UNIVERSIDAD POLITECNICA DE VALENCIA

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Tesina del Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital.

Diseño e implementación de un planificador para entornos de e-learning

Autor: Vicente Adriá Bohigues

Directores: Eva Onaindia de La Rivaherrera
Antonio Garrido Tejero

INDICE.

1. INTRODUCCIÓN.

A lo largo de los años, la enseñanza siempre se ha entendido como una actividad destinada a amplios colectivos donde el profesor imparte unos conceptos a través de una clase presencial a un conjunto, en principio, heterogéneo de alumnos. Tradicionalmente, el proceso de enseñanza no ha tenido en cuenta los diferentes conocimientos o capacidades de los alumnos, pero actualmente el proceso educativo está evolucionando progresivamente hacia modelos más personalizados que permiten atender las necesidades individuales de los alumnos en un tiempo más corto. Con la inclusión de las nuevas tecnologías de la información en los procesos educativos, se ha conseguido que además de dar la clase común a todos los alumnos se pueda reforzar el material docente con otros instrumentos como ejemplos, simulaciones… con los que se puede ampliar conocimientos si fuese necesario en el mismo momento en que se están explicando debido a que estos materiales se pueden facilitar con antelación en algún formato en el que el alumno tenga acceso durante la clase (pdf, transparencias…). Así en estos nuevos modelos muchas de las dudas planteadas se pueden resolver in-situ, aunque se continúe dependiendo de las tutorías. Por diversos motivos siempre hay personas que no pueden asistir a esos cursos presenciales (trabajo, incapacidad…), por eso surgió un nuevo modelo de tele enseñanza, actualmente conocido como *e-learning*, donde la enseñanza se imparte de forma remota. Es en este escenario donde las tecnologías de la información han jugado un papel más importante ya que ha permitido que la distribución de contenidos y la realimentación por parte del alumno fuese mucho más factible, propiciando así que muchas más personas puedan acceder a cursos que les sería imposible acceder presencialmente. Uno de los problemas que plantea el *e-learning* es que dado que tiene mucha mas accesibilidad que los cursos tradicionales, los alumnos pueden llegar con conocimientos muy distintos adquiridos en distintas ramas profesionales, esto hace que una enseñanza estándar para todos resulte inadecuada para la mayoría, ya que a unos les resultará demasiado complicado por no tener suficientes conocimientos iniciales, y a otros les resultara sencillo. Es aquí, concretamente, donde se hace necesario que el proceso de educación se adapte a cada alumno y para ello las técnicas tradicionales no sirven puesto que sería muy costoso preparar un curso o temario ad-hoc a cada persona.

Actualmente ya existen especificaciones para el diseño y modelado de procesos de *e-learning*, como el estándar IMS [IMS08], donde se definen métodos para plantear y diseñar cursos, donde un curso se ve como una especie de mapa el cual contiene varios caminos para llegar desde el principio del curso a la finalización del mismo. Una vez planteado el curso se debe tener en cuenta los perfiles de los estudiantes para seleccionar un camino, o ruta de aprendizaje como se llama habitualmente en este campo, concreto para cada estudiante. Es en este punto donde la tarea puede complicarse si el curso es muy amplio u además se ha tenido muy en cuenta los diferentes tipos de perfiles de los alumnos a la hora del diseño. Por eso la planificación en IA tiene mucho que ofrecer al campo del *e-learning* ya que la introducción de sistemas inteligentes que ayuden al diseño de los cursos y a la recomendación de rutas concretas dependiendo de las características de cada alumno en un gran avance. De hecho ya se han hecho importantes avances en la generación automática de cursos para *e-learning*, aunque desde diversos planteamientos. Uno de los primeros proyectos en esta dirección es el trabajo de Peachy y McCalla [Peachy&McCalla86], en el que el material didáctico se estructura en conceptos de aprendizaje y se definen prerrequisitos conceptuales que establecen relaciones causales entre los distintos conceptos. Esta

perspectiva instruccional junto con técnicas de planificación en IA se combinan para generar secuencias de conceptos que los alumnos tienen que adquirir para conseguir los objetivos planteados. El trabajo de Vassileva [Vassileva97] también diseña un sistema que genera dinámicamente cursos basados en una representación explicita de una estructura de conceptos y librerías de recursos. Otros trabajos [Karampiperis&Sampson] sugieren el uso de ontologías y LOM (metadatos de objetos de aprendizaje) [LOM07], que no es mas que un dialecto especificado en XML para describir dichos objetos. Otros [Ullrich07] proponen el uso de planificadores jerárquicos para representar objetivos pedagógicos y tareas para conseguir esos objetivos. Y PASER [PASER07], uno de los trabajos más recientes, propone un sistema basado en ontologías que consiste en un repositorio de metadatos, un sistema deductivo orientado a objetos de razonamiento sobre los metadatos y un planificador que genera rutas automáticamente. En general, el comportamiento de la mayoría de estos sistemas se basa en la adquisición de objetos guardados en repositorios y su combinación mediante las relaciones definidas como metadatos para construir una ruta o plan que satisfaga los objetivos propuestos.

En el proyecto que engloba esta tesina se ha optado por un enfoque instruccional que además cumpla los estándares actuales de *e-learning*, como podrían ser las diferentes versiones de IMS. Y que proporcione además un soporte completo desde el diseño del curso hasta la ejecución y monitorización del mismo.

2. MARCO DE TRABAJO.

El proyecto Adaptaplan (Adaptación basada en aprendizaje, modelado y planificación para tareas complejas orientadas al usuario, MEC TIN2005-08945-C06-06) es un proyecto dependiente del MEC (Ministerio de Educación y Ciencia) que en colaboración con otras universidades españolas cuyo objetivo es estudiar la mejora de los procesos de *e-learning* mediante la aplicación de técnicas de IA durante todo el proceso de enseñanza, es decir, desde que se diseña un curso, hasta que se evalúa el resultado del mismo. Su aplicación en este campo consiste en desarrollar un tutor personal inteligente capaz de proponer un plan de actuación respecto a las posibles vías que existan en el transcurso de una asignatura. Cada plan consistirá en una secuencia de tareas (por ejemplo, realización de ejercicios, estudio de un tema de teoría, pruebas puntuables, etc.) predefinidas con anterioridad por el diseñador del curso y que estarán lo más adaptadas posibles al perfil del estudiante para que al seguir el plan establecido se consigan los objetivos propuestos. Para ello se ha determinado como objetivos del proyecto:

- La realización de un análisis de las necesidades existentes y los componentes necesarios para crear rutas de aprendizaje. La elaboración de estas rutas se corresponde con un proceso de planificación en IA el cual tendrá que ser adaptado para adecuarse a las necesidades establecidas por los entornos de *e-learning*.

- La construcción de un sistema que sea capaz de generar rutas de aprendizaje totalmente adaptadas al perfil de cada estudiante y que garantice una mayor calidad en el proceso de enseñanza-aprendizaje y su posterior puesta en marcha y evaluación de este sistema respecto a su finalidad.

Para conseguir estos objetivos se ha distribuido el proyecto en cuatro fases, que son:

1. La definición del dominio, donde se especifican los elementos del dominio y como se representan o adecuan a una representación para planificación como pueden ser el alumno, los materiales del curso, los recursos del centro, los conceptos y las tareas que intervendrán en el curso.

2. La generación propiamente de las rutas, en las que utilizando un planificador se pueda obtener una solución en función de los perfiles, los objetivos concretos del curso, y las prioridades entre los objetivos. Estas rutas contendrán la información, en forma de secuencia de acciones, necesaria para alcanzar los objetivos determinados. Aunque se debe permitir cierto grado de flexibilidad en el caso de que surjan imprevistos en la realización del curso.

3. La tercera fase consiste en la ejecución y monitorización de las rutas, donde se debe permitir durante la realización del curso una readaptación o replanificación a partir de un punto concreto en el caso de que no se pueda seguir encamino marcado.

4. Y por ultimo la evaluación del sistema desarrollado con la realimentación proporcionada por los estudiantes y el profesor para introducir mejoras si fuese necesario.

Este proyecto tiene como objetivo mejorar el proceso de educación desde diversos puntos de vista. Desde un punto de vista científico, se obtendrá un ciclo completo de desarrollo en la generación de cursos personalizados y su seguimiento. Desde un punto de vista social, los alumnos se benefician, ya que un proceso de educación mas adaptado redunda en un mejor aprovechamiento del tiempo dedicado a su formación y los profesores se benefician de la utilización de un sistema que permite realizar desde la planificación de contenidos hasta la evaluación del curso. Y por ultimo, desde un punto de vista económico supondrá un claro ahorro al reducir los costes necesarios para diseñar y poner en marcha un curso determinado ya que la utilización de los recursos necesarios estará optimizada.

Concretamente este trabajo se centra en la parte del desarrollo del planificador que generará las rutas automáticamente en base a unas especificaciones determinadas. La solución generada por el mismo debe poder visualizarse a modo de una secuencia de tareas, las cuales están relacionadas por enlaces causales, que conduzca a la consecución del objetivo. En los momentos en los que la decisión del camino a tomar dependa de la ejecución del plan (dependiendo de la nota de un examen) se tendrá en cuenta una opción por defecto para no representar todas las opciones posibles, pero si en el momento de la ejecución no se cumplen las condiciones esperadas se replanificara a partir del momento concreto en el que se encuentre el plan y se mostrara el nuevo camino a seguir. Además debe generarse un plan por cada alumno, es decir, una secuencia de acciones por cada alumno representado en el problema, estas acciones de diferentes alumnos se instanciarán en el tiempo mediante un CSP/Scheduler el cual combinará la ejecución de todos los planes para satisfacer las restricciones temporales impuestas en el problema además de establecer los recursos comunes que utilizara cada alumno en cada momento. Por ejemplo, si una tarea debe realizarse por 5 alumnos, el

scheduler instanciará esta tarea concreta de 5 alumnos para que se realice al mismo tiempo, y si es necesario asignará los recursos a utilizar.

3. ARQUITECTURA.

La arquitectura propuesta para este proyecto consiste en cuatro módulos principales, los cuales se muestran en la figura 1. La idea es que el profesor o el diseñador del curso especifique las características del curso basándose en un modelo de representación explicito en el cual se debe especificar los conocimientos del curso y las relaciones entre ellos mediante las tareas que los pueden conseguir. Para ello el primer módulo con el que cuenta el profesor es la herramienta de modelado, donde progresivamente puede ir refinando los conceptos y el planteamiento del curso hasta llegar a un nivel básico de tareas que deben realizar los alumnos. En esta herramienta también se debe definir el conjunto de estudiantes que tendrá el curso incluyendo sus perfiles y demás características. La herramienta de diseño genera automáticamente a partir de los esquemas especificados, una versión en XML y otra en PDDL [PDDL98] (PDDL es un lenguaje de representación de problemas específicamente diseñado para planificación), la cual usará el planificador, se utiliza PDDL debido a que es un estándar ampliamente usado en la comunidad de planificación en IA, aunque esto es totalmente transparente al usuario de la herramienta.

Una vez diseñado el curso incluyendo los alumnos se pasa al siguiente modulo el cual esta compuesto por un planificador y un CSP, el primero, que toma el fichero generado por la herramienta, se encarga de seleccionar una secuencia de tareas que tras su ejecución cumplan los objetivos propuestos, además estas tareas estarán completamente adaptadas al perfil de cada estudiante. A continuación, esta primera solución pasa al CSP, el cual es el encargado de instanciar en el tiempo esas tareas y de asignar los recursos necesarios para su correcta ejecución. En este momento se tiene un plan de acción completamente temporizado, en el caso de que el CSP no pudiera satisfacer las restricciones demandaría al planificador otra solución, ya que la actual no es viable.

Cuando se ha obtenido un plan correcto y viable se pasa mediante XML a los dos módulos siguientes, el primero de ellos es el modulo de visualización, donde se representa de manera gráfica a modo de diagrama gantt o cualquier otro requerido el plan de actuación obtenido. El otro módulo es el de ejecución y monitorización, este módulo entra en funcionamiento cuando se ha aceptado el plan como una buena solución y se pone en práctica, es aquí donde el alumno realiza el seguimiento de las tareas que debe realizar y de sus progresos. Además el profesor recibe una valiosa realimentación de información para poder adaptar en un futuro el diseño del curso. Por ejemplo, si existe un alto índice de fracaso en una tarea, puede que se necesite reforzar la forma de adquirir esos conocimientos. También el profesor puede realizar el seguimiento de los alumnos en este módulo y si no se cumple el plan previsto, replanificar la ruta a partir del punto actual.

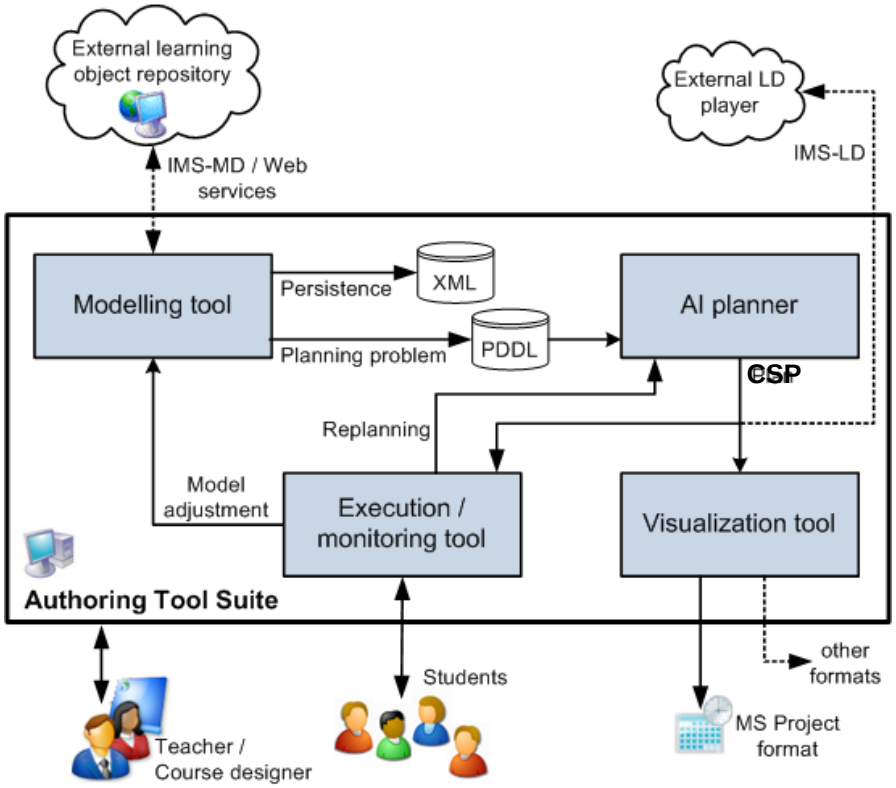


Figura 1. Arquitectura de la herramienta.

Una vez vista la interacción entre los distintos módulos cada uno de los cuales se puede realizar de forma independiente mientras se mantenga la comunicación entre ellos, se amplían los detalles de cada uno en las siguientes secciones.

3.1 HERRAMIENTA DE DISEÑO.

La herramienta de diseño proporciona al profesor un método sencillo como es una herramienta grafica basada en "arrastrar y soltar" de plantear y diseñar cursos que sean lo suficientemente flexibles como para poder adaptarlos específicamente a distintos tipos de estudiantes. Siguiendo un modelo instruccional podemos identificar tres elementos principales, tareas, conceptos, y perfiles de alumnos. Para definirlos en diferentes pasos y que el proceso de diseño sea un proceso de refinamiento desde un diseño más general a un diseño específico, la herramienta cuenta con tres niveles de diseño o vistas correspondientes a los elementos principales identificados, según el concepto que se vaya a diseñar:

- La vista conceptual es la primera vista donde se empieza a plantear la estructura del curso. En esta vista se representan los conceptos en forma de un mapa conceptual, al estilo de IMS-LD [IMS08], un estándar utilizado en entornos de educación, el cual plantea que tipo de relaciones pueden tener unos conceptos con otros, por ejemplo: *basado en*, *requerido por*, *recomendado para*… También se pueden establecer restricciones AND/OR entre distintas relaciones (un concepto puede requerir varios más a la vez). Con esta vista se consigue un grafo que representa los conceptos importantes que intervienen en el curso además de especificar el punto de partida y finalización del mismo.

- La vista de tareas es la segunda vista que utiliza la herramienta, cada concepto de la vista anterior se descompone en un diagrama de esta vista. Aquí es donde se representa cómo se consigue dicho conocimiento. Esta vista se compone únicamente de las tareas necesarias para obtener el conocimiento de la vista anterior. Las relaciones entre las tareas especifican los requisitos que necesita cada una de ellas, habitualmente relaciones de precedencia. También se define aquí la duración de las mismas y su número de repeticiones, que habitualmente será uno, pero que puede tratarse de una tarea de refuerzo la cual se puede repetir varias veces.

- La vista relacionada con los perfiles de alumnos es la tercera vista. Cada esquema generado por la segunda vista se traslada automáticamente a esta vista, en la cual se realiza la adaptación de las tareas genéricas dependiendo del perfil del alumno. Esta vista es ligeramente diferente a las otras dos ya que combina en una misma vista conceptos y tareas. Las relaciones existentes en esta vista, que también pueden contener restricciones de tipo *requires/basis-for* entre ellas, están limitadas a concepto-tarea y tarea-concepto, no permitiendo la relación entre dos elementos de la misma clase. Esto es así porque permite una estructura semejante a la utilizada en planificación, además de ser un punto de partida donde el profesor puede refinar el modelo adaptando cada tarea a cada tipo de perfil si la tarea lo requiere. En cada tarea se especifica además de los materiales y recursos utilizados por la misma, el resultado obtenido en función del perfil del estudiante (para un perfil determinado unas tareas determinadas pueden ser más provechosas que para los demás).

Para ver la forma en la que se plantea el diseño de un curso se ha desarrollado el siguiente ejemplo sobre un curso de programación en JAVA. En la figura 2, correspondiente a la primera vista, se puede observar que se han definido los conceptos generales que se tratan en el curso como la *estructura de un programa*, *variables y tipos*, *estructuras condicionales*, etc. Además se ha representado el inicio y el fin del curso. Las relaciones que aparecen representan el modo en que se relacionan los conceptos como por ejemplo que el concepto *variables y tipos* se requiere para las sentencias condicionales.

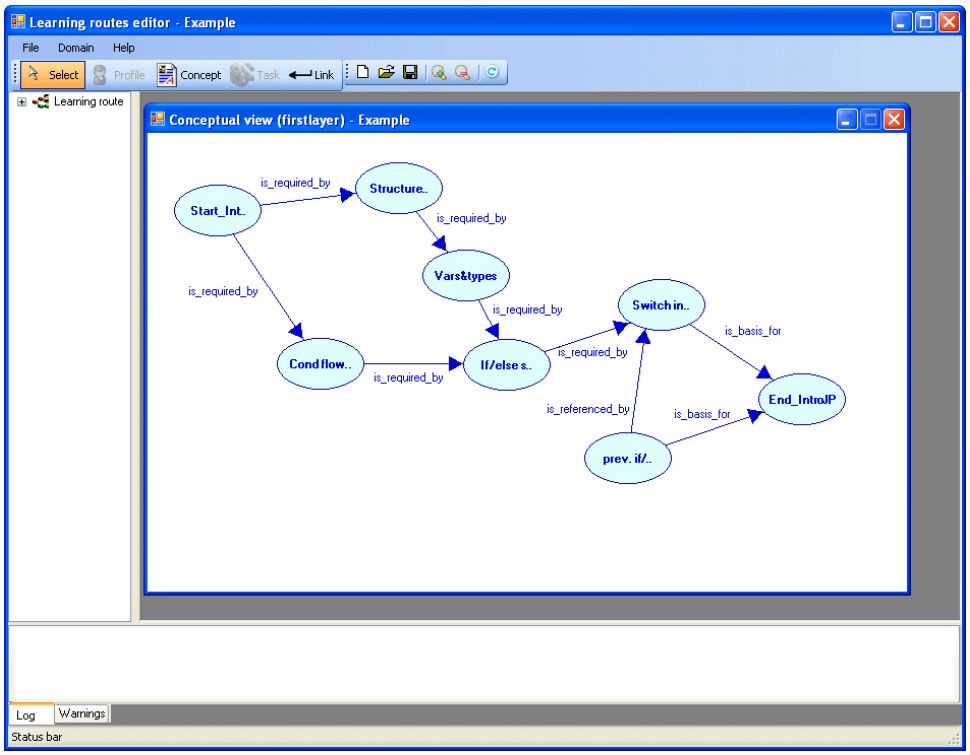


Figura 2. Primera vista de la herramienta de diseño.

En la figura 3, correspondiente a la segunda vista, se puede observar el desglose correspondiente a las tareas del concepto *variables y tipos*, el cual también tiene una tarea de inicio y de fin. En esta vista se determinan cuales son las tareas que se llevarán a cabo para conseguir dicho concepto, se puede observar también que existen varios caminos para lograrlo.

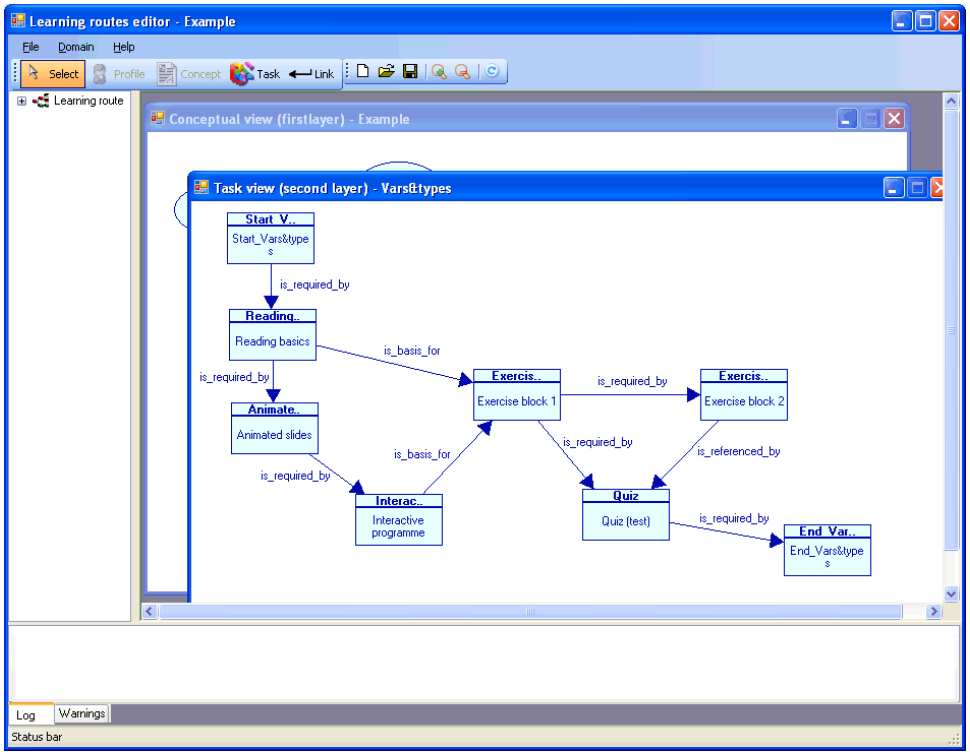


Figura 3. Segunda vista de la herramienta de diseño.

En la figura 4, correspondiente a la tercera vista, se ve como además de las tareas y los conceptos auto-generados, se introducen los perfiles y como en función de ellos las tareas generan mas o menos porcentaje del concepto conseguido. Tomando como referencia la tarea *animación* la cual recibe comportamientos para varios perfiles diferentes, vemos que genera un 20% si el alumno no se adapta a ningún perfil de los indicados, un 50% extra si el perfil es visual, un 50% extra si el perfil es verbal, y/o un 20% extra si el perfil es inductivo.

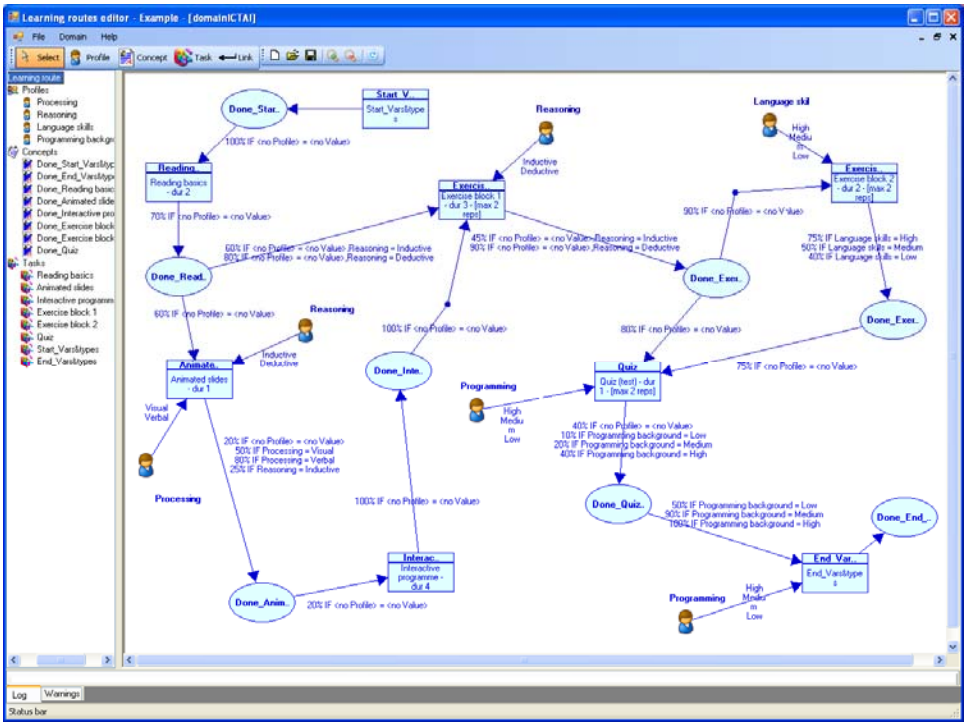


Figura 4. Tercera vista de la herramienta de diseño.

Esta herramienta ya está desarrollada y en funcionamiento, siendo utilizada para preparar escenarios de uso para el planificador.

3.2 HERRAMIENTAS DE PLANIFICACIÓN.

Este módulo se compone de dos aplicaciones separadas la primera de ellas es el planificador, el cual proporciona la secuencia de acciones que debe seguirse para conseguir los objetivos, además de asegurar que estas acciones estén adaptadas lo mejor posible al perfil de cada estudiante. El planificador se ha diseñado y desarrollado específicamente para este proyecto, el cual cubre esta tesina y se explica en detalle en la sección 4. El planificador necesita una representación en PDDL propia del entorno de planificación, para ello se toma la tercera vista y se genera a partir de ella tanto el dominio como el problema. La estructura proporcionada en la tercera vista es muy similar a la forma clásica en la que se representan los dominios y problemas en planificación, de este modo, las tareas se convierten en acciones, los conceptos en las condiciones y efectos de las acciones y las restricciones sobre perfiles se añaden como condiciones de las acciones. El planificador no tiene en cuenta datos como los recursos a la hora de planificar, ya que estos se utilizaran posteriormente en el siguiente paso, el CSP.

El CSP es el encargado de tomar la secuencia parcialmente ordenada obtenida del bloque anterior e instanciar cada acción en el tiempo de forma que todas las tareas tengan un momento determinado para su realización. Además, también es el encargado de asignar los recursos disponibles para la realización de las tareas a los alumnos correspondientes de manera que se cumplan las restricciones de capacidad y temporales para que el plan se lleve a cabo con éxito. Si no se puede cumplir el plan por falta de recursos, o no cumplir restricciones temporales se llama al planificador para que le entregue otra solución que las satisfaga. El CSP se encuentra aun en fase de diseño.

3.3 VISUALIZACION DEL PLAN.

Este modulo es la capa de presentación de los resultados del planificador y del CSP donde se muestra un plan ya válido y factible ya que cumple todas las restricciones impuestas para culminar el curso con éxito. Esta representación puede ser visualizada en varios formatos como HTML o XML, y en medios distintos al PC pueden ser teléfonos móviles, PDAs, lo que aumenta la movilidad de los alumnos cuando siguen un curso mediante *e-learning*. Actualmente se ha optado por una representación básica, exportando el plan un formato XML que utiliza MS Project (conocido habitualmente por estudiantes y profesores) el cual permite una rápida y sencilla visualización mediante un diagrama gantt como el de la figura 5. Además, en esta representación se comprueba fácilmente información como la duración total del plan, los recursos que consume, el coste de puesta en marcha, etc.

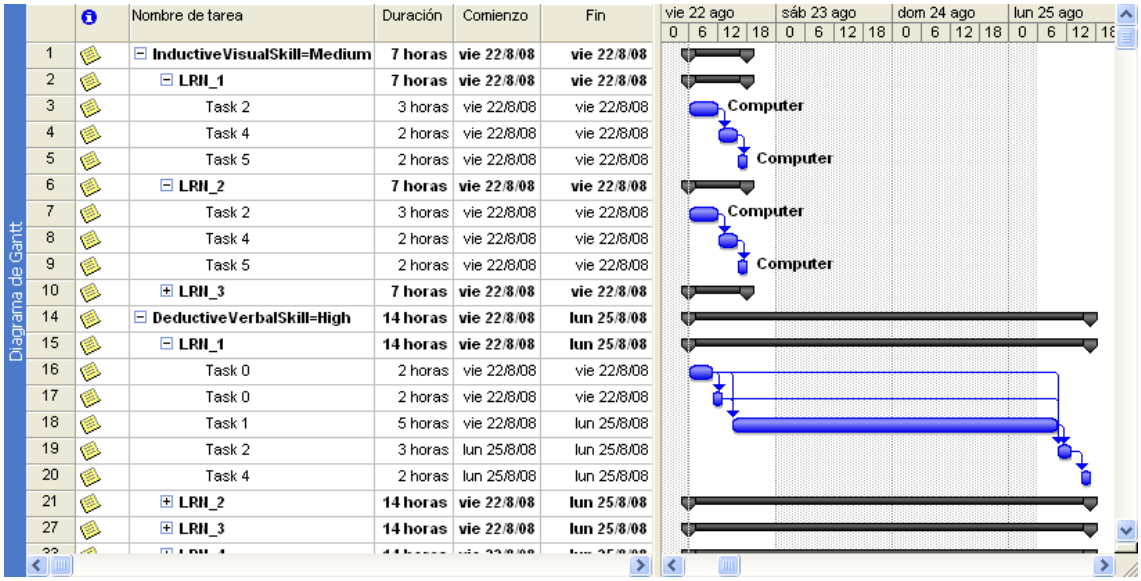


Figura 5. Vista de los resultados en MS Project.

3.4 EJECUCIÓN Y MONITORIZACIÓN.

Este módulo permite al alumno hacer el seguimiento de las acciones que debe realizar mostrando las que ya ha realizado y las que le quedan por cumplir. En el caso de que el estudiante no cumpliese correctamente una tarea, por ejemplo sacase menos nota de la requerida en un examen, este módulo iniciaría un proceso de replanificación a partir del estado actual para adaptarse a las nuevas condiciones. También el profesor

puede realizar un seguimiento de los alumnos y recibir una realimentación conforme está evolucionando el curso para, dado el caso, modificar el diseño del mismo de cara a mejorar el proceso de enseñanza-aprendizaje para la siguiente edición. Por ejemplo, si se falla sistemáticamente en una tarea, podría ser recomendable reforzar los conocimientos necesarios para la misma. Este módulo aún está en fase de diseño.

4. MÓDULO DE PLANIFICACIÓN.

4.1. UN POCO DE HISTORIA SOBRE LA PLANIFICACIÓN EN IA.

Un problema de planificación en IA [Ghallab04] se puede definir como un problema de búsqueda que requiere encontrar una secuencia eficiente de acciones para conducir un sistema desde un estado inicial a un estado objetivo. Por lo tanto el objetivo del campo de la planificación en IA es construir algoritmos capaces de encontrar una secuencia de acciones que resuelvan el problema planteado.

Si pensamos en un ejemplo sencillo como puede ser el mundo de bloques donde el “mundo” se compone de bloques apilables y un brazo capaz de moverlos, apilarlos y desapilarlos, quizá parezca una tarea fácil debido a que las personas pueden encontrar una solución relativamente rápida si el numero de bloques no es muy elevado. Por el contrario los algoritmos de planificación tratan el problema como una búsqueda dentro de todas las combinaciones posibles existentes en este “mundo”. Visto así no tendría mucho sentido utilizar estos algoritmos pero en situaciones donde el número de bloques fuese muy elevado sería muy costoso para una persona trazar un plan de actuación. E incluso si trasladamos el mundo de bloques al mundo real, una aplicación bastante similar sería la carga y descarga de un buque de contenedores donde existen una serie de contenedores que deben ser apilados y desapilados dentro del buque. Esto ya de por sí añade mucha complejidad debido a que un buque de estas características transporta 3000 contenedores sin mucha dificultad, si además sabemos que hay que tener en cuenta el destino de los contenedores que hay ya en el buque como el de los nuevos que se carguen, que el peso debe estar compensado, etc. estamos frente a un problema que puede desbordar a una persona. Es aquí donde los algoritmos de planificación juegan un papel importante ya que son capaces de obtener un plan para este problema.

Como la combinatoria existente en el problema antes mencionado haría imposible la exploración completa de todas las posibilidades, un elemento clave de estos algoritmos es la introducción de funciones heurísticas, que no son más que conocimiento expresado de forma que el sistema pueda entenderlo, para guiar este proceso de búsqueda. Así, un algoritmo de planificación puede resolver problemas complejos de búsqueda en situaciones donde no es obvio cuál es el siguiente paso a tomar para llegar a cumplir el objetivo.

De todas formas aplicar la planificación al mundo real sería demasiado costoso y haría imposible encontrar una solución debido al enorme numero de factores que pueden influir, por eso la mayoría de planificadores trabajan sobre un modelo restringido del mundo real, el cual se conoce habitualmente como planificación clásica. Este modelo asume ciertas condiciones que relajan el problema real y hacen mucho más fácil la labor de buscar una solución a un problema. Estas asunciones son:

- **Modelo observable:** El planificador conoce todos los datos relevantes al problema y no hay información desconocida por el.

- **Modelo estático:** Se puede predecir el estado del mundo en un momento determinado ya que los elementos de este no cambian si no es por la acción del planificador. No existen factores externos que puedan afectar las condiciones del estado actual.

- **Modelo determinista:** El resultadote aplicar una acción en unas determinadas condiciones es siempre el mismo, lo que permite que dada una secuencia reacciones se pueda predecir el estado obtenido.

- **Enfoque proposicional:** Todas las variables del modelo pertenecen al dominio lógico, es decir que las variables empleadas solo pueden tomar valores *verdadero* o *falso*.

- **Acciones instantáneas:** Todas las acciones del modelo tienen la misma duración y son instantáneas, lo cual impide que una acción modifique datos que esta utilizando otra acción.

- **Acciones no descomponibles:** Las acciones son atómicas y no se pueden descomponer en sub-acciones.

- **Planificación offline:** El proceso de planificación se lleva a cabo antes de la ejecución de cualquier parte del plan obtenido. Se debe obtener un plan completo antes de proceder a la ejecución del mismo.

Incluso con estas restricciones estos problemas son demasiado complejos para los métodos tradicionales de resolución de problemas, es por eso que el uso de algoritmos de planificación extiende el espacio de problemas que se pueden resolver.

4.1.1. EVOLUCIÓN DE LA PLANIFICACIÓN.

El sistema STRIPS, que fue el primer sistema-lenguaje desarrollado para resolver problemas de planificación tal y como se entienden actualmente, define los problemas como el estado inicial del problema, el objetivo a conseguir y el conjunto de operadores que pueden actuar sobre el estado actual. Un estado se entiende como la representación del conjunto de características o propiedades de los objetos que intervienen en el problema junto con el valor de las mismas. En STRIPS se trabaja con literales de primer orden, es decir que cada variable tiene un valor *verdadero* o *falso*. Además los estados se componen solo de variables cuyo valor es verdadero y dando a entender que cualquier variable no instanciada tiene un valor falso por defecto, esto es así porque se asume la hipótesis de mundo cerrado. Por ejemplo, en el problema anterior del mundo de bloques las variables representarían a cada bloque y su posición. En el caso del predicado pos(?bloque, ?objeto) representaría que un bloque determinado (?bloque) se encuentra en sobre el objeto (?objeto) ya sea este otro bloque o el suelo. Cualquier combinación que no se liste explícitamente no es una situación que se de en el estado actual, y por tanto tiene un valor falso.

Los objetivos se definen del mismo modo que los estados, mediante los valores de las propiedades que se quieran conseguir, pero a diferencia del estado inicial, en los objetivos solo se especifica aquellos valores que realmente deban tener ese valor, dejando los valores de las propiedades cuyo valor es indiferente sin definir en el objetivo.

Los operadores son el medio por el cual el estado es modificado, un operador toma como entrada un estado actual del problema, efectúa los cambios requeridos en el mismo, y devuelve el nuevo estado. Una acción es la instanciación de unos valores concretos en un operador. Cada operador consiste en tres partes que lo definen:

- Nombre y lista de argumentos, por ejemplo un operador mover aplicado al problema del mundo de bloques podría ser mover (?bloque, ?objeto) que representaría que el bloque indicado se mueva sobre el objeto indicado. El nombre del operador es mover, y los argumentos son ?bloque y ?objeto.

- Precondiciones, son las condiciones necesarias para que el operador se pueda aplicar, por ejemplo que tanto el bloque como el objeto de destino estén libres. Estas condiciones se deben definir mediante un conjunto de literales positivos que tienen que estar representados en el estado actual del problema, como podría ser libre (?objeto) el cual representaría que dicho objeto, ya sea un bloque o el suelo esta libre.

- Efectos, son las modificaciones que aporta el operador al aplicarlo. Existen efectos positivos, los cuales se añaden a la lista de literales, y efectos negativos, los cuales indican el literal que debe ser borrado de la lista. Por ejemplo si movemos el bloque 1 sobre el bloque 2, ambos libres en este momento se añadiría un literal que represente la nueva posición del bloque 1 y se borraría la vieja posición del bloque 1 y que el bloque 2 esta libre.

Una acción se puede aplicar cuando todas sus precondiciones tengan un valor verdadero en el estado actual. Si no se puede aplicar, no produce ningún efecto ya que los cambios no se han producido. Una de las grandes aportaciones de STRIPS es la asunción de que los únicos cambios producidos en el sistema son aquellos literales que se mencionan explícitamente como efectos de las acciones, y que el resto de literales no involucrados continúan satisfaciéndose en el nuevo estado, esta asunción evita la complejidad del problema marco introducida en el trabajo de McCarthy [MCCARTHY69].

La solución al problema se denomina plan, cuya representación toma forma de una secuencia ordenada de acciones cuya ejecución conduce del estado inicial a un estado en el que se cumplen los objetivos.

El lenguaje STRIPS se planteo como una forma de diseñar algoritmos simples y eficientes que no complicaran en exceso la definición de problemas reales. Dada la limitada expresividad permitida por el mismos, en los últimos años se han desarrollado nuevos lenguajes entre los que destaca PDDL, el cual se ha convertido en el estándar actual de planificación y del que existen varias versiones siendo la mas reciente PDDL3 donde re engloban las características de las versiones precedentes y se aportan nuevas

funcionalidades. Las aportaciones de PDDL respecto a STRIPS y el modelo clásico de planificación son:

- **Acciones con duración**: las acciones ya no son instantáneas sino que requieren determinado tiempo para su ejecución, esto implica que las precondiciones y efectos tengan asociado un punto de ejecución en el que o bien se debe cumplir la condición o se produce el efecto, este punto puede ser el inicio o final de la acción para los efectos y los mismos o durante toda la acción para las precondiciones.

- **Expresiones y variables numéricas**: además de contar con los predicados con valores verdadero/falso, PDDL añade la posibilidad de utilizar variables con valores numéricos pudiendo representar las precondiciones por ejemplo en forma de umbrales.

- **Métricas:** Permiten definir el uso de funciones de optimización como una combinación lineal de uno o varios factores del problema y así obtener una medida de cuan buena es la solución obtenida y especificar un grado de calidad deseado. Por ejemplo se pede pedir que se solucione el problema en menos de *n* acciones.

- **Ventanas temporales:** Se utilizan para representar eventos externos al problema cuyo valor y momento serán conocidos por el planificador, por ejemplo que un recurso esta disponible en un horario determinado.

Con estas aportaciones PDDL permite que el modelado de problemas sea más cercano a la realidad y a la vez más fácil su representación, aportando funcionalidad no solo a la forma de conseguir el plan sino también elementos para establecer criterios de calidad de las soluciones.

4.1.2. PLANIFICACIÓN BASADA EN ESTADOS.

Este enfoque de planificación es el mas sencillo de todos, consiste en realizar una búsqueda en un espacio de estados, donde se genera un árbol de búsqueda cuyos nodos representan un estado determinado, entendiendo como estado el conjunto de literales definido en el punto anterior. Dado que las precondiciones y efectos se representan mediante literales existen se puede realizar la búsqueda en varias direcciones, hacia delante desde el estado inicial y hacia atrás desde los objetivos.

Búsqueda hacia delante.

La búsqueda hacia delante empieza representando el nodo raíz del árbol de búsqueda como el estado inicial del problema sobre el que se esté buscando, este estado viene determinado por el valor de las distintas variables al definir el problema. A partir de ese momento, por cada acción aplicable al estado que se este examinando en ese momento, se genera un nuevo estado en el que se ha aplicado la acción correspondiente en el cual esta el estado anterior a la acción con las modificaciones que produce esa acción. Una vez generado el nuevo estado, este pasa a una lista de “abiertos” o nodos no

explorados. Cuando se explora completamente un estado, es decir, se aplican todas las acciones posibles a ese estado, este se desecha y se extrae el primer nodo de la lista de "abiertos" para repetir el proceso del estado anterior. La ordenación de los nodos o estados de esa lista depende del método de búsqueda utilizado.

En la figura 6 se puede ver un árbol de búsqueda hacia delante en la se representa un momento concreto de la fase de búsqueda donde cada estado del problema esta representado por un circulo y los estados explorados tienen un doble borde en el caso de la figura se han explorado ya los nodos 1 y 2 que a su vez han generado nuevos estados con el estado anterior mas los efectos de la acción aplicada que viene representada por la flecha que une los estados. Tomando como ejemplo el nodo numero 2, este representa el estado numero uno con los cambios que efectúe la acción 1 y cuando se explora este nodo se ve que se puede aplicar las acciones 2, 3 y 4 que generan estados en los cuales se ha partido del estado del nodo anterior y se le han efectuado los cambios que produce la acción aplicada.

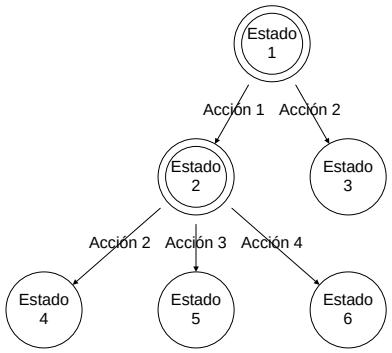


Figura 6. Árbol búsqueda hacia delante

Una vez se llega a un estado en que se cumplen todos los objetivos basta con recorrer el árbol desde del estado objetivo hasta el nodo raíz siguiendo los "padres" que generan dicho estado para obtener el conjunto de acciones que obtienen el resultado de forma ordenada.

Este tipo de búsqueda tiene el inconveniente de que se aplican a cada estado todas las acciones que cumplan sus condiciones, aunque no influyan en el resultado para obtener el plan objetivo. Esto provoca que el factor de ramificación sea muy elevado en el caso de dominios en que muchas de sus acciones sean aplicables a la vez. Si una acción deshiciese lo que se había producido anteriormente es posible que se produjesen ciclos en el árbol, pero este caso un puede darse en el dominio sobre el que trabajamos dado que todos los efectos son incrementales. Con estos argumentos se ve que no es una forma de búsqueda muy eficiente, pero con la aplicación de buenas heurísticas se convierte en una buena forma de búsqueda que mejora mucho su eficiencia ya que la heurística encamina hacia los objetivos reduciendo el número de estados intrascendentes para la obtención del plan solución.

Búsqueda hacia atrás.

La búsqueda hacia atrás es semejante ala búsqueda hacia delante, pero en este caso para generar el primero nodo se tienen en cuenta los objetivos del problema en

lugar de los valores iniciales de las variables. En cada estado de la búsqueda se representa un conjunto de objetivos, por eso también se denomina a esta, búsqueda por estados objetivos. Partiendo de un estado objetivo extraído de la lista de "abiertos" se elige un objetivo concreto de los posibles y se aplican las acciones cuyos efectos contribuyan a la consecución de ese objetivo, es decir que si una acción aporta algo para obtener el objetivo seleccionado se aplica. Cuando se aplica una acción se genera un nuevo estado objetivo en el cual están representados todos los objetivos anteriores, descontando los efectos producidos por la acción que se ha aplicado, más los objetivos propios de esa acción. Este nuevo estado objetivo es introducido en la lista de "abiertos" para su posterior exploración. Como en el caso de búsqueda hacia delante, la ordenación de la lista de "abiertos" viene determinada por el método utilizado, y no por la direccionalidad de la búsqueda.

En la figura 7 se puede ver un árbol de búsqueda hacia atrás en la se representa un momento concreto de la fase de búsqueda donde cada estado objetivo del problema esta representado por un circulo y los estados objetivo explorados tienen un doble borde. En el caso de la figura se han explorado ya los nodos 1 y 2 que a su vez han generado nuevos estados objetivo con los objetivos del estado anterior descontando los efectos producidos por la acción aplicada mas las condiciones de esa acción que se toman como nuevos objetivos del problema. Tomando como ejemplo el nodo numero 2, este representa los objetivos del estado numero 1 menos el objetivo que satisface la accion1 y añadiendo las condiciones de la acción 1. Cuando se explora este nodo se ve que se puede aplicar la acción 3 que genera un estado objetivo en el cual se ha partido de los objetivos representados en el nodo anterior y se le han añadido y descontado las condiciones y efectos que tiene esa acción

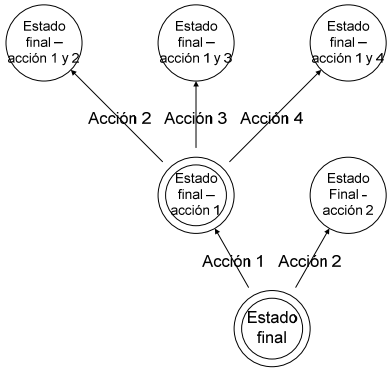


Figura 7. Árbol búsqueda hacia atrás

Una vez se llega a un estado en que no quedan objetos por satisfacer se ha llegado a una solución del problema. Para obtener el plan hay que recorrer el árbol desde el nodo sin objetivos hasta el nodo raíz para obtener la secuencia de acciones que consiguen los objetivos desde el estado inicial del problema.

Si se llega a un nodo en el que para el objetivo seleccionado no existen acciones que lo consigan, esta rama de la búsqueda se poda y se continúa buscando par los otros objetivos. Con este tipo de búsqueda se consigue un menor factor de ramificación que en una búsqueda hacia delante ya que solo intervienen en la búsqueda acciones que puedan aportar algo a la solución del problema. Además a este tipo de búsqueda también se le pueden aplicar las mismas heurísticas que a una búsqueda hacia delante

por lo tanto normalmente se consiguen mejores resultados ya que no se pierde tiempo y recursos explorando caminos con acciones que no son relevantes a la solución.

4.1.3. PLANIFICACIÓN DE ORDEN PARCIAL.

En planificación de orden total, como puede ser la basada en estados, las acciones se obtienen en el mismo orden en que posteriormente deberían ejecutarse, Esto pude provocar que si el planificador selecciona una acción equivocada deberá deshacer los cambios con otra acción adicional. Si contamos que en problemas complejos puede haber múltiples objetivos y que una acción consiga un objetivo pero deshaga un paso necesario para conseguir otro resulta que se generaran muchos caminos incorrectos. En el momento que un planificador empieza a tener esto en cuenta y debe tomar decisiones sobre como afectan las acciones que resuelven una parte del problema a la resolución del resto de problemas la planificación pasa a ser de orden parcial (POP) ya que se seleccionan acciones sin determinar su orden final en el plan. Esto permite tomar las decisiones importantes en primer lugar e ir concretando el plan poco a poco. El los POP se realiza una búsqueda desde los objetivos para ver que acciones son necesarias para lograrlos, pero a diferencia de la búsqueda en estados no se establece un orden concreto entre las acciones, el orden se determina en base a relaciones causales y otras restricciones. Además se sigue una estrategia de menor compromiso, lo que significa que se dejan sin especificar todos aquellos valores que no sea necesario, concretándolos solo cuando haga falta por razones de planificación.

Los planes parciales generados durante la búsqueda se representan mediante las siguientes características:

- **Pasos del plan:** Un paso es una versión total o parcialmente instanciada de un operador del problema. Esto significa que cada paso es una acción del problema en la que puede que no todos los parámetros estén concretados. Conforme avance la búsqueda estos irán concretándose hasta el momento en que el plan sea solución, si en este momento algún parámetro no se ha definido aun puede tener cualquier valor del domino especificado.

- **Restricciones de orden:** Las restricciones de orden se dan entre dos pasos del plan y representan que uno de ellos debe ejecutarse antes que el otro. Esto no significa que deba ejecutarse a continuación sino que el primer paso debe ejecutarse antes que el segundo, independientemente de si hay mas pasos entre ellos

- **Restricciones entre variables:** Las variables que representan los parámetros pueden tener restricciones que afectan al dominio de la misma, si un parámetro tiene todo el dominio disponible y se establece una restricción que impide que tome un valor determinado, al dominio disponible se le resta el valor de la restricción.

- **Enlaces causales y amenazas:** Un enlace causal entre dos pasos del plan representa que el primer paso consigue un literal que necesita el segundo paso. además establece una restricción de orden entre los mismos. Las amenazas se producen cuando un efecto de un paso borra un literal necesario para otro paso,

en esta situación se debe resolver mediante una restricción de orden entre ellos para que se ejecute primero el paso afectado o bien mediante una restricción sobre la variable afectada del segundo.

Las búsquedas en los POP se realizan sobre un espacio de planes donde cada nodo representa un plan parcial. El nodo raíz es un plan parcial que contiene únicamente dos pasos I y F. El paso I es un paso que contiene solo efectos en los cuales corresponden con el estado inicial, y el paso F solo contiene las condiciones representados por los objetivos. Además se establece una relación de orden entre ellos I → F. Este nodo se añade a una lista de planes donde se irán añadiendo los nodos frontera del árbol. En cada ciclo de la búsqueda se selecciona un plan de esta lista y se obtienen las precondiciones no resueltas y amenazas del plan y se introducen en una lista de pendientes, si no hay nada pendiente se ha encontrado una solución y se devuelve dicho plan. En caso contrario se selecciona uno de los problemas pendientes y se buscan todas las formas de resolverlo, por cada forma se generara otro nodo con un plan parcial añadiendo esta forma de solución y se agregara al listado de planes. Las precondiciones se pueden resolver de dos formas, la primera de ellas es reutilizar un paso existente si este resuelve dicha precondición, y su uso no genera ciclos ni amenazas. La segunda es la adición de un nuevo paso que aporte una acción que resuelva dicha precondición. Si el problema pendiente es una amenaza se soluciona mediante promoción, democión o separación de los pasos afectados.

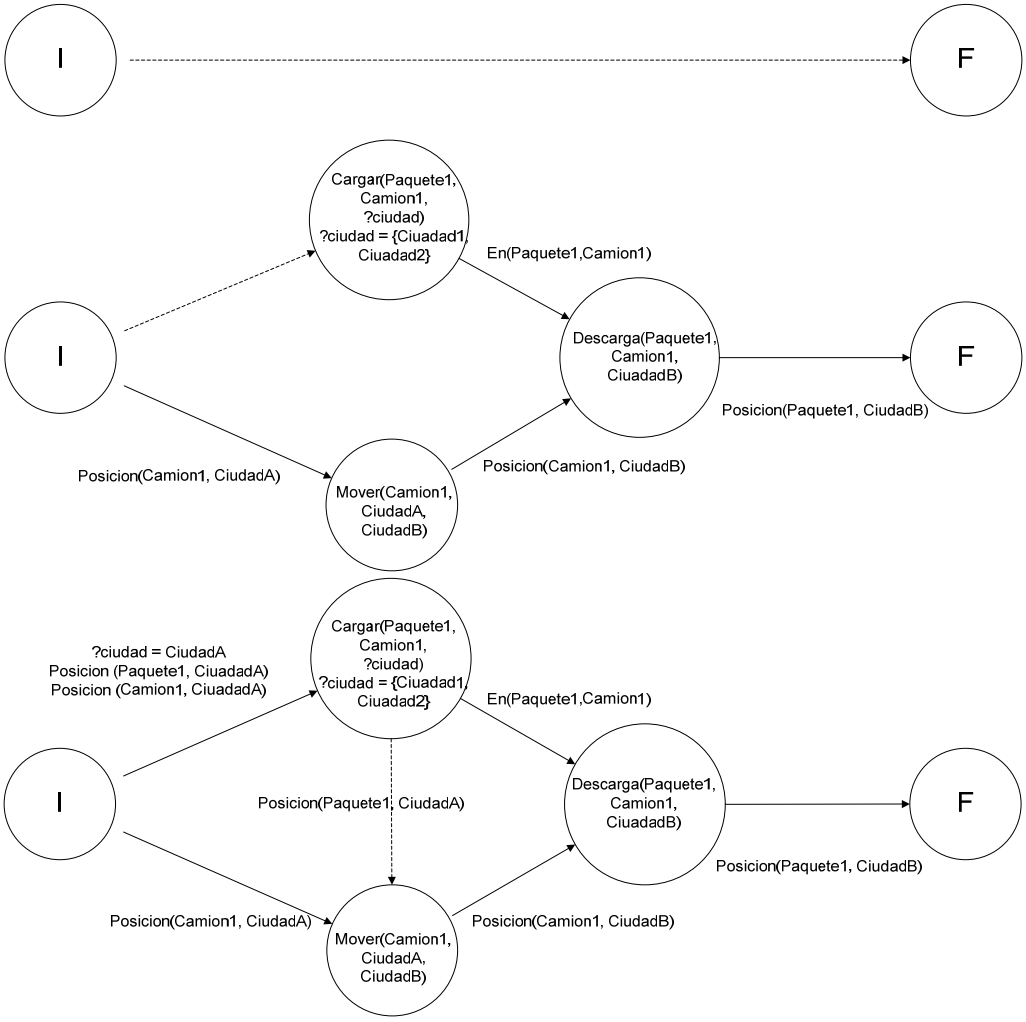


Figura 8. Representaciones de un nodo en un proceso de planificación POP

En la figura 8 se puede ver la representación de varios nodos concretos del espacio de búsqueda en un problema en el que se quiere mover un paquete de una ciudad a otra mediante un camión. El primero nodo representa el nodo raíz del árbol de búsqueda en el que solo existe el paso inicial y el paso final unidos por una restricción de orden. En el segundo nodo ya se han añadido los pasos necesarios para solucionar el problema, no obstante aun no se ha decidido el orden entre dos de ellos ya que en principio ambos se pueden ejecutar desde el principio. Pero existe una amenaza entre ellos ya que el paso mover deshace la posibilidad de cargar ya que el camión no se encontraría en la misma ciudad que el paquete. En el tercer nodo se ve su solución, que consiste en agregar una restricción de orden entre ellos y además es una solución del problema.

4.1.4. PLANIFICACIÓN BASADA EN GRAFOS.

La planificación basada en grafos consiste en la construcción de una estructura bidimensional que representa de forma compacta las diferentes posibilidades que pueden darse durante la búsqueda del plan, para una vez logrados los objetivos extraer el plan de dicha estructura. En dicho grafo se representan tanto las variables, las acciones y las relaciones entre ellas. Un grafo de planificación es una estructura de orden polinómico que modela la arte estática (proposiciones y acciones) y la parte dinámica (relaciones de orden). La única restricción de representación de este grafo es que solo se pueden manejar desde un enfoque proposicional, es decir, literales positivos y deben estar totalmente instanciadas.

Los grafos de planificación se representan mediante niveles, donde cada nivel representa las acciones que pueden ejecutarse con el estado conseguido en el nivel anterior, y el conjunto de proposiciones conseguido por las mismas. Cada nivel se compone de dos sub-niveles correspondientes a las acciones y a las proposiciones. Las relaciones entre los distintos nodos (acciones y proposiciones) pueden ser de tres tipos: aristas-pre, efectos positivos y efectos negativos. Su construcción no conlleva restricciones muy estrictas. En un nivel de acción A[t] aparecen todas las acciones (incluidas unas acciones especiales llamadas no-op cuyo objetivo es mantener las proposiciones ya conseguidas en el siguiente nivel, pero que no aportan nada) cuyas precondiciones se satisfacen en un nivel anterior de proposiciones P[t-1], con independencia de si dichas acciones pueden coexistirla mismo tiempo, o si se pueden ejecutar concurrentemente en el mundo real. En un nivel de proposiciones P[t] aparecen todas las proposiciones que estuvieran ya en P[t-1] (mediante las acciones no-op) y las nuevas proposiciones conseguidas por las acciones del nivel A[t]. Así pues en un nivel de proposiciones no esta representado un estado individual valido sino un conjunto de ellos, los cuales puede que no se satisfagan simultáneamente en el mundo real. Habitualmente se conoce como grafo de planificación relajado a aquel que no contempla los efectos negativos. Aun así, se puede deducir con certeza que si una proposición no existe en un determinado nivel de proposiciones, significa que no existe una combinación posible que genere dicha proposición. Esto es consecuencia directa de las acciones no-op, que mantienen la proposición invariable desde que se consigue hasta el nivel actual.

Estos grafos cumplen además varias propiedades que los hacen muy interesantes para el proceso de planificación:

- Mantienen una noción interna del tiempo mediante los niveles de planificación, y mediante los enlaces entre los nodos se representan implícitamente muchas restricciones entre acciones y preposiciones.

- Es una representación a mitad de camino entre una representación basada en estados y una de orden parcial. No hay un orden de ejecución como tal, salvo el impuesto por los niveles, de tal forma, todas las acciones del nivel t deben ejecutarse antes que las del nivel t+1, pero sin importar el orden entre ellas.
- Se construye con complejidad temporal y espacial polinómica respecto al tamaño del problema

- Si existe un plan valido en el nivel t ya esta representado mediante un subgrafo del mismo.

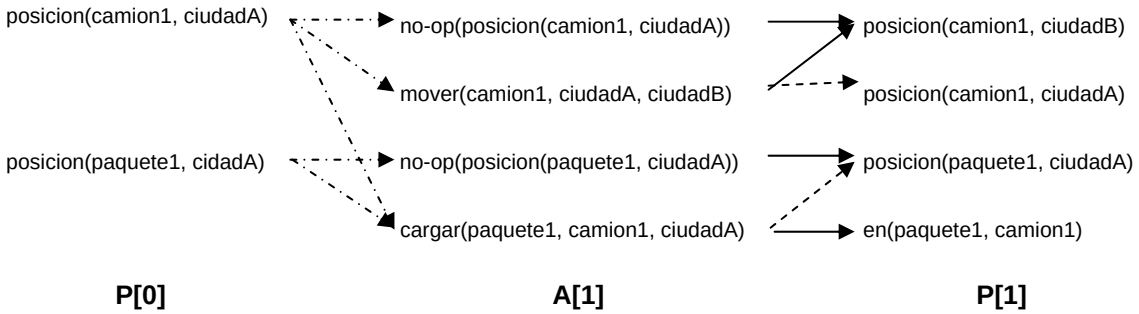


Figura 9. Representación de un grafo de planificación.

En la figura 9 se puede observar una represtación de dicho grafo con sus respectivos niveles y relaciones entre los nodos. Las aristas entre el nivel P[0] y A[1] representan las aristas-pre cuyo significado es que la acción requiere dicha proposición. Entre los sub-niveles A[1] y P[1] las aristas continuas representa los efectos negativos y las aristas discontinuas los efectos negativos. Como se puede observar no es posible que el camión se encuentre en dos lugares a la vez. Es por ello que después de realizar el grafo ha de extraerse el plan de él.

Graphplan es el planificados basado en grafos mas conocido. Introduce algunos cambios en lo explicado anteriormente, añadiendo ciertas restricciones a ese grafo relajado. Debido a que se dan situaciones como la de la figura en la que no pueden darse al mismo tiempo dos proposiciones diferentes es necesario incluir un mecanismo que represente que dichas proposiciones se excluyen mutuamente, además de poder realizar una propagación de dichas exclusiones a los siguientes niveles. A estas relaciones de exclusión se le llaman mutex y pueden darse entre acción-acción y proposición-proposición, Si existe un mutex entre dos acciones significa que no puede existir ningún plan que contenga a ambas en el mismo nivel. Como el cálculo de estas restricciones puede resultar muy costoso, en graphplan se siguen una serie de reglas para realizar la propagación de ellas al mismo tiempo que se genera el grafo. En el caso de las acciones, dos acciones *a* y *b* son mutex cuando: a borra una precondición o efecto positivo de b y cuando una precondición de a y una de b se han marcado como mutuamente excluyentes en el nivel de proposición anterior. En el caso de las proposiciones, dos proposiciones p

y q son mutex si todas las formas de alcanzar p son excluyentes con todas las formas de alcanzar q. Normalmente los mutex entre proposiciones son consecuencia directa de la propagación de los mutex entre acciones. Así cuando dos acciones dejan de ser mutex entre ellas, el mutex entre sus efectos también desaparece porque existe alguna forma de lograrlos todos. Toda la información sobre los mutex se guarda en otra estructura independiente del grafo.

4.1.5. TRATAMIENTO DEL TIEMPO EN LOS MODELOS.

Cuando se intenta representar un problema de planificación mas realista que los descritos anteriormente, normalmente no es aceptable que las acciones sean instantáneas, ya que el proceso de ejecución de dichas acciones puede alargarse en el tiempo, mientras otras se realizan a la vez. Para solventar esta situación se debe relajar alguna de las asunciones del modelo conservativo, para tener en cuenta que las acciones pueden tener distinta duración y es el planificador el que debe gestionar las características temporales de forma explicita. Esta modificación en el planteamiento de los problemas incrementa la complejidad de la resolución de los problemas debido a que además de encontrar una correcta secuencia entre las distintas acciones para alcanzar objetivos se debe tener en cuenta el instante concreto de su ejecución y finalización. La implicación que tiene esta medida en la optimización de los planes es que en lugar de optimizar en función de los pasos de ejecución se debe realizar sobre la duración completa del plan.

Además, como repercusión directa de este cambio, el tratamiento de las precondiciones y efectos también debe cambiar. En el caso de las precondiciones, en este nuevo modelo pueden darse en tres situaciones diferentes, con implicaciones también distintas. Estas situaciones son al inicio de la acción, al finalizar la misma, o durante toda la ejecución de la misma, lo que significa que la condición debe cumplirse antes de empezar. Antes de finalizar o antes de empezar sin que se incumpla hasta la finalización de la acción respectivamente

Si tomamos como ejemplo el caso del camión comentado anteriormente, las diferentes situaciones que podrían ocurrir de las mencionadas son:

- **Duración de las tareas:** un ejemplo de duración de las tareas seria el viaje entre dos puntos, habitualmente no se tarda el mismo tiempo en recorrer 20Km que 50Km, en cuyo primer caso se podrían hacer dos viajes de ese tipo mientras aun esta en proceso el otro.

- **Precondición en el momento inicial:** El camión debe estar cargado antes de marcharse.

- **Precondición antes de finalizar:** El camión tiene gasolina para concluir la acción.

- **Precondición durante la ejecución de la acción:** Si el camión reposta gasolina debe estar parado durante todo el reportaje.

- **Efecto al inicio de la acción:** En el momento que inicia el viaje ya no esta en el origen.

- **Efecto tras finalizar la acción:** En el momento en que finaliza el viaje, esta en el destino.

En un principio la planificación temporal se abordo desde el punto de vista de una planificación de orden parcial, ya que las estrategias de menor compromiso utilizadas en dicho modelo favorecen mucho el tratamiento del tiempo al no hacer la instanciacion hasta que sea estrictamente necesario, con lo que se reduce la complejidad del problema a tratar. No obstante el planteamiento temporal puede introducirse en los otros modelos con planteamientos ligeramente diferentes. En el caso de la planificación basada en estados, dado que cada acción modifica el estado anterior y se genera un nuevo nodo en la búsqueda, la inclusión del tiempo afecta principalmente a la elección de las tareas a realizar en base a la duración total del plan en lugar de al número de tareas. Si se establece además relaciones causales y restricciones de orden entre elementos se puede trabajar con paralelismos entre tareas que no se afecten entre ellas (enviar dos camiones distintos de viaje al mismo tiempo). La planificación de orden parcial es, como se ha dicho, el modelo que intuitivamente mejor realiza el tratamiento temporal, ya que la metodología de este modelo incluye todo el funcionamiento necesario para contemplar las relaciones entre acciones, y mecanismos de ajuste entre ellas. Por ultimo la incorporación del tiempo en el modelo de grafos supone realizar una modificación en el tratamiento de los mutex y restricciones entre las acciones. Además, se debe cambiar el planteamiento de pasos de ejecución en los niveles del grafo, ya que en este momento los niveles representan un tiempo concreto, pudiendo darse que los efectos de una acción terminen más de un nivel después.

4.1.6. ESTRATEGIAS DE BUSQUEDA.

Las estrategias de búsqueda sirven para guiar la misma hacia un camino u otro dependiendo de la estrategia. Esta guía se basa en establecer una puntuación para cada nodo de ABIERTOS y así obtener una ordenación que indique el orden en que deben explorarse. Normalmente esta puntuación se establece mediante la función $f(n) = g(n) + h(n)$, en la que $f(n)$ representa la puntuación asignada al nodo n, $g(n)$ es el coste de la aplicación de las acciones que han conducido hasta este nodo, y $h(n)$ que representa el valor de una función heurística, que no es mas que la estimación del coste desde el nodo actual hasta lograr los objetivos propuestos. Dependiendo de que peso se le otorgue a cada componente de la función, podemos obtener estrategias basadas únicamente en el coste actual ($g(n)$), basadas solo en la función heurística ($h(n)$), o una mezcla de las dos.

Las búsquedas sin información son aquellas en las que solo se tiene en cuenta el valor de $g(n)$, es decir el coste acumulado hasta el momento actual. Entre ellas se incluyen la búsqueda por primero en anchura, primero en profundidad y por coste uniforme. Cada una de ellas se explica a continuación.

Primero en anchura.

El primero de estos métodos es la búsqueda por anchura, en este método cuando se explora un nodo, los nuevos nodos generados se introducen al final de la lista de "abiertos" con lo que dicha lista pasa a funcionar como una cola en la que se da preferencia a los nodos mas antiguos o que lleven mas tiempo en la lista. Con ello se consigue una exploración por niveles en el que el nivel 0 es en nodo raíz, el nivel 1 son los nodos que son sucesores del nivel 0 y así sucesivamente. Por tanto las soluciones aportadas son en función del número de acciones aplicadas ya que en cada nivel incluye una acción más al estado actual. En este método la métrica es obviada en el proceso de búsqueda, tan solo sirve para obtener la calidad de la solución final.

La búsqueda en anchura es completa debido a que no se realiza poda en el proceso de búsqueda pero solo es admisible, es decir encontrara la solución optima, si todas las acciones tienen el mismo coste, sino la solución que se obtenga será mínima en numero de acciones pero no necesariamente en coste. Este tipo de búsqueda no es aplicable a problemas complejos debido a que aumenta de formar exponencial el número de nodos que se tienen en cada nivel de la lista de "abiertos" y por tanto también aumenta exponencialmente el espacio necesario para guardar toda la estructura y el tiempo para examinarla. Haciendo que un problema con una solución en un nivel alto se haga imposible de resolver bien por la capacidad del ordenador o bien por e tiempo requerido para encontrar la solución. Se ha implementado este método de búsqueda en nuestro planificador para comprobar con ejemplos sencillos en los que se pueda realizar la traza a mano el funcionamiento del mismo.

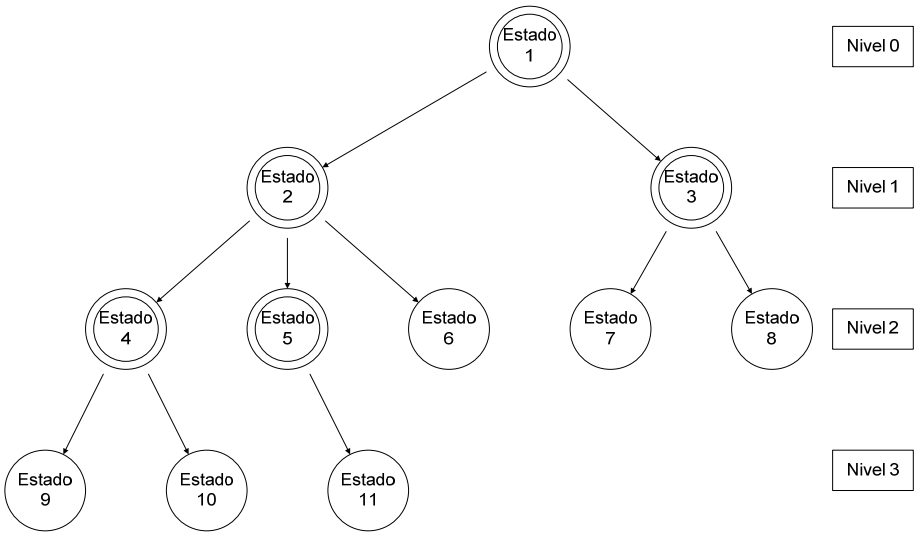


Figura 10. Búsqueda en anchura

En la figura anterior se puede observar el proceso de expansión de los nodos, se han numerado los estados en orden creciente conforme el planificador ha ido creándolos en la exploración, se ve también como se exploran los nodos por niveles.

Primero en profundidad.

La búsqueda por profundidad es el segundo de los métodos comentados en el apartado anterior. En este caso, a diferencia del método de anchura, los nuevos nodos producidos de la exploración del nodo actual se introducen al principio de la lista de "abiertos", es decir, se da prioridad de exploración a los nodos recién llegados, lo que provoca que funcione como una pila. Normalmente esta estrategia de búsqueda no es completa ni admisible porque si la búsqueda entra en una rama infinita puede que no acabe nunca de realizar la búsqueda, no obstante debido a particularidades del dominio sobre el que estamos trabajando como por ejemplo que las acciones siempre tienen un numero máximo de repeticiones, estas ramas de búsqueda ya no pueden convertirse en infinitas ya que al agotar las acciones disponibles en el dominio se obtendría un tope de profundidad, consiguiendo además que se vuelva una estrategia de búsqueda completa y admisible (la admisibilidad se obtendría siempre y cuando se explore todo el espacio de búsqueda). Además en la primera rama de búsqueda ya se obtendría si hay solución o no (debido a que en el dominio todos los efectos son incrementales, si se ejecutan todas las acciones del dominio y no se consigue el objetivo se sabe con seguridad que no se va a conseguir. Por el contrario si hay solución se obtendrá de camino al nodo mas profundo), aunque esta no sea la optima y por tanto continúe sin ser una estrategia admisible. Aun así, al contar con una primera solución se puede actualizar un valor máximo de búsqueda para buscar una solución mejor a un nivel menos profundo. Como en el caso de anchura la solución se basa en el numero de acciones y por tanto puede que el plan con menos acciones no sea necesariamente el optimo porque no se esta teniendo en cuenta la métrica. En esta estrategia normalmente el espacio requerido para guardar la estructura de nodos es mucho menor que la de anchura ya que solo se guarda en memoria la rama en exploración actualmente más los nodos hijos que aun no se han explorado. Pero debido a la gestión que se realiza de nodos repetidos, se guardan en otra lista (la lista de "cerrados") los nodos ya explorados, por tanto no se gana en ocupación de memoria. En el peor de los casos tampoco se gana en coste temporal ya que se exploraría el árbol completo si la solución se encontrase en la última rama o explora el árbol completo de un nivel anterior para garantizar que no hay una solución mejor. Pese a todo se ha implementado en nuestro planificador para comprobar el buen funcionamiento de este y porque es una forma rápida de saber si hay solución o no en el dominio actual.

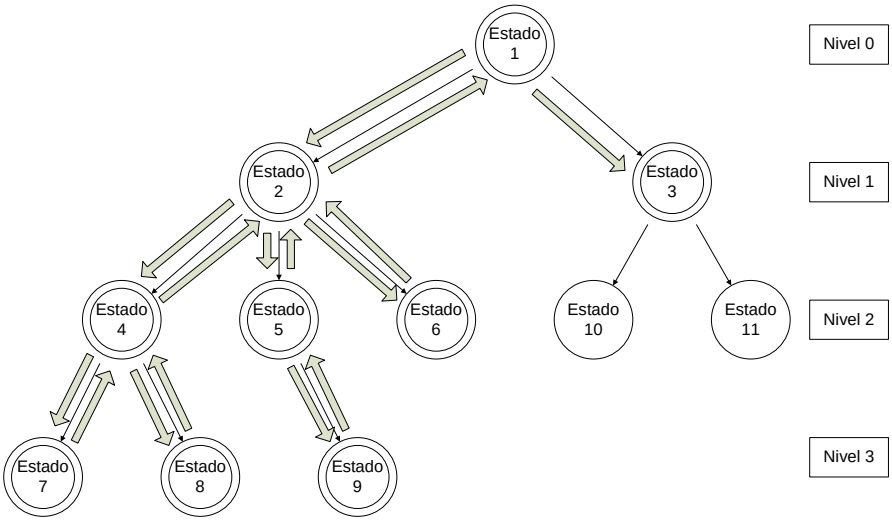


Figura 11.Búsqueda en profundidad

En la figura se puede observar la evolución del proceso de búsqueda realizado en profundidad, por un lado el numero de estado representa el orden de generación de los mismos, es decir al explorar el estado 1 se generaría el estado 2 y 3 y se añadirían al principio de la lista de “abiertos”, a continuación se exploraría el estado 2 y se repetiría el proceso. Siguiendo las flechas sombreadas se puede ver el orden en el que se examinarían los diferentes nodos del grafo. Como se ve, a diferencia de la búsqueda en anchura el salto entre niveles de búsqueda se realiza tan pronto como hay un nodo del siguiente nivel, aunque queden nodos del nivel anterior por explorar.

Coste uniforme.

La búsqueda por coste uniforme es el tercer método de búsqueda de los comentados anteriormente. En este tipo de búsqueda se expande primero el nodo con un menor coste acumulado hasta el momento, y el coste viene determinado por la métrica indicada en el problema en ese nodo. En el caso de que la medica fuese el numero de acciones o similar (tiempo total, y todas las acciones duran lo mismo) esta técnica funciona exactamente como la búsqueda en anchura, de hecho también se realiza una búsqueda por niveles, pero en este caso los niveles vienen determinados por la métrica y no por el numero de acciones introducidas en el plan. De esta forma la búsqueda por coste uniforme sigue siendo completa ya que si hay solución la encontrará, pero seguro que es admisible ya que en caso de encontrar una solución esta será la de menor coste de la métrica y por tanto la optima. Cuando se genera un nuevo nodo se calcula su peso g(n) y se añade a la lista de “abiertos” en la posición correspondiente al peso de ese nodo, así se explorarán primero los caminos que a priori parezcan más favorables porque tienen un menor coste asociado. Pero esto no garantiza de que se encamine la búsqueda por el camino correcto ya que si una acción tiene un coste muy elevado pero es necesaria para la solución se explorarán antes todas las alternativas con coste menor. Este tipo de búsqueda se ha implementado en el planificador para poder disponer de un método de búsqueda sobre el cual poder aplicar una heurística cuyos resultados se puedan combinar con el coste hasta el momento para tener una búsqueda guiada y admisible. Ya que si la heurística funciona sobre la métrica, combinado con el numero de acciones que es el coste de los otros dos métodos vistos anteriormente, al ser dos tipos de unidades totalmente diferentes no se produce una ordenación correcta, en cambio si se suma el valor de la métrica con otro valor de la métrica si es una ordenación fiable del nodo mas prometedor. La heurística empleada se explica en el siguiente apartado.

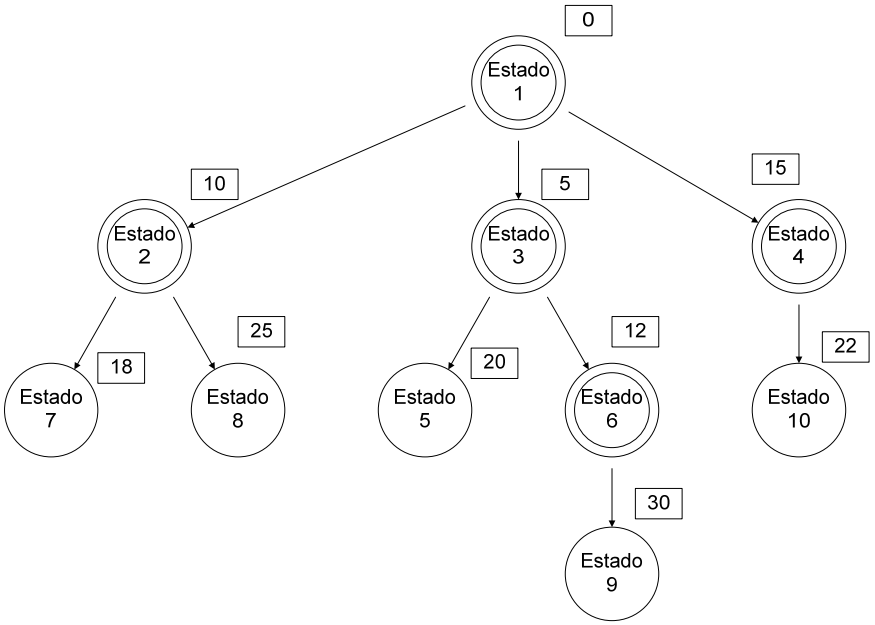


Figura 12. Búsqueda por coste uniforme

En la figura se puede observar la traza de los primeros pasos de una búsqueda por coste uniforme. El numero de estado continua representando el orden en el que se generan los nodos. El orden de exploración viene determinado por el valor indicado en cada recuadro, que representa un valor de la métrica (coste g(n)) hipotéticamente acumulado hasta el momento. A la hora de explorar el siguiente nodo se seleccionaría el que menos valor acumulado tuviese hasta el momento de los estados generados.

Búsquedas basadas en heurísticas.

Cuando se usa una función que realice una estimación del coste de las acciones que quedan para llegar a los objetivos estamos ablando de funciones heurísticas. A diferencia de los métodos sin información, las heurísticas sirven para intentar realizar una búsqueda guiada en la que se exploren antes las zonas del árbol que parezcan más prometedoras antes que las que lo parezcan menos. Una heurística no es más que un conjunto de criterios reglas o métodos para intentar discernir cual es la mejor alternativa entre varias posibles para alcanzar los objetivos. Básicamente es información sobre el problema expresada de forma que el planificador puede entenderla. Las heurísticas, como toda estimación suele basarse en conocimiento impreciso con lo cual puede que haya mejores aproximaciones que otras, cuanto mas conocimiento, mas real será la estimación pero mayor el coste de conseguir dicha estimación. Por lo tanto hay que llegar a un punto de compromiso en el cual se obtenga una buena estimación sin sobrecargar el coste de la búsqueda.

Un ejemplo de búsqueda basada puramente en heurística es el algoritmo de primero el mejor, en el que la lista de ABIERTOS se ordena en función del valor de h(n) para cada nodo. Es decir, siempre se explorará primero el nodo más prometedor de los que haya pendientes. Esta valoración se puede obtener de diversas maneras, normalmente se intenta solucionar una versión mas relajada del problema con el que se este tratando y se toma como valor heurística el coste de la solución del problema relajado.

4.2. DOMINIO Y PROBLEMA EN *E-LEARNING*.

Un problema de planificación se representa mediante un conjunto de operaciones llamadas acciones o tareas, y los estados que se consiguen aplicando dichas acciones. Además, en un problema de planificación se debe especificar el estado inicial y el estado objetivo que se quiere conseguir. Este tipo de problemas se soluciona obteniendo una secuencia de tareas que tras la aplicación de las mismas desde el estado inicial nos conduzcan al estado final. Si pensamos en estos términos, el problema de *e-learning* se puede representar de una forma relativamente parecida, aunque con algunas diferencias. A continuación se expone el tipo de conocimiento que se debe representar tanto en la herramienta de diseño, como en el planificador.

Después de un análisis del problema de *e-learning* se han definido los siguientes elementos: el alumno, el material que se puede utilizar en el curso, los recursos disponibles, las tareas y como afecta su ejecución en el tiempo transcurrido, y conceptos que se consiguen o requieren en las tareas. Concretando un poco más, tenemos:

- **Alumno:** La representación del alumno debe permitir la representación del perfil del estudiante (activo o reflexivo, sensitivo o intuitivo, visual o verbal, inductivo o deductivo, secuencial o global, etc. Desde el punto de vista de la clasificación de Felder [Felder88]), los conocimientos previos que poseyese el estudiante (no todos los estudiantes parten con los mismos conceptos) y los recursos propios que poseyese el alumno (si en el curso se requiere visitar una web, el alumno necesita un ordenador, debe poder representarse si el alumno tiene este recurso ya que sino se lo deberían proporcionar en el curso).

- **Material del curso:** El material proporcionado en el curso como apuntes, videos, etc. Se debe representar mediante su tipo y si necesita otros recursos adicionales. Por ejemplo un archivo en pdf requiere un ordenador para poder visualizarlo.

- **Recursos disponibles:** Aparte de los recursos que posea el propio alumno en el curso se pueden ofrecer ciertos recursos como laboratorios, ordenadores, libros, etc. de los que el alumno puede hacer uso para la realización del curso. Estos recursos se deben representar mediante la capacidad que tienen, la cantidad de la que se dispone en el curso, su coste de utilización, y su disponibilidad (puede que el recurso no este siempre disponible)

- **Tiempo:** La granularidad con que se mida el tiempo de duración de los planes es un factor decisivo en la complejidad del problema, ya que una representación del tiempo de forma realista con fechas y días concretos complicaría demasiado el proceso de búsqueda de una solución. Así pues, se puede representar, por ejemplo, mediante un cálculo de horas en el que una tarea tenga una duración de 1,5 horas y/o un determinado objetivo se debe cumplir antes de 30 horas.

- **Conceptos:** Son la representación de los conocimientos que poseen los alumnos o que puede adquirir, y se representa su valor de “aprendizaje” mediante porcentajes. Los conceptos pueden ser requeridos (un valor concreto) para realizar una determinada tarea, o pueden ser generados por una tarea, además el

estado inicial y final también se representa mediante los porcentajes de los distintos objetivos.

- **Tareas:** Son el medio por el cual un alumno puede incrementar si conocimiento sobre algún concepto determinado. Estas tareas contienen unos requisitos que el alumno debe cumplir (como son el perfil, conocimientos previos, etc.) para poder realizarla. También puede darse el caso que como requisito previo a una tarea se deba haber realizado otra anteriormente (restricción de orden), o disponer de unos materiales concretos para la misma. Además en una tarea se puede indicar que sea colaborativa o no, es decir si se necesita que varios alumnos colaboren con la misma tarea para poder llevarla a cabo.

Con esta representación queda suficientemente concretado el escenario en el cual se debe buscar una solución, la cual debe ser totalmente adaptada al perfil de cada alumno y además debe ser una secuencia que se puede ejecutar con los recursos disponibles en el curso y/o por los alumnos. La herramienta de diseño permite la representación de todo lo dicho anteriormente, en la sección n correspondiente a la misma se explica como se representan dichas características.

La solución debe poder visualizarse a modo de una secuencia de tareas, las cuales están relacionadas por enlaces causales, que conduzca a la consecución del objetivo. En los momentos en los que la decisión del camino a tomar dependa de la ejecución del plan (dependiendo de la nota de un examen) se tendrá en cuenta una opción por defecto para no representar todas las opciones posibles, pero si en el momento de la ejecución no se cumplen las condiciones esperadas se replanificará a partir del momento concreto en el que se encuentre el plan y se mostrara el nuevo camino a seguir. Además debe generarse un plan por cada alumno, es decir, una secuencia de acciones por cada alumno representado en el problema; estas acciones de diferentes alumnos se instanciarán en el tiempo mediante un CSP/Scheduler el cual combinará la ejecución de todos los planes para satisfacer las restricciones temporales impuestas en el problema además de establecer los recursos comunes que utilizará cada alumno en cada momento. Por ejemplo, si una tarea debe realizarse por 5 alumnos, el scheduler instanciará esta tarea concreta de 5 alumnos para que se realice al mismo tiempo, y si es necesario asignará los recursos a utilizar.

El dominio sobre el que trabajamos, contiene muchas de las características de un problema típico en PDDL, aunque por tratarse de un dominio que intenta reflejar las distintas posibilidades que pueden ocurrir en el ámbito de la educación, más concretamente en la educación a distancia, éste tiene una serie de características especiales que delimitan ciertas funcionalidades de PDDL con lo cual, al implementar un planner ad-hoc, no es necesario tenerlas en cuenta.

Características de planificación especificas para el problema de *e-learning*.

1. **Duración de las acciones:** En nuestro caso las acciones que se pueden realizar en el dominio son acciones con una duración determinada, que normalmente no será nula, con lo cual al contar con acciones que no sean instantáneas de debe manejar información temporal, ya que debido a que diversos estudiantes pueden

realizar distintas acciones en un mismo momento, el paralelismo al ejecutar dichas acciones debe tenerse en cuenta.

2. **Información numérica:** Toda la información que se maneja en el dominio, es numérica. Se ha prescindido de los predicados booleanos porque los conocimientos se representan en porcentajes, no si se saben o no.

3. **Monotonía:** En un curso o periodo formativo que se modele en la herramienta para obtener las distintas rutas posibles, hemos supuesto que los conocimientos adquiridos no se olvidan, ya que dentro del mismo periodo se deben tener "frescos" los conocimientos adquiridos durante el mismo porque suelen ser necesarios durante todo el periodo. Por lo tanto, los conocimientos solo se pueden aprender, es decir el nivel del conocimiento solo puede incrementar, lo que produce que no se produzcan efectos delete en el proceso de planificación, cosa que simplifica bastante la tarea de planificar.

4. **Repetición de tareas:** Las tareas se pueden repetir un numero especificado de veces, en el caso de las tareas de refuerzo, tal vez sea necesario realizarlas varias veces para poder seguir avanzando.

5. **Contexto temporal:** En el dominio *e-learning* se considera también que todas las condiciones de la acción se deben cumplir antes de comenzar para poder realizarse, por lo tanto, independientemente del momento indicado en el fichero del dominio, se comprueban como si fueran condiciones "at start". Esto es debido a que cuando se requiere un determinado valor de un concepto, se debe cumplir antes de poder ejecutar la tarea, y además, como hemos dicho anteriormente este valor no puede decrecer durante el transcurso de la misma, por lo tanto se mantendrá la condición valida durante toda la duración de la tarea. También los efectos sufren la misma situación, aunque en este caso los efectos se aplican al finalizar la acción, independientemente del momento indicado, ya que en el caso de ampliar conocimientos se considera que para hacer efectivo el aumento de conocimiento se debe completar la tarea por completo. El único caso en que podría interesar que el efecto se aplicase al comienzo de la acción sería un valor que indicase que la tarea ya se esta realizando para que a partir del momento que empiece la ejecución de la tarea se tuviese ya en cuenta para otras acciones, pero como el paralelismo solo se da entre diferentes alumnos, y estos solo interactuarán a partir de la fase del CSP, y para un mismo alumno las acciones deben tener un orden secuencial, no es importante que el efecto se aplique al principio porque ninguna otra acción podrá requerirlo hasta que acabe la ejecución de la actual.

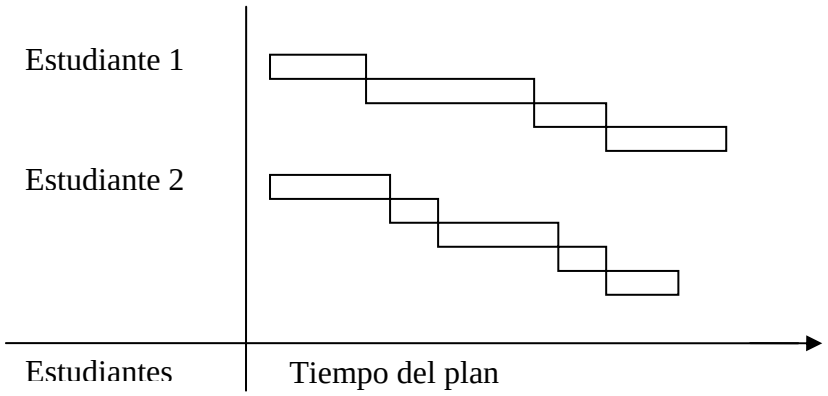


Figura 13. Paralelismo entre estudiantes

Esta figura representa la ruta que deben tomar dos estudiantes distintos, como son estudiantes diferentes pueden tener rutas diferentes para cada uno, tal como se muestra. Si tomamos en cuenta las acciones concernientes a un solo estudiante, podemos comprobar que las acciones no empiezan antes de que termine la anterior, por lo tanto, que se apliquen los efectos al iniciar o al terminar la acción no tiene relevancia sobre el resto de tareas. Si además tenemos en cuenta que la asignación de tareas de un estudiante es independiente de la asignación del otro, sus tareas no pueden interferirse entre ellas. Como se ha dicho anteriormente, si compartieran alguna tarea que requiere que la hagan "al mismo tiempo", el CSP se encargara de estirar los planes para que coincidan.

6. **Tipos de condiciones:** Existen dos tipos de condiciones en el dominio, las condiciones de obligado cumplimiento, que son necesarias para poder ejecutar una tarea, y las recomendaciones. Estas últimas se utilizan para recompensar de alguna forma su cumplimiento, es decir, para potenciar el cumplimiento de determinados objetivos. La forma más usual de buscar esta recompensa, suele ser mediante la inclusión de la misma en la métrica, así cuantas mas condiciones de este tipo se cumplan, suele considerarse una mejor solución.

7. **Paralelismo:** Para el desarrollo del plan, se tiene también en cuenta que las acciones para un mismo estudiante no pueden realizarse en paralelo entre ellas, ya que se considera que un estudiante no puede comenzar una nueva acción mientras no haya acabado la anterior. No obstante para diversos estudiantes sí puede haber paralelismo entre acciones de varios de ellos, para que posteriormente el CSP pueda asignar recursos compartidos al conjunto de estudiantes/tareas.

8. **Sincronización:** Además, existen restricciones de sincronización entre tareas, recursos compartidos, colaboración de varios alumnos en una misma tarea, etc. Estas restricciones no se pueden representar en PDDL directamente, y el planificador desarrollado tampoco tiene en consideración estas restricciones al calcular los diferentes planes. Para resolver esta parte del problema debe resolverse con un CSP, tomando como entrada, planes previamente calculados y el conjunto de restricciones.

El tiempo total del plan se calcula, como ya se ha dicho secuencialmente para cada alumno, lo que significa que el tiempo total de un alumno es la suma de las duraciones de todas las tareas que involucran a ese alumno. Dado que las acciones de diferentes alumnos se pueden realizar en paralelo, para establecer la duración total del conjunto de planes calculados para los distintos alumnos, se considera la duración del plan individual más largo, ya que mientras se realizan las acciones del mismo, las acciones de los otros planes también se pueden ejecutar y terminaran antes que la finalización de éste. Este tiempo, que se considera definitivo en la fase de planning podrá ser modificado por el CSP si una vez tenidas en cuenta las restricciones sobre recursos compartidos, es necesario alargar el plan. Aunque el tiempo calculado en la fase de planning siempre será una cota mínima.

4.2.1. REPRESENTACION EN PDDL.

Los escenarios de planificación habitualmente se representan en PDDL, que como se ha dicho se ha convertido en un estándar de representación de este tipo de problemas. Este lenguaje se basa en la representación de un conjunto de variables las cuales representan los diferentes estados del problema y un conjunto de operadores que pueden actuar sobre las mismas para llegar mediante su actuación a una solución viable al problema planteado.

Para aportar toda la expresividad de la que se ha hablado anteriormente, en PDDL se sigue una estructura que diferencia el entorno en el que se producirán los cambios, es decir la definición de las variables y los operadores que pueden interactuar con que variables. El problema en concreto que se quiere solucionar, esto es el estado inicial del problema y el objetivo que se quiere lograr. Y además para controlar la calidad de la solución que aporta el planificador se establece una métrica, que no es más que una formula con la que se establece esta calidad. Esta representación se divide en dos ficheros, correspondientes al dominio y al problema.

A continuación se detalla la representación de dicha estructura en lo referente a la parte de PDDL usada en este proyecto, ya que el lenguaje permite representaciones más extensas y complejas que no se usan aquí. Para esto vamos a seguir un ejemplo en el cual un estudiante debe realizar dos tareas para lograr los conocimientos necesarios para superar un curso.

Declaración de variables.

La declaración de las variables que definen el entorno se realiza en el fichero de dominio, es en este fichero donde mediante la instrucción `:functions` se declaran todas las variables que estén involucradas en el problema representado. En los problemas de *e-learning* existen dos tipos de variables a representar, los conceptos que definen el grado de conocimiento sobre el mismo y otras que indican si se ha realizado o no una tarea determinada.

Siguiendo el ejemplo planteado tenemos los conceptos cuyo valor se representa en porcentaje, y el valor de la tarea hecha, donde un 0 implica que no se ha realizado y un 1 que si. La representación seria la siguiente:

```
(:functions
   (Learning_Value_LRN1_Concept0)
   (Learning_Value_LRN1_Concept1)
   (Learning_Value_LRN1_Concept2)
   (Task1_LRN1_DONE)
   (Task2_LRN2_DONE)
   (BONUS)
)
```

Estado inicial del problema.

En el estado inicial del problema es donde se establece que valores toman las variables antes de comenzar la búsqueda, por ejemplo las tareas no estarán realizadas y puede que se tengan unos conocimientos previos. Esta inicialización se realiza mediante una asignación en el fichero de problema mediante la instrucción `:init`.

```
(:init
  (= (Learning_Value_LRN1_Concept0) 100)
  (= (Learning_Value_LRN1_Concept1) 0)
  (= (Learning_Value_LRN1_Concept2) 0)
  (= (Task1_LRN1_DONE) 0)
  (= (Task2_LRN2_DONE) 0)
)
```

Esta inicialización representa que no se ha realizado ninguna tarea aún, pero que ya se tiene un conocimiento del 100% del Concepto0

Estado final del problema.

En el estado final se definen las condiciones cuya satisfacción se toma como una solución al problema. La definición de estas condiciones se realiza en el fichero de problema mediante la instrucción `:goal`.

```
(:goal
(and
   (>= (Learning_Value_LRN1_Concept1) 100)
   (>= (Learning_Value_LRN1_Concept2) 100)
)
)
```

Tomamos como condición que el alumno consiga el 100% de conocimiento de los conceptos 1 y 2. Como se puede observar, se puede establecer una jerarquía mediante un árbol AND/OR las diferentes combinaciones que se puedan requerir en los problemas.

Métrica.

La métrica es la forma de definir el criterio de calidad que tendrá el plan solución, esta se define también en el fichero de problema. Este criterio se especifica mediante una formula en la notación de PDDL que habrá de minimizar o maximizar.

Habitualmente se busca minimizar el tiempo del plan, maximizar las bonificaciones o una combinación de ambas. Para definir la métrica se utiliza la instrucción `:metric`.

```
(:metric minimize
  (+
    (*
      (-1)(Bonus)
    )
    (TOTAL-TIME)
  )
)
```

En este caso se quiere minimizar el tiempo a la vez que se maximiza el bonus en proporciones iguales.

Operadores.

Los operadores son la parte esencial de un problema de planificación, ya que es mediante ellos la forma de producir cambios en las variables del problema. Los operadores se llaman acciones, o en el caso de planificación temporal *durative-actions*. Estas acciones se definen en el fichero de dominio, y deben contener la especificación de cómo actuar con la variable, es decir precondiciones y efectos, aunque puede que no tenga alguno de los dos. Además también se debe establecer su duración. Ejemplos de acciones serian:

```
(:durative-action Task1_LRN1
   :parameters ()
   :duration (= ?duration 1)
   :condition
     (and
       (at start (= (Task1_LRN1_DONE) 0))
     )
   :effect
     (and
       (at end (increase (Learning_Value_LRN1_Concept1) 50))
       (at end (increase (Task1_LRN1_DONE) 1))
     )
)

(:durative-action Task2_LRN1
   :parameters ()
   :duration (= ?duration 1)
   :condition
     (and
       (at start (= (Task2_LRN1_DONE) 0))
       (at start (>= (Learning_Value_LRN1_Concept1) 50))
       (at start (>= (Learning_Value_LRN1_Concept0) 100))
     )
   :effect
     (and
       (at end (increase (Learning_Value_LRN1_Concept2) 100))
       (at end (increase (BONUS) 1))
       (at end (increase (Task2_LRN1_DONE) 1))
     )
)
```

El conjunto de acciones que se definan para un escenario debe poder llegar a resolver el problema.

Al final del proceso de definición del escenario se tienen dos ficheros con la siguiente estructura:

```
(define (domain nombre_dominio)

(:requirements :typing :fluents
:durative-actions)
(:predicates)
(:functions
   ...
)
(:durative-action nombre_accion
   ...
)
   ...
(:durative-action nombre_accion
   ...
)
)
```

Fichero de dominio

```
(define (problem nombre_problema)
(:domain nombre_dominio)
(:init
   ...
)
(:goal
   ...
)
)
(:metric
   ...
)
)
```

Fichero de problema

4.3. IMPLEMENTACIÓN.

Para resolver los problemas que se definan sobre este dominio, se ha implementado un planificador ad-hoc, debido a que los planificadores genéricos que podemos encontrar, o bien carecen de una forma sencilla de adaptación a nuestro problema, por ejemplo con la inclusión de nuevas heurísticas que encaminen en proceso de búsqueda hacia una solución mas rápidamente, o bien por el hecho de que al intentar resolver problemas de cierta complejidad, son incapaces de devolver una solución.

4.3.1. REPRESENTACION DEL ESCENARIO.

Al igual que en PDDL, en nuestro planificador también se debe poder representar todo lo necesario para que el problema quede bien definido. Dado que para la programación se ha utilizado el lenguaje ©# y no existen librerías para representar las características de los problemas de planificación se ha implementado una estructura de clases que corresponde con las definiciones realizadas en PDDL.

Para representar las variables o *functions*, como se llaman en planificación, existe una clase `Function` cuya misión es representar estas variables. En el planificador se guarda una lista de estas functions para tener una referencia de las variables declaradas en el dominio planteado. Axial pues, a esta lista se puede acceder para comprobar si la variable existe o no, pero en los casos donde se le asigne un valor se mantendrá con un soporte adicional.

Siguiendo el mismo orden en el que se ha explicado la representación en el modelo de PDDL, ahora es el momento de definir el estado inicial. Para ello la clase

`InitState` mantiene una referencia a la lista de variables y crea una lista paralela con los valores asignados a cada una de ellas, es decir para una posición determinada, una de las listar aporta el nombre de la variable, y la otra el valor. Esto es así porque en lugar de realizar una copia de cada variable cuando se crea el valor inicial que podría llevar a inconsistencias entre las diferentes listas de variables, aquí solo se debe mantener los valores de las mismas ya que las definiciones están como referencia para todo el entorno.

El estado final del problema se define ligeramente diferente ya que se deben representar además del valor de las variables, un operador de comparación con respecto a ese valor. Estas condiciones de finalización pueden establecerse en forma de árbol, para ello la clase `GoalState` que representa el estado final esta compuesta de la clase `ConditionTreeNode` la cual representa un nodo de dicho árbol de condiciones.. Cada nodo de este árbol puede estar representado bien por condiciones simple (que se explicara junto a las acciones), por otros nodos, o combinaciones de ambos. Para establecer la relación entre los distintos componentes se debe añadir un operador lógico de tipo AND/OR que determine si se deben cumplir todas/alguna de las condiciones respectivamente.

La métrica se representa de una forma similar al estado final, la formula utilizada para obtener la calidad se representa mediante un árbol de nodos de la clase `OperationTreeNode`, cuyo contenido es un operador matemático y una lista de operadores que pueden ser a su vez otros nodos, el valor de variables de las declaradas en la lista de *functions* o valores numéricos.

Por ultimo para completar la representación hacen falta las acciones que están definidas por la clase `DurativeAction`. En ella se representan el nombre, duración, precondiciones y efectos tal como se define en PDDL. Mientras que el nombre y la duración son datos elementales las precondiciones y efectos tienen un tratamiento de árbol similar al descrito en los casos anteriores con alguna particularidad. El árbol de condiciones se representa con la misma clase que el del estado objetivo, con la clase `ConditionTreeNode`. La diferencia estriba en que las condiciones de una acción además del comparador, la variable y el valor, necesita un momento de aplicación (al inicio, al final…). Por ello la clase `Condition` representa los cuatro datos enumerados, aunque en el caso de una condición de finalización el campo “momento” estará vacío. El caso de los efectos es idéntico con las clases `EffectTreeNode` y `Effect`, con la salvedad de que en lugar de un comparador tenemos una operación que afecta a esa variable (*increase/decrease*).

A todo esto hay que añadirle la representación de ciertos datos que no soporta PDDL pero que nuestra aplicación debe considerar para una mayor flexibilidad de representación y una correcta comunicación con el resto de módulos del proyecto. Estos metadatos que se pasan en forma de comentarios en el fichero de PDDL representan por un lado Información a las acciones (estudiante, duración mínima, duración máxima, número de estudiantes simultáneos, perfil de los estudiantes, etc.) y por el otro, información sobre el dominio como son los recursos disponibles.

4.3.2. PARSER.

El planificador cuenta con su propio parser, también implementado para la ocasión, que lee PDDL *groundeado* (termino usado habitualmente en entornos de planificación para indicar que todas las variables están instanciadas), esto es necesario ya que al estar toda la herramienta escrita en C# no existe ningún parser que se pueda utilizar. El parser lee PDDL *groundeado*, debido a que la herramienta grafica para el diseño de problemas genera más fácilmente un PDDL totalmente instanciado que uno en forma de predicados, además la instanciacion de la herramienta grafica se realiza de forma más rápida que el groundeado de un PDDL estándar ya que la representación lo permite. El parser esta definido el la clase `Parser`, la cual establece un flujo de lectura con los ficheros seleccionados y devuelve los datos ya convertidos. La gramática aceptada por el parser se encuentra en el ANEXO I.

4.3.3. PLANIFICADOR.

El planificador utiliza el modelo de planificación basada en estados ya que las características descritas anteriormente de este dominio hacen que en este modelo resulte sencilla su representación y control. Aunque el método de planificación se ha basado en este modelo, se ha modificado ligeramente la forma de proceder ya que se tienen en cuenta características como la paralelización entre alumnos, enlaces causales y relaciones de orden entre acciones, etc. En nuestro planificador se han implementado cuatro métodos de búsqueda que son: anchura, profundidad, coste uniforme y heurística. Funcionando las dos ultimas sobre la métrica y las dos primeras sobre el número de acciones. Aunque la principal funcionalidad la proporciona la búsqueda heurística, las otras se han implementado para tener una referencia en cuanto a costes de búsqueda. También se ha implementado la búsqueda hacia delante y hacia atrás por los mismos motivos que los métodos de búsqueda.

Representación del estado.

Tal como se describe en el modelo de planificación basado en estados, la búsqueda se realiza mediante un árbol de búsqueda en el que cada nodo representa un estado completo del problema.

Cada nodo creado contiene, independientemente de la dirección en la que se esté buscando (*forward* o *backward*), una referencia al listado de las acciones del dominio, la representación del estado inicial o actual, que se usa en algunos casos para la heurística, y los objetivos que se desean alcanzar. Si se trata de un proceso de búsqueda hacia delante, en los nodos se representan el estado actual del problema y el estado final, es decir conforme avanza la búsqueda el estado actual se va modificando pero se mantienen los objetivos tal cual. Si se trata de un proceso de búsqueda hacia atrás, los nodos guardarán el estado inicial del problema y los objetivos que deben satisfacerse en el estado actual del problema, es decir el estado inicial queda invariable mientras se van modificando los objetivos que quedan por cumplir en el momento actual.

Otras características que se guardan en cada nodo para representar el estado actual de la búsqueda son el valor de la métrica, el f(n) o coste del nodo, las acciones

introducidas en el plan hasta el momento, y según se van introduciendo se crea una lista de *causalLinks* abiertos en los cuales se indica que una acción produce o requiere una determinada variable. También se añade a otra lista diferente los *causalLinks* cerrados que son los que indican que una acción produce un efecto cuya variable necesita otra acción. Esto se explica mas adelante en el proceso de añadir una nueva acción al nodo.

Para diferenciar unos de otros y evitar duplicados, cuentan con un identificador univoco que viene determinado por las acciones que se han incluido hasta ahora en el plan. Para construir este identificador se toma como referencia el listado de acciones posibles del dominio y según si están o no en el plan se introduce un 1 o un 0 a un string que se toma como identificador, quedando un identificador de la siguiente forma "001001010100110". Esto es posible dado que el orden en que se consigan las cosas no es importante mientras se llegue al objetivo, es decir, da lo mismo primero realizar la tarea A y luego la B que al revés, siempre y cuando no haya restricciones de orden entre ellas que limitaría una de las posibilidades y por tanto no podría llegar al mismo estado.

El proceso de búsqueda.

En la figura 14 se puede observar el pseudo código básico con el que se realiza el proceso de búsqueda en el planificador que corresponde con el modelo de planificación basado en estados. No obstante se han realizado mejoras en la eficiencia de la búsqueda que lo modifican ligeramente y que se explican en los siguientes párrafos.

```
-Mientas haya nodos en la lista de "abiertos".
   -Se extrae el primero nodo de la lista.
   -Si es solución y el plan es el mejor hasta el momento.
      -se añade al listado de planes.
   -Si no es solución.
      -Para cada acción que se pueda ejecutar en el estado actual.
         -Se crea un nuevo nodo con el estado actual más la nueva acción y se añade a
la lista de "abiertos".
-Si hay algún plan en el listado de planes, se devuelve true, sino, false.
```

Figura 14. Pseudo-código del proceso de búsqueda.

Para hacer que el proceso de búsqueda sea independiente de la dirección en la que se este buscando se ha modificado ligeramente la forma en que se exploran los nodos hacia atrás. En lugar de escoger un objetivo determinado y aplicar las acciones que aporten algo a dicho objetivo, se aplican todas las acciones que aporten algo al estado objetivo, independientemente del objetivo que satisfagan. Con esto se consigue que el proceso de búsqueda sea el mismo independientemente de la direccionalidad ya que la búsqueda se resume en escoger el primero nodo de la lista de abiertos y explorarlo. Esta exploración es la misma en una dirección u otra salvo que hacia delante se comprueba que se cumplan las condiciones de la acción y hacia atrás se comprueba que tenga en sus efectos alguna variable que se requiera como objetivo. Esto es posible porque encada nodo estamos representando un estado concreto y unos objetivos a cumplir.

Otras modificaciones realizadas a la búsqueda básica en nuestro planificador ha sido la introducción de la gestión de nodos repetidos, así cuando llega a un mismo estado que ya se ha explorado no es necesario volver a repetir toda la búsqueda que parte de ese nodo porque ya se ha buscado anteriormente. Así en vez de desechar un nodo cuando ya esta explorado, se añade a una lista de "cerrados" en la que se encuentran todos los nodos ya cerrados. Esta inclusión de la lista de "cerrados" añade un gasto de memoria razonable ya que se guardan todos los nodos explorados previamente, pero aun así los beneficios aportados por el ahorro de tiempo de búsqueda al no explorar caminos repetidos compensa dicho gasto. La comprobación de si un nodo se encuentra ya explorado se realiza justo antes de comprobar si es solución una vez ha sido recuperado de la lista de "abiertos". La identificación de cada estado o nodo se explica en el apartado anterior.

En cuanto a implementación, los nodos del árbol de búsqueda están representados por la clase `SearchNode`, la cual incluye todos los métodos necesarios para realizar correctamente la búsqueda. El proceso de creación de nuevos nodos, salvo en el caso del nodo inicial, se realiza mediante un proceso de clonado en el cual se replican todas las variables que sean susceptibles de modificación y se pasa una referencia de las que sean de solo consulta. Así, el nodo original se mantiene sin modificaciones mientras que se dispone de un nuevo nodo para seguir trabajando con el.

Una vez creado el nuevo nodo es el momento de añadir una nueva acción para que el estado pueda evolucionar, para ello esta adición se produce en dos fases, en la primera de ellas se tiene en cuenta si el modo de búsqueda funciona en modo *forward* o modo *backward*, ya que según el modo se actualiza el estado actual (*forward*) o se añaden objetivos al goal (*backward*). En la segunda fase se recalcula el peso del nodo en función del algoritmo de búsqueda seleccionado (anchura, profundidad, métrica o métrica con heurística). Siendo posible en el caso de la heurística que se obtenga una solución mejor que la actual, tal y como se explica en el siguiente apartado.

Heurística diseñada.

La heurística diseñada para este tipo de dominios esta basada en una búsqueda en grafos de planificación. Para obtener un plan de una forma más rápida que una búsqueda normal se relaja el problema sobre el que estamos trabajando quitando la secuencialidad de las acciones, es decir todas las acciones se pueden ejecutar en paralelo mientras se cumplan sus condiciones. Además las acciones se aplicaran tan pronto se cumplan sus condiciones. Con estas modificaciones sobre el problema original se puede generar un grafo de planificación muy rápidamente en el cual se obtiene una solución al problema planteado, o bien se comprueba igual de rápido que el problema planteado no tiene solución.

El grafo esta compuesto de varios niveles que vienen determinados por valores de la métrica y en cada uno de ellos se representan todas las variables existentes en el problema. Si solo estuviese esto no seria mas que un listado de niveles así que para construir el grafo se examina el nivel actual y para cada acción que se pueda aplicar se genera una flecha (enlace) que parte del nivel actual y termina en el nivel que devuelva un calculo de la métrica de un nuevo nodo contando las acciones que se han ejecutado

hasta este nivel mas la que acabamos de ejecutar. Es decir si partimos del nivel 0 de métrica y una vez se aplica la acción el nodo tiene un valor de métrica 10, se generaría una flecha que parte del nivel 0 y acaba en el nivel 10. Estos enlaces se guardan en una colección indicando también si se han aplicado o no ya que si el efecto se produce en un nivel que aun no se ha explorado aun esta pendiente de aplicar.

Una vez se generan todos los efectos de todas las acciones que se pueden aplicar en el nivel actual se pasa a buscar el siguiente nivel mirando de los efectos que queden pendientes de aplicar cual es el que tiene un menor nivel, aquí se pueden dar tres situaciones, que el efecto acabe en un nivel inferior a la métrica (mejora la solución), que acabe en el mismo nivel (no aporta nada) y que acabe en un nivel superior a la métrica (empeora la solución). El primer caso puede ocurrir cuando en la métrica se están buscando objetivos en direcciones contrapuestas, es decir se esta intentando maximizar un objetivo y minimizar otro a la vez (en el caso de minimizar el tiempo total y maximizar la bonificaciones) en cuyo caso se toma como valor de finalización el nivel actual mas un determinado *Epsilon* para evitar repercusiones sobre los efectos conseguidos hasta el momento. En el segundo caso también se considera que el siguiente nivel es volver a explorar el nivel actual añadiéndole un *Epsilon* para que posteriormente se conserven las relaciones de orden entre las distintas acciones. Para ello se actualizan los valores de las variables afectadas por los efectos que terminen en el nivel a explorar y se vuelve a explorar el grafo como si fuera la primera vez que se explora tomando el nivel actual como nivel inicial. Por ultimo en el caso de que sea un nivel superior al se explora el nuevo nivel siguiendo el mismo procedimiento. Siempre que se genera un nuevo nivel del grafo se toman todos los valores del nivel anterior y se actualizan con los efectos que terminen en ese nivel. Con este planteamiento se consigue tener en cada nivel del grafo el valor máximo de cada function alcanzado hasta ese nivel además de tener en otra colección las acciones que han producido dichos resultados.

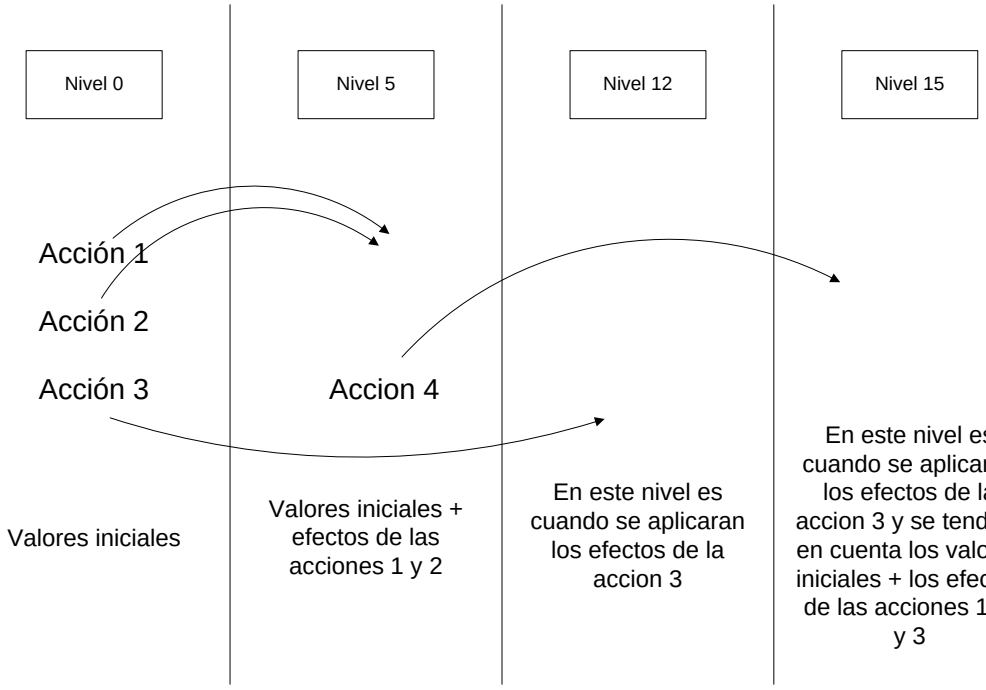


Figura 15. Ejemplo de creación del grafo

La búsqueda puede parar por dos situaciones diferentes, la primera de ellas es que se hayan explorado todos los niveles y no queden mas acciones que se puedan aplicar (se han aplicado todas, o las que quedan no cumplen sus condiciones) por tanto no existe solución al problema planteado. O por el contrario se obtiene un nivel en el que se cumplan todos los objetivos, en cuyo caso se para la generación del grafo y se procede a buscar la solución dentro del mismo. Esta solución se calcula tomando cada objetivo de los propuestos y busca el menor nivel en el que se ha satisfecho dicho objetivo, si ya se satisfacía en el nivel inicial se pasa al siguiente objetivo pero sino se busca en la lista de efectos que acción o acciones han producido que se consiga ese efecto en ese nivel y se busca el nivel en que se cumplieron las condiciones de dicha acción y así recursivamente. Una vez se ha obtenido de forma recursiva un conjunto de acciones que generan el estado en el que se cumplen todos los objetivos tenemos un conjunto de acciones que son solución al problema sobre el que estamos trabajando. Y que cuentan además de restricciones de orden entre ellas.

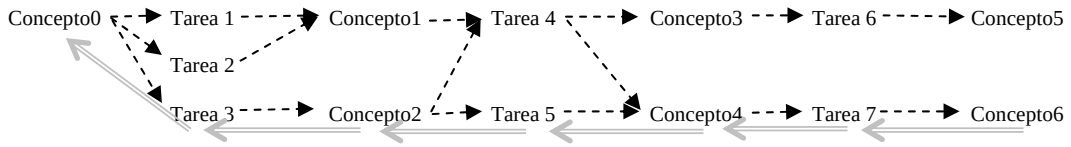


Figura 16. Obtención del plan dentro del grafo

Ejemplo: En un problema determinado cuyo objetivo era conseguir el Concepto C6 se ha generado el grafo de la figura 16. Una vez generado el grafo, empieza la búsqueda del camino más “eficiente”, según la métrica, que consiga dicho objetivo. Empezando por C6, se ve que solo hay una tarea para conseguir dicho concepto, por tanto T7 estará en el plan. A partir de esta acción, se lanzan tantos procesos de búsqueda como conceptos haya en las precondiciones que tenga dicha tarea. En este caso solamente C4, pero resulta que C4 puede conseguirse de dos modos distintos (T4 y T5) por lo que aquí habrá una bifurcación en el proceso de búsqueda en el que cada acción debería resolverse por si misma y la que menor coste en métrica lleve asociado será la seleccionada. Por ejemplo, para conseguir que se ejecute la tarea T4 son necesarias las tareas T1, T2 y T3, mientras que para poder ejecutar la tarea T5 solo es necesaria la tarea T3. Si consideramos que la métrica es el número de acciones, la tarea elegida seria la tarea T5. Con lo que T5 formaría parte del plan. Además como en el subproceso de búsqueda ya se ha encontrado el camino minio para llegar a todas las acciones precedentes a dicha acción, en el momento que se elige esa acción se hereda su camino mínimo y por tanto todas las acciones necesarias para llegar a ese punto. Quedando al final de este ejemplo como el marcado en la figura, el cual representa el plan completo.

Esta heurística toma un estado inicial y un estado final para acotar su búsqueda, en el caso de una búsqueda hacia delante el caso inicial será el estado que se esté explorando actualmente y el estado final siempre será los objetivos del problema. En cambio si se esta realizando una búsqueda hacia atrás el estado inicial siempre será el estado inicial del problema y los objetivos serán el estado objetivo que estemos explorando actualmente. Con lo que como solución de la heurística no se obtendrá un valor sino el conjunto de acciones que sumado a las que ya se han introducido en el plan consiguen una solución al plan. Esta heurística funciona conjuntamente con el método de búsqueda por métrica así que para calcular el peso de cada nuevo nodo se genera un plan secuencializando las acciones que haya devuelto la heurística para un mismo

alumno y se calcula el valor de la métrica que se aplica como peso del nodo que se acaba de generar. Dicho de otro modo, cuando realizamos la búsqueda con este método la función f(n) = g(n) + h(n) no se calcula de la forma habitual que seria el g(n) el peso acumulado hasta el momento y la h(n) el valor devuelto por la métrica. Sino que la propia heurística devuelve un valor f(n) calculado conjuntamente por las acciones que ya estuvieran en el plan mas las devueltas por la heurística. Esto es así, ya que al utilizar el valor de la métrica como "coste" se vuelve imposible discernir lo aportado por cada conjunto de acciones debido a los semi-paralelismos. En el siguiente ejemplo se explica claramente porque no se puede obtener como dos valores separados.

Supongamos un plan para dos estudiantes de cuatro acciones, dos para cada uno. Si utilizamos como métrica el numero de acciones teóricamente el coste del plan seria cuatro (uno por cada acción), pero si tenemos en cuenta que las acciones de diferentes estudiantes se pueden ejecutar en paralelo, el coste del plan vendrá determinado por el sub-plan mas largo correspondiente a un alumno, con lo que el coste real el plan seria 2. Si en un momento determinado de la búsqueda resulta que se han añadido ya al plan las dos acciones del primer estudiante el g(n) acumulado seria dos, y el h(n) (las acciones que quedan por incluir) seria dos. Por tanto f(n) = g(n) + h(n) = 2 + 2 = 4, que es un calculo erróneo. Por eso para calcular la f(n) correspondiente al nodo actual se instancia un plan con las acciones ya incluidas mas las devueltas por la heurística y se calcula el valor de la métrica de ese plan que se devuelve como f(n), en este caso 2.

Tras cada ejecución de la heurística siempre se consigue un plan completo lo cual sirve para ya en el primero nodo saber si no hay solución o contar con una solución de referencia para acotar la búsqueda de un plan mejor. Además como con cada nuevo nodo generado se vuelve a calcular un plan, si este resulta mejor que el almacenado hasta el momento también se devuelve como una mejor al plan anterior. Con esto se consigue una búsqueda guiada en la que se cuenta con una buena solución en muy poco tiempo y que se puede refinar dejando el proceso de búsqueda durante más tiempo. Pero aunque se encuentre la solución optima en muy poco tiempo solo se tiene garantía de que es optima cuando se haya explorado a todo el espacio de búsqueda lo cual tiene el mismo coste que una búsqueda por coste uniforme.

Estos planes se guardan en una lista formada por instancias de la clase Plan, en la que se guardan además de las acciones, valores como la duración total, valor de la métrica, nodos explorados… Estos planes son los que posteriormente se exportan mediante XML para que el CSP pueda trabajar con los resultados.

5. EXPERIMENTOS.

5.1. PRUEBAS DE RENDIMIENTO.

Para probar el rendimiento del planificador desarrollado y compararlo con otro planificador estándar se ha desarrollado el escenario de la figura 17, el cual consiste en un dominio en el que todas las acciones consiguen el 100% del concepto que llevan asociado y tienen una duración de una unidad. Además cada concepto se puede conseguir igualmente por todas las tareas relacionadas con él. Con ello se consigue un dominio en el que cualquier solución encontrada sea la óptima y por tanto se puedan

comparar en igualdad de condiciones nuestro planificador al que llamaremos LNRPlanner y MIPS-XXL [EDELKAMP06] que son los dos planificadores sobre los que se han realizado las pruebas.

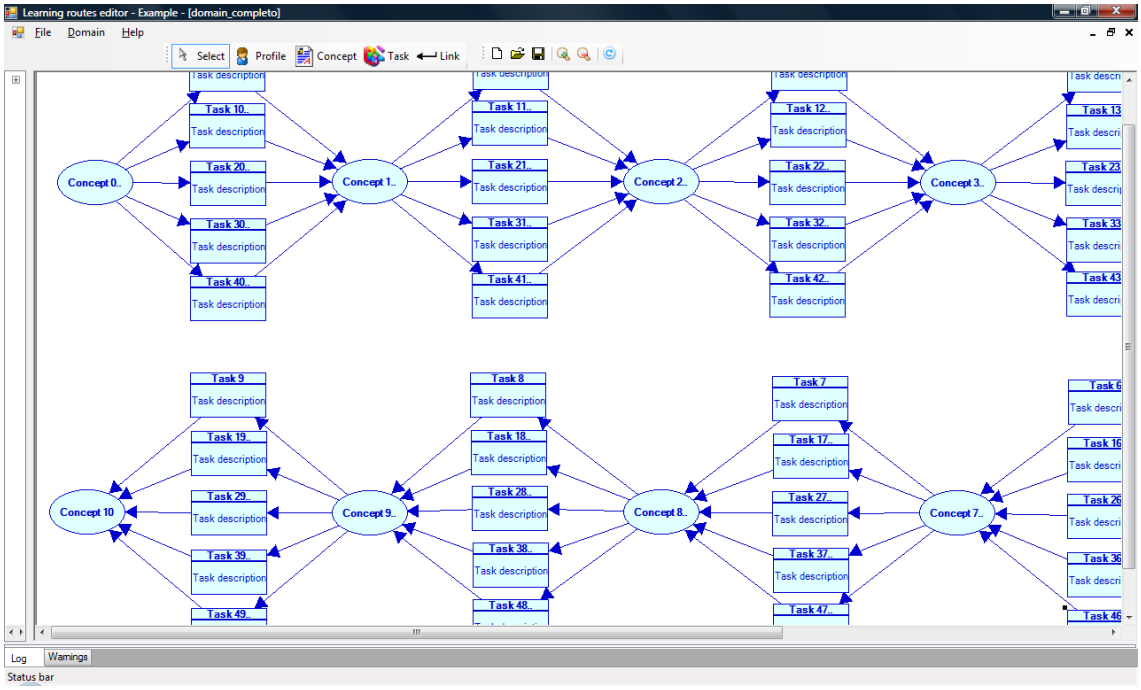


Figura 17. Esquema del dominio diseñado.

Este experimento se ha llevado a cabo en base a tres factores que pueden definir la complejidad del problema. El primero de ellos es el número de estudiantes, ya que conforme aumenta su número es más costoso y complicado encontrar una solución. El segundo factor tenido en cuenta (al que llamaremos factor r) es el número de acciones que pueden conseguir cada concepto, este factor influye directamente sobre la complejidad a la hora de decantarse por una acción u otra al buscar las soluciones. El ultimo factor es la profundidad de la solución, la cual al incrementarse afecta en gran medida a la explosión combinatoria producida en el espacio de búsqueda. En la figura se muestra un factor r de 5 acciones, las pruebas llevadas a cabo van desde un factor r = 1, hasta el que se muestra en la figura. Así mismo la profundidad mostrada en la figura es de 10 (hacen falta 10 tareas para llegar al concepto 10) y se han realizado las pruebas con profundidades desde 2 hasta 10 en escalones de dos en dos. El número de estudiantes tratado ha sido de 1, 2, 5, 10, 20, 50, 100 quedando algunos casos sin llegar a obtener una solución debido al tiempo de respuesta de los planificadores, que superaba los 30 min.

En las siguientes tablas se pueden observar los resultados en función de los factores comentados.

R=1

	Profundidad									
	2		4		6		8		10	
Estudiantes	LNRPlanner	MIPS	LNRPlanner	MIPS	LNRPlanner	MIPS	LNRPlanner	MIPS	LNRPlanner	MIPS
1	0,047	0,004	0,047	0,000	0,031	0,000	0,031	0,004	0,047	0,008
2	0,031	0,000	0,047	0,004	0,047	0,008	0,047	0,008	0,047	0,008
5	0,031	0,020	0,047	0,080	0,062	0,296	0,078	0,744	0,125	1,524
10	0,047	36,302	0,078		0,156		0,328		0,655	
20	0,062		0,047		1,123		2,855		5,959	
50	0,437		5,444		22,932		65,926		148,652	
100	4,087		60,356							

R=2

	Profundidad									
	2		4		6		8		10	
Estudiantes	LNRPlanner	MIPS	LNRPlanner	MIPS	LNRPlanner	MIPS	LNRPlanner	MIPS	LNRPlanner	MIPS
1	0,047	0,004	0,031	0,004	0,047	0,004	0,047	0,004	0,047	0,012
2	0,031	0,004	0,047	0,012	0,047	0,036	0,062	0,108	0,062	0,280
5	0,047	0,468	0,062	37,270	0,156		0,312		0,655	
10	0,062		0,265		1,014		2,714		5,741	
20	0,047		2,246		9,890		29,044		62,962	
50	3,292		53,914		265,902				2058,092	
100	36,754									

R=3

	Profundidad									
	2		4		6		8		10	
Estudiantes	LNRPlanner	MIPS	LNRPlanner	MIPS	LNRPlanner	MIPS	LNRPlanner	MIPS	LNRPlanner	MIPS
1	0,047	0,000	0,047	0,004	0,047	0,004	0,047	0,008	0,062	0,024
2	0,047	0,000	0,047	0,020	0,062	0,088	0,094	0,280	0,156	0,716
5	0,062	19,509	0,125		0,546		1,045		2,215	
10	0,094		0,858		3,666		10,218		24,102	
20	0,577		8,596		39,874		119,136		285,293	
50	12,496		236,012							
100	152,896									

R=4

	Profundidad									
	2		4		6		8		10	
Estudiantes	LNRPlanner	MIPS	LNRPlanner	MIPS	LNRPlanner	MIPS	LNRPlanner	MIPS	LNRPlanner	MIPS
1	0,047	0,004	0,031	0,008	0,047	0,008	0,047	0,016	0,062	0,028
2	0,047	0,012	0,047	0,032	0,094	0,152	0,172	0,516	0,359	1,352
5	0,047		0,265		0,967		2,621		5,663	
10	0,172		2,168		9,563		28,580		62,696	
20	1,420		23,416		111,337		344,354			
50	33,025						12751,362			
100	434,678									

R=5										
	Profundidad									
	2		4		6		8		10	
Estudiantes	LNRPlanner	MIPS	LNRPlanner	MIPS	LNRPlanner	MIPS	LNRPlanner	MIPS	LNRPlanner	MIPS
1	0,047	0,000	0,047	0,004	0,047	0,016	0,094	0,028	0,125	0,048
2	0,031	0,012	0,078	0,056	0,156	0,248	0,359	0,772	0,640	2,284
5	0,062		0,499		1,997		5,663		12,184	
10	0,328		4,571		20,795		64,646		143,208	
20	0,047		50,528							
50	72,119									
100										

En estas agrupaciones podemos observar como en los casos más simples MIPS obtiene una solución mas rápidamente que nuestro planificador; esto se da cuando solo existe una forma de conseguir los objetivos pero en el momento en que existe más de una solución, normalmente MIPS no solo da un peor tiempo de respuesta, sino que la curva parabólica que se produce en el tiempo que se tarda en obtener una solución es mucho más pronunciada en MIPS que en nuestro planificador, haciendo que prácticamente con 5 alumnos no sea viable obtener una solución con MIPS. También se puede observar que nuestro planificador tiene un tiempo mínimo de respuesta por el cual, independientemente de la sencillez de la búsqueda no se obtiene una respuesta mas rápidamente.

5.2. PRUEBAS CON OTROS EJEMPLOS.

PROBLEMA 1

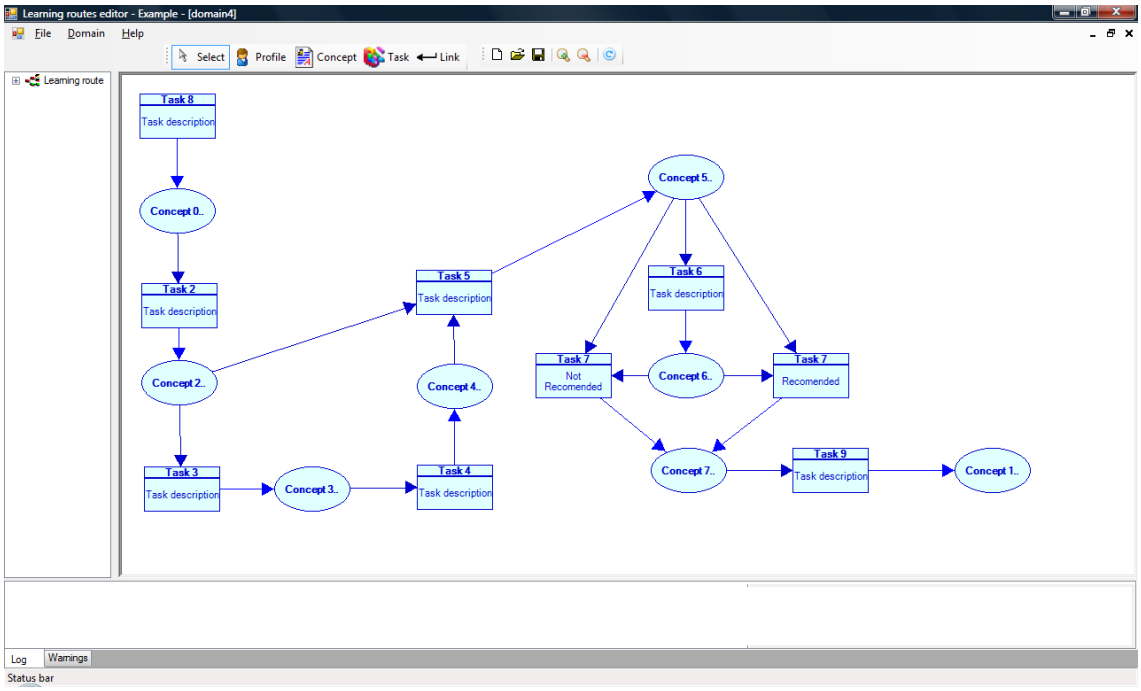


Figura 18. Esquema del dominio del problema 1.

Este dominio consiste en llegar a obtener el concepto1 habiendo dos elecciones en el camino a la solución, la primera de ellas es la forma de obtener los conceptos necesarios para la tarea 5, de los dos conceptos que tiene como precondición sólo se necesita uno, independientemente de cual sea. La segunda elección se produce en la tarea 7, la cual tiene una versión recomendada cuya duración es mayor y cuenta con una bonificación, y una versión alternativa que no está recomendada, con menor coste temporal pero sin bonificación, pero que consigue los mismos efectos que la opción recomendada.

Las particularidades de este dominio son:

- Todas las tareas tienen una duración de 1 menos la tarea 7 recomendada que tiene una duración de 3 y un punto de bonificación.

- Todas las tareas necesitan que se obtenga el concepto anterior excepto la tarea 7 no recomendada que no necesita el concepto 6.

- La tarea 5 tiene sus precondiciones en modo OR

- El objetivo de este dominio es conseguir el concepto1 en el menor tiempo posible,

- Se ha planificado para cinco alumnos con el mismo perfil

Resultado del planificador para minimizar el total-time.

	Tiempo de respuesta	Num. acciones	Valor de la métrica
Primera solución LNRPlanner	0,062s	20	4
Primera solución MIPS	20.805s	20	4

Para este ejemplo, se ha utilizado la opción de agotar el espacio de búsqueda, que no es más que garantizar que la solución que se ha encontrado es la óptima mediante la comprobación de todas las posibles combinaciones de tareas que tengan un coste menor al obtenido en la solución heurística. En nuestro planificador ha logrado esto en 6,770s, mientras que en MIPS no es posible realizar esta función a menos que se realice una búsqueda n anchura para garantizar la optimalidad y estoes mucho más costoso.

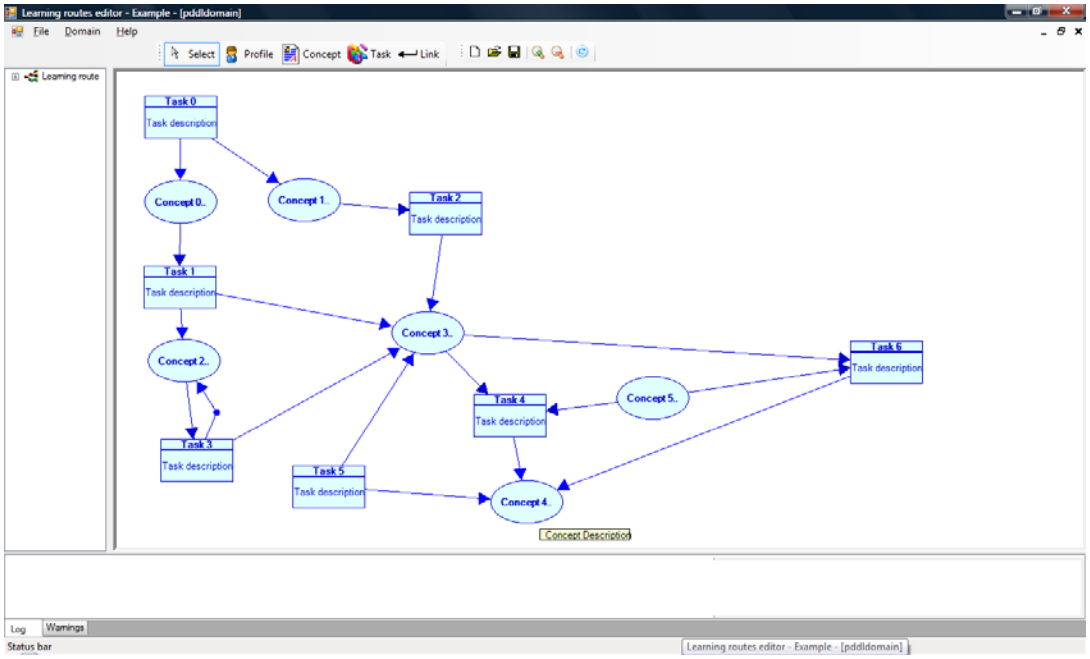
Las soluciones obtenidas tienen 20 tareas que corresponden al plan que se muestra a continuación para cada uno de los 5 alumnos, mientras que la duración total del plan, que se ha utilizado como métrica, tiene un valor de 4 ya que las acciones que incluye el plan tienen duración 1 y las tareas de diferentes alumnos se paraleliza.

El valor de la métrica obtenido con MIPS se ha recalculado a posteriori para que sus criterios coincidan con los de nuestro planificador, recordemos que nuestro planificador obtiene planes secuenciales para un alumno en particular, pero paraleliza los planes entre distintos alumnos.

Plan solución obtenido (idéntico para los distintos alumnos y planificadores):

```
Task2
Task5
Task7 Not Recommended
Task9
```

PROBLEMA 2



En este dominio se han definido tres perfiles diferentes para los alumnos, y se debe obtener el concepto 4. En este problema se pretende demostrar como en función de los perfiles diseñados, la solución se adapta a cada uno de ellos.

La definición de las tareas es la siguiente:

- Task 0 – todos los perfiles
  Duración: 1
  Condiciones: Sin condiciones
  Efectos: Concept0 100%, Concept1 75%

- Task 1 – solo tipo 1
  Duración: 1
  Condiciones: Concept0 100%
  Efectos: Concept2 80%, Concept3 20%

- Task 2 – todos los perfiles
  Duración: 1
  Condiciones: Concept1 50%
  Efectos: Concept3 75% (tipo 1) 50% (tipo 2, tipo 3)

- Task 3 – todos los perfiles
  Duración: 1
  Condiciones: Concept2 40%
  Efectos: Concept2 20%, Concept3 25%

- Task 4 – todos los perfiles
  Duración: 1
  Condiciones: Concept3 25%, Concept5 50%
  Efectos: Concept3 10%, Concept4 50%
  Repetible una vez
  Si es recomendada: bonus 1

- Task 5 – todos los perfiles
  Duración: 1
  Condiciones: sin condiciones
  Efectos: Concept3 35%(tipo 1) 25%(tipo 2) 15%(tipo 3), Concept4 25(t1) 10%(tipo 2, tipo 3)

  El estado inicial del problema según los perfiles es:

- Tipo 1:
  Concept0: 20%
  Concept4: 10%
  Concept5: 20%

- Tipo 2:
  Concept2: 10%
  Concept4: 10%
  Concept5: 50%

- Tipo 3:
  Concept1: 15%
  Concept5: 90%

  Y los objetivos a cumplir son:

  Tipo 1: Concept4 >= 80%
  Tipo 2: Concept4 >= 100%
  Tipo 3: Concept4 >= 75%

Además se han definido 2 estudiantes para el tipo 1, 2 estudiantes para el tipo 2 y un estudiante para el tipo 3.

Resultado del planificador para minimizar el total-time.

	Tiempo de respuesta	num. acciones	Valor de la métrica
Primera solución LNRPlanner	0.047s	14	4
Primera solución MIPS	0.524s	14	4

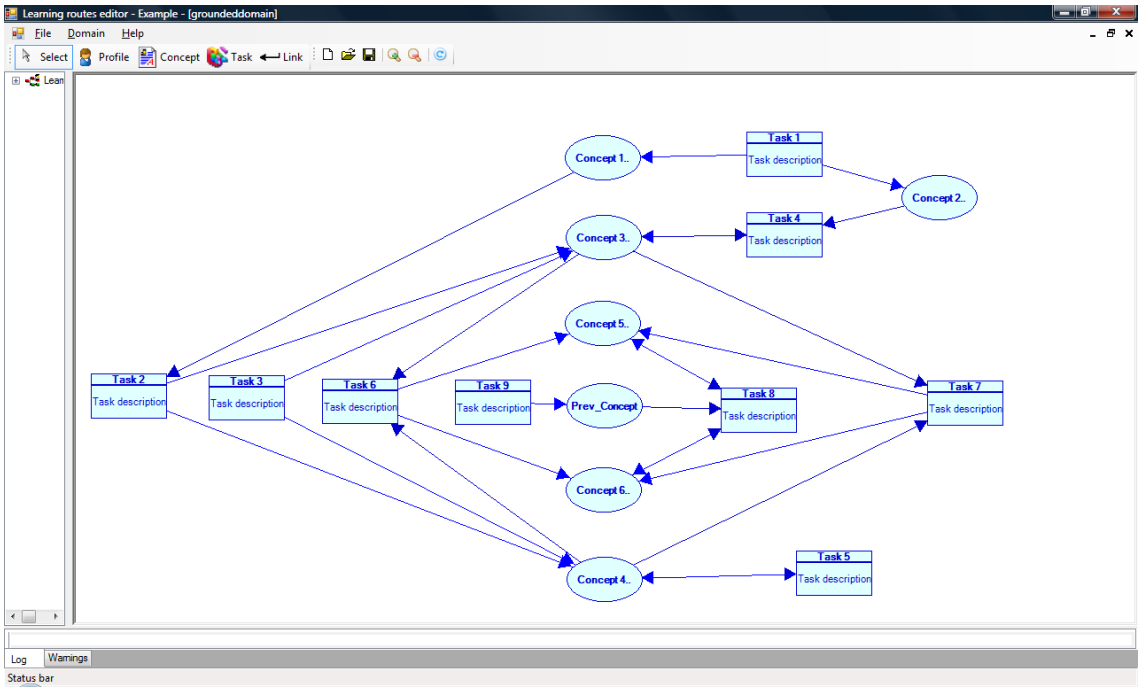
Plan solución obtenido (idéntico en ambos planificadores.

Tipo 1	Tipo 2	Tipo 3
Task5 Task4 Not Recommended	Task5 Task4 Recommended Task4 Recommended	Task0 Task2 Task4 Recommended Task4 Recommended

En este problema podemos comprobar el ejemplo de paralelización. Para cada perfil de estudiante se ha obtenido un plan diferente, y como duración total se ha calculado la duración del más largo de los sub-planes de cada estudiante. En este caso no se puede garantizar que sea la solución óptima debido a que realizar la exploración completa del espacio de búsqueda requiere muchos mas tiempo del recomendado para que sea viable realizarla.

Al igual que en el ejemplo anterior el número de acciones obtenido es la suma de las acciones de cada sub-plan de los diferentes alumnos, y se ha recalculado el valor de la métrica aportado por MIPS que no realiza esta "paralelización" por sí mismo.

PROBLEMA 3



En este escenario se ha prescindido de los perfiles, asignado a cada estudiante los valores directamente. Tampoco se ha tenido en cuenta la paralelización da duración total del plan para tener una mejor comparación con MIPS.

- Task 9 – todos los estudiantes
  Duración: 2
  Condiciones: Sin condiciones
  Efectos: prev_concept 25%

- Task 8 – todos los estudiantes
  Duración: 2
  Condiciones: prev_concept 50%, concept6 50%, concept5 50%, concept4 100%, concept3 100%
  Efectos: concept5 50%, concept6 25%

- Task 7 – todos los estudiantes
  Duración: 3
  Condiciones: concept4 100%, concept3 100%
  Efectos: concept5 50%, concept6 50%

- Task 6 – todos los estudiantes
  Duración: 5
  Condiciones: concept4 100%, concept3 100%
  Efectos: concept5 50% concept6 75%(stu10, stu9, stu7, stu4, stu1) 50% (stu8, stu5) 100% (stu6, stu2)

- Task 5 – todos los estudiantes
  Duración: 2
  Condiciones: concept4 25%
  Efectos: concept4 25%
  Repetible una vez

- Task 4 – todos los perfiles
  Duración: 2
  Condiciones: concept3 50%, concept2 100%
  Efectos: concept3 50%, concept4 25%

- Task 3 – todos los perfiles
  Duración: 2
  Condiciones: concept1 100%
  Efectos: concept4 50%(stu10, stu8, stu5, stu4, stu2) 75%(stu9, stu7, stu6, stu3, stu2), concept3 50%

- Task 2 – stu10, stu8, stu5, stu4, stu1
  Duración: 3
  Condiciones: concept1 100%
  Efectos: concept4 75%(stu10, stu5, stu1) 25%(stu8, stu4), concept3 50%

- Task 1 – todos los perfiles
  Duración: 2
  Condiciones: Sin condiciones
  Efectos: concept1 100%, concept2 100%

  El estado inicial del problema según el estudiante es:

- Stu10, stu3: Prev_Concept 50%
- Stu3, stu6, stu7, stu9: Prev_Concept 25%

Y los objetivos requeridos son:

stu10: concept6 >=100.00%
stu9: concept6 >= 75.00%
stu8: concept6 >= 75.00%
stu7: concept6 >= 100.00%
stu6: concept6 >= 50.00%
stu5: concept6 >= 75.00%
stu4: concept6 >= 50.00%
stu3: concept6 >= 100.00%
stu2: concept6 >= 75.00%
stu1: concept6 >= 100.00%

Resultado del planificador LNRPlanner para minimizar el total-time

	Tiempo de respuesta	num. acciones	Valor de la métrica
1ª solución	0.359s	54	144
2ª solución	49.015s	53	142
3ª solución	1min 33.616s	52	140
4ª solución	1min 42.086s	52	139
5ª solución	2min 33.005s	52	138
6ª solución	3min 29.602s	51	136
7ª solución	4min 33.281s	51	135
8ª solución	4min 45.215s	50	133
9ª solución	5min 40.175s	49	131
10ª solución	7min 29.202s	48	129
11ª solución	7min 35.770s	48	128
12ª solución	9min 02.038s	47	126
…			

En este ejemplo vemos como la heurística, pese a obtener una solución rápida, no es la optima, y con el proceso de búsqueda se va refinando dicha solución. No se ha agotado el espacio de búsqueda ya que al ser un problema algo complejo este tiempo es lo suficientemente grande como para que no se opte por esta opción. En el caso de MIPS no se ha podido obtener la primera solución en 30 minutos, con lo cual la comparación es obvia.

Aun así se ha reducido el problema al numero de estudiantes para el que MIPS pudiera encontrar una solución y así poder realizar una comparación entre ambos. La prueba realizada para esta comparación cuenta con tan solo los tres primeros estudiantes, cuya solución se expone a continuación.

	Tiempo de respuesta	Num. acciones	Valor de la métrica
1ª solución LRNPlanner	0,062s	22	58
2ª solución LRNPlanner	0,577s	21	56
3ª solución LRNPlanner	0,858s	21	55
4ª solución LRNPlanner	1,669s	21	54
5ª solución LRNPlanner	2,079s	20	52
6ª solución LRNPlanner	2,699s	19	50
...			
Primera solución MIPS	0,760s	14	39

Tras esta prueba podemos comprobar que MIPS genera una solución mejor en un tiempo razonable, aunque como ya hemos dicho, después de simplificar el problema.

Ejemplos de las mejoras en los planes solución obtenidos

	1ª solución LNRPlanner	12 ª solución LNRPlanner	1ª solución MIPS
Estudiante 2	task1 task3 task5 task4 task6 Duración: 13	task1 task3 task4 task6 Duración: 11	task1 task 3 task4 task6 Duración: 11
Estudiante 3	task1 task3 task5 task4 task6 task7 Duración: 16	task1 task3 task4 task6 task8 Duración: 13	task1 task3 task4 task6 task8 Duración: 13

6. CONCLUSIONES Y TRABAJO FUTURO.

En este trabajo se ha descrito el proyecto global Adaptaplan consistente en proporcionar una herramienta que use técnicas de IA, en concreto planificación, al entorno del e-learning. Este proyecto se está llevando a cabo entre diversas Universidades de toda España como son la Universidad de Granada y la Carlos III de Madrid y que tiene como objetivo integrar la herramienta con la plataforma de e-learning Moodle (http://moodle.org) y para realizar diferentes pruebas con otros planificadores.

Nuestro diseño en varios módulos requiere un planificador que aporte soluciones a las rutas diseñadas mediante la herramienta grafica. La funcionalidad del modulo del planificador consiste por una parte el planificador cuya función es aportar un conjunto de tareas que consiga los objetivos requeridos, y posteriormente tomando ese conjunto de tareas y los recursos establecidos un CSP se encargará de instanciar concretamente

en el tiempo dichas tareas, además de realizar una sincronización entre ellas si fuera necesario, y asignar los recursos en función de las necesidades de cada tarea.

Es por esto que en esta tesina se ha planteado el diseño e implementación de un planificador propio para estos entornos para poder compararlo con planificadores generales y ver si resulta conveniente la opción del planificador a medida. Y a la vista de los resultados podemos concluir que optar por un planificador propio para el entorno de *e-learning*, el cual se adapte a las características concretas de este dominio, aporta una capacidad de procesamiento muy superior a la de los planificadores generales. Sin embargo el planificador tratado en estos experimentos y esta tesina es una primera versión que debe ser refinada para la futura integración, tanto con el CSP como con el resto del proyecto.

Por ultimo decir que este proyecto ha dado lugar a varias publicaciones entre las que se encuentran las indicadas en la siguiente sección, y se esta preparando otra publicación especifica para el planificador desarrollado.

7. REFERENCIAS.

[IMS08] IMS global learning consortium, 2008. http://www.imsglobal.org
[Peachy&McCalla86] D. Peachy and G. McCalla. Using planning techniques in intelligent systems. International Journal of Man-Machine Studies, 24:77-98, 1986.
[Vassileva97] J. Vassileva. Dynamic course generation. Communication and Information Technologies, 5(2):87-102, 1997.
[LOM07] LOM. Draft standard for learning object metadata. IEEE. 15 july 2002. 6 oct. 2007. 2002. http://ieeeltsc.org
[Ullrich07] C. Ullrich. Course generation as a Hierarchical Task Network Planning Problem. Ph.D. Dissertation, University of Saarlandes, 2007.
[PASER07] E. Kontopoulos, D. Vrakas, F. Kokkoras et al. An ontology-based planning system for e-course generation. Expert Systems with Applications, 35(1/2):398-406, 2007.
[PDDL98] McDermott, D. et al (1998) "PDDL – The Planning Domain Definition Language", AIPS-98 Planning Competition Committee.
[Ghallab04] M. Ghallab, D. Nau and P.Traverso. Automated Planning. Theory and Practice. Morgan Kaufmann, 2004.
[MCCARTHY69] J. McCarthy and Hayes P.J. Some philosophical problems from the standpoint of artificial ontelligence. Machine Intelligence, 4:463-502, 1969.
[FELDER88] Felder, R.M. and Silverman, L.K. Learning and Teaching Styles in Engineering Education. Engr. Education. 78(7): 674--681. 1988.
[EDELKAMP06] Edelkamp, S. and Jabbar, S. and {Nazih}, M. Large-Scale Optimal {PDDL3} Planning with {MIPS-XXL}. Proc. Int. Conference on Automated Planning and Scheduling ({ICAPS}-2006) -- International Planning Competition. 28-30. 2006.

Referencias relacionadas con el proyecto.

A. Garrido, E. Onaindia, O. Sapena. Planning and Scheduling in an E-learning Environment. A constraint-programming based approach Engineering Applications of Artificial Intelligence. Elsevier. ISSN: 0952-1976 Vol. 21, No. 5, pp. 733-743, 2008.

E. Onaindía, A. Garrido, O. Sapena. LRNPlanner: Planning Personalized and Contextualized E-learning Routes ICTAI-08. IEEE Conference on Tools with Artificial Intelligence. IEEE Computer Society Press. 2008.

ANEXO I – Gramática aceptada por el parser.

Gramática que acepta el parser para el dominio:

`(define (`DOMAIN | REQUIREMENTS | PREDICATES | FUNCTIONS | DURATIVE-ACTION`))`

DOMAIN => `domain` nombre_del_dominio
REQUIREMENTS => `:requirements` :requerimiento … :requerimiento
PREDICATES => `:predicates`
FUNCTIONS => `:functions (`nombre_funcion_1`)` … `(`nombre_funcion_n`)`
DURATIVE-ACTION => `:durative-action` nombre_accion LEARNER | PARAMETERS |
DURATION | CONDITION | EFFECT.
LEARNER => `;;learner (`identificador_del_estudiante`)`
PARAMETERS => `:parameters`
DURATION => `:duration ( = ?duration` valor_duracion`)`
CONDITION => `:condition (` C_AND | C_OR | C_AT`)`
C_AND => `and (` C_AND | C_OR | C_AT`)`
C_OR => `or (` C_AND | C_OR | C_AT`)`
C_AT => `at` momento `(`COMPARER `(`nombre_funcion`)` valor_funcion`)`
COMPARER => `>=` | `<=` | `=` | `<>` | `>` | `<`
EFFECT => `:effect (` E_AND | E_OR | E_AT`)`
E_AND => `and (` E_AND | E_OR | E_AT`)`
E_OR => `or (` E_AND | E_OR | E_AT`)`
E_AT => `at` momento `(`ACTION `(`nombre_funcion`)` valor_accion`)`
ACTION => `increase` | `decrease`

Gramática que acepta el parser para el problema:

`(define (`PROBLEM | DOMAIN | INIT | GOAL | METRIC`))`

PROBLEM => `problem` nombre_del_problema
DOMAIN => `:domain (`nombre_del_dominio`)`
INIT => `(= (`nombre_de_la_funcion`)` valor_de_la_funcion`)`
GOAL => `:goal (`G_AND | G_OR | G_COMPARER`)`
G_AND => `and (`G_AND | G_OR | G_COMPARER`)`
G_OR => `or (`G_AND | G_OR | G_COMPARER`)`
G_COMPARER => `>=` | `<=` | `=` | `<>` | `>` | `< (`nombre_de_la_funcion`)` valor_de_la_funcion
METRIC => `:metric minimize | maximize (`OPERADOR | nombre_de_la_funcion | TOTAL_TIME `)`
OPERADOR => `+` | `-` | `*` | `/ (`OPERADOR | nombre_de_la_funcion | TOTAL_TIME | valor`)`
`(`OPERADOR | nombre_de_la_funcion | TOTAL_TIME | valor`)`