», recurso libre, disponible en http://www.plg.inf.uc3m.es/software/pltool/manual-pdf/manual-es.pdf, 2007.

[11] DECKER, K., «TAEMS: A Framework for Environment Centered Analysis & Design of Coordination Mechanisms», *Foundations of Distributed Artificial Intelligence, Chapter 16*, 1996.

[12] DUQUE MÉNDEZ, N. D., C. JIMÉNEZ RAMÍREZ y J. A. GUZMÁN LUNA, «IA Planning for automatic generation of customized virtual courses», *IOS Press*, 2005.

[13] DYNARSKI, M., L. CLARKE, B. COBB, J. FINN, R. RUMBERGER y J. SMINK, «Dropout Prevention», *National Center for Education Evaluation and regional Assistance. Institute of Education Sciences. U.S. Department of Education*, 2000.

[14] FERNÁNDEZ, S. y D. BORRAJO, «Using linear programming to solve clustered oversubscription planning problems for designing e-courses», *Expert Systems with Applications*, **39**(5), págs. 5178 – 5188, 2012, URL `http://www.sciencedirect.com/science/article/pii/S095741741101551X`.

[15] FOX, M. y D. LONG, «PDDL 2.1: An Extension to PDDL for Expressing Temporal», *Journal of Artificial Intelligence Research*, 2003.

[16] GAREY, M. R. y D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, W.H., 1979.

[17] GARRIDO, A. y E. ONAINDIA, «On the Application of Planning and Scheduling Techniques to E-learning», *IEA/AIE'10 Proceedings of the 23rd Interna-*

tional Conference on Industrial Enginnering and other Applications of applied Intelligent System, 2010.

[18] GARRIDO, A., E. ONAINDIA, L. MORALES, L. CASTILLO, S. FERNÁNDEZ y D. BORRAJO, «Modeling E-Learning Activities in Automated Planning», recurso libre, disponible en http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.147.9985, 2012.

[19] GARRIDO, A., E. ONAINDIA y O. SAPENA, «Automated planning for personalised course composition», *Advanced Learning Technologies, ICALT*, 2009.

[20] GEREVINI, A., A. SAETTI y I. SERINA, «Planning through Stochastic Local Search and Temporal Action Graphs», *Journal of Artificial Intelligence Research (JAIR)*, 2003.

[21] GHALLAB, M., D. NAU y P. TRAVERSO, *Automated Planning: Theory and Practice*, primera edición, Morgan Kaufmann Publishers Inc., 2004.

[22] HOFFMANN, J. y B. NEBEL, «The FF planning system: fast plan generation through heuristic search», *Journal of Artificial Intelligence Research*, **14**, págs. 253–302, 2001.

[23] HSU, C.-W. y B. W.WAH, «The SGPlan Planning System in IPC-6», *Association for the Advancement of Artificial Intelligence*, 2008.

[24] ICAPS, «Home Page ICAPS», recurso libre, disponible en http://ipc.icaps-conference.org/, 2013.

[25] KARAMPIPERIS, P. y D. SAMPSON, «Adaptive learning resources sequencing in educational hypermedia systems», *Educational Technology and Society*, **8**, págs. 128–147, 2005.

[26] KEENOY, K., M. LEVENE y D. PETERSON, «WP4 Deliverable 4.2 Personalisation and Trails in Self e-Learning Networks», *Informe técnico*, Birkbeck College and Institute of Education, University of London, 2003.

[27] KEENOY, K., A. POULOVASSILIS, G. PAPAMARKOS, P. T. WOOD, V. CHRISTOPHIDES, A. MAGKANARAKI, M. STRATAKIS, P. RIGAUX y

N. SPYRATOS, «Adaptive Personalisation in Self e-Learning Networks», en *Proceedings of the 1st International Kaleidoscope Learning Grid Special Interest Group Conference on Distributed e-Learning Environments*, KLGW'05, British Computer Society, Swinton, UK, UK, págs. 2–2, 2005, URL `http://dl.acm.org/citation.cfm?id=2215738.2215740`.

[28] LAWRENCE, S. M., K. D. LAWRENCE y G. R. REEVES, «Allocation of teaching personnel: A goal programming model», *Socio-Economic Planning Sciences*, **17**(4), págs. 211–216, 1983.

[29] MAYA PADRÓN, C., *Modelación de planes de estudio usando técnicas avanzadas de planificación*, Tesis de Maestría, Universidad Autónoma de Nuevo León, 2013.

[30] MCDERMOTT, D. M., «The 1998 AI planning systems competition», *AI magazine*, **2**, págs. 35–55, 2000.

[31] MORALES REYNAGA, L. C., *Generación Automática de Diseños de Aprendizaje: Diferentes Enfoques de Planificación*, Tesis de Maestría, Universidad de Granada, 2008.

[32] MORALES REYNAGA, L. C., *Generación Automática de Diseños de Aprendizaje: Diferentes Enfoques de Planificación*, Tesis Doctoral, Universidad de Granada, 2011.

[33] MORALES REYNAGA, L. C. y M. G. JIMÉNEZ GUZMÁN, «Planificación Automática de Disenos Instruccionales en Entornos Virtuales de Aprendizaje: un Reto Multidisciplinario», *Temas de Ciencia y Tecnología*, **15**(47), págs. 11–20, 2012.

[34] MUSTAFA, A. y M. GOH, «Multi-criterion models for higher education administration», *Omega*, **24**(2), págs. 167–178, 1996.

[35] ORTIZ, I. y E. PALAFOX, «Problemas de los estudiantes con relaci´on a su ingreso, trayectoria escolar y egreso.», *Informe técnico*, Seminarios de Diagn´ostico Locales. Comisi´on Especial para el Congreso Universitario. Universidad Aut´onoma de M´exico., 2014.

[36] PAL, B. B., M. KUMAR y S. SEN, «A priority-based goal programming method for solving academic personnel planning problems with interval-valued resource goals in university management system», *International Journal of Applied Management System*, **4**(4), págs. 284–312, 2013.

[37] PAPADIMITRIOU, C. H., *Computational Complexity*, Addison Wesley, 1993.

[38] PASHLER, H., P. BAIN, B. BOTTGE, A. GRAESSER, K. KOEDINGER, M. MCDANIEL y J. METCALFE, «Organizing Instruction And Study to Improve Student Learning, A Practice Guide», *National Center for Education Research. U.S. Department of Education*, 2007.

[39] PETCH, J., «Instructional Management Systems (IMS)», recurso libre, disponible en http://www.ncgia.ucsb.edu/ige98/report/ims.html, 2012.

[40] RUSSELL, S. y P. NORVIG, *Inteligencia Artificial. Un enfoque Moderno*, primera edición, Prentice Hall Hispanoamerica, S.A., 1996.

[41] RUSSELL, S. y P. NORVIG, *Inteligencia Artificial. Un enfoque Moderno*, segunda edición, Pearson Educación, 2004.

[42] RUSSELL, S. J. y P. NORVIG, *Artificial Intelligence: A Modern Approach*, segunda edición, Prentice Hall, 2003.

[43] SCHAEFFER, S. E., «Diseño y análisis de algoritmos», Notas de clase, disponible en http://elisa.dyndns-web.com/teaching/aa/pdf/diap.pdf, 2014.

[44] SCHROEDER, R. G., «Resource planning in university management by goal programming», *Operations Research*, **22**(4), págs. 700–710, 1974.

[45] SMITH, G. E. y S. THRONE, *Differentiating Instruction with Technology in k-5 Classrooms*, primera edición, International Society for Technology in Education (ISTE), 2007.

[46] ULLRICH, C. y E. MELIS, «Complex Course Generation Adapted to Pedagogical Scenarios and its Evaluation», *Educational Technology and Society*, **13**(2), págs. 102–115, 2010.

[47] VASSILEVA, J., «Dynamic courseware generation: at the cross point of CAL, ITS and authoring», *Proceedings International Conference on Computers in Education, ICCE*, **95**, págs. 290–297, 1995.

[48] VASSILEVA, J., «Dynamic Course Generation», *Journal of Computing and Information Technology*, **5**, págs. 87–102, 1997.

[49] VASSILEVA, J. y R. DETERS, «Dynamic courseware generation on the WWW», *British Journal of Educational Technology*, **29**(1), págs. 5–14, 1998.

[50] WALTERS, A., J. MANGOLD y E. HARAN, «A comprehensive planning model for long-range academic strategies», *Management Science*, **22**(7), págs. 727–738, 1976.

[51] WINSTON, P. y B. HORN, *LISP*, tercera edición, Addison-Wesley, Estados Unidos de América, 1989.

FICHA AUTOBIOGRÁFICA

Cristina Maya Padrón

Candidato para el grado de Doctor en Ingeniería
con especialidad en Ingeniería de Sistemas

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Tesis:

DISEÑO Y EVALUACIÓN DE MODELOS DE PLANIFICACIÓN DE INTELIGENCIA ARTIFICIAL Y PROGRAMACIÓN MATEMÁTICA PARA GENERAR RUTAS DE APRENDIZAJE

Estudió la Maestría en Ciencias en Ingeniería de Sistemas en la Facultad de Ingeniería Mecánica y Eléctrica de la Universidad Autónoma de Nuevo León, egresada en junio de 2013. Licenciada en Informática Administrativa por la Facultad de Contaduría Pública y Administración de la Universidad Autónoma de Nuevo León, egresada en el año 2008 con Reconocimiento "Mención Honorífica" por desempeño académico sobresaliente.