

# Integración de Planificación Temporal y Abstracción de Recursos\*

Antonio Garrido, Eva Onaindía y Federico Barber  
Dpto. Sistemas Informáticos y Computación  
Universidad Politécnica de Valencia  
Camino de Vera, s/n, Valencia (Spain)  
{agarridot, onaindia, fbarber}@dsic.upv.es

**Abstract:** *En la resolución de problemas reales de planificación es necesario tener en cuenta aspectos temporales sobre las acciones a ejecutar, ya que cada una de ellas puede tener distinta duración. Además, las acciones suelen requerir para su ejecución recursos compartidos. Todo ello incrementa la dificultad de los sistemas prácticos de planificación, a la vez que supone una línea de interés muy activa. En este artículo se presenta un sistema que integra un proceso de planificación temporal con uno de gestión de recursos abstractos. Las contribuciones principales del sistema son i) obtención de un plan temporal de duración óptima, ii) separación de la etapa de planificación y de asignación de recursos, iii) reducción del tamaño del espacio de búsqueda, y iv) facilitar la escalabilidad del sistema a medida que aumenta el número de recursos disponibles.*

**Palabras clave:** planificación temporal, scheduling, abstracción de recursos

## 1 Introducción

Los modelos de planificación clásica no permiten trabajar con problemas con características temporales pues asumen implícitamente que todas las acciones son instantáneas [2]. Esta simplificación no resulta adecuada en problemas reales donde las acciones pueden tener distinta duración [6]. Por ejemplo, en un problema de logística, la duración del operador `fly-plane` puede ser mayor que la de `load-plane`, y la acción `fly-plane(Valencia,Bilbao)` será mayor que la de `fly-plane(Valencia,Alicante)`.

Por otra parte, las acciones de los problema reales también requieren la utilización, y asignación en el tiempo, de recursos compartidos. En los planificadores clásicos basados en

---

\*Este trabajo ha sido parcialmente financiado por el proyecto n. 20010017, *Planificación de Tareas en la Navegación Autónoma de Robots Móviles*, de la Universidad Politécnica de Valencia.

Graphplan [2], el uso no simultáneo de recursos se resuelve mediante métodos añadidos al proceso de forma *artificial*. Estos métodos consisten básicamente en introducir nuevas proposiciones para representar la disponibilidad de recursos (**free-plane**, **empty-arm**, etc.) con las consecuentes relaciones de exclusión mutua (*mutex*) entre proposiciones y acciones. Sin embargo, esto presenta dos serios inconvenientes: i) se incrementa la complejidad del proceso de planificación, y ii) se incrementa el espacio de búsqueda del planificador, por lo que el rendimiento empeora exponencialmente a medida que el número de recursos aumenta (comportamiento no escalable frente al número de recursos [9]).

A pesar del claro interés en la resolución eficiente de problemas que no incluyan temporalidad de acciones y gestión explícita de recursos [10], también se han realizado esfuerzos para diseñar planificadores capaces de manejar estas características presentes en problemas reales. Actualmente, existen diversos enfoques para resolver problemas de planificación con recursos. Uno de los enfoques propone la total separación de la asignación de recursos de la etapa de planificación [7, 8, 9]. Aunque éste es el enfoque utilizado tradicionalmente, recientemente se ha suscitado un creciente interés hacia la integración de los procesos de planificación y scheduling [1, 3].

La motivación principal de este trabajo es la de intentar resolver los problemas anteriores mediante un sistema integrado de planificación temporal y scheduling [3]. Este sistema está desarrollado sobre TPSYS (*Temporal Planning SYStem* [4]), un planificador temporal basado en Graphplan. El sistema propuesto separa las características propias de planificación (a resolver por un planificador temporal) de las de gestión y asignación explícita de recursos (a resolver por un scheduler). Las principales aportaciones del sistema que proponemos son:

- Maneja acciones con distinta duración y genera planes en los que dichas acciones se pueden solapar, garantizando la optimalidad del plan de acuerdo a su duración.
- Separa las acciones que son necesarias para alcanzar los objetivos *relevantes* del problema de las acciones necesarias para garantizar la disponibilidad de recursos (abstracción de recursos). El plan construido es independiente de los recursos existentes, facilitando la escalabilidad del sistema en problemas con elevado número de recursos. Una vez obtenido este plan abstracto, un scheduler (como módulo razonador de recursos) se encarga de asignar los recursos según su disponibilidad.
- Reduce el tamaño del espacio de búsqueda haciendo que los objetivos a satisfacer interfieran lo menos posible al abstraerse de la gestión de recursos.

## 2 Revisión de TPSYS

En este apartado presentamos una breve revisión del planificador temporal TPSYS [4] puesto que constituye la base para el sistema integrado de planificación y scheduling que proponemos en este artículo. TPSYS surge como combinación de las ideas de Graphplan [2] y TGP (*Temporal Graphplan* [6]) para resolver problemas de planificación temporal.

TPSYS consiste en tres etapas. En la primera etapa se calculan todas las relaciones de exclusión estáticas entre las acciones que nunca se pueden ejecutar en paralelo, porque sus proposiciones entran en conflicto. Estas relaciones de exclusión son estáticas porque sólo dependen de la definición de las acciones. De forma similar, se calculan las relaciones de exclusión *ap-mutex* entre acciones y las proposiciones que cada acción elimina. El objetivo de esta etapa es explotar al máximo todas las relaciones de exclusión que se pueden *precalcular* a partir de la definición del dominio.

La segunda etapa consiste en la expansión de un grafo temporal (grafo dirigido multi-nivel que alterna niveles de proposiciones y acciones al igual que Graphplan). La diferencia principal con respecto al grafo de planificación de Graphplan es que, en TPSYS, cada nivel está etiquetado con el instante de tiempo en el que las proposiciones o acciones pueden aparecer. Además, para reducir el tamaño del grafo, las acciones `no-op` y las aristas `delete` no se almacenan. El grafo temporal se expande incrementalmente. En cada nivel temporal de acciones, se generan todas las acciones cuyas precondiciones están presentes y no son excluyentes en el mismo nivel de proposiciones. Tras la generación de cada acción, se actualiza el nivel temporal de proposiciones (dependiente de la duración de la acción) con todas las proposiciones que añade dicha acción. La expansión del grafo finaliza cuando todos los objetivos del problema están presentes en un nivel temporal de proposiciones y no son excluyentes entre sí.

La tercera etapa realiza la extracción de un plan ejecutable mediante la búsqueda de un *flujo* de acciones. Para cada proposición se estudian todas las acciones que producen dicho objetivo y se selecciona aquella acción que no sea excluyente con las acciones ya planificadas, obteniéndose el plan de mínima duración.

### 3 Modelo de problema de planificación y scheduling

En este trabajo, un problema de planificación y scheduling se especifica por la tupla  $\{\mathcal{I}_s, \mathcal{A}, \mathcal{F}_s, \mathcal{R}, \mathcal{D}_{\max}\}$  donde  $\mathcal{I}_s$  y  $\mathcal{F}_s$  representan la situación inicial y final. El tiempo viene modelado por  $\mathbb{R}^+$ .  $\mathcal{A}$  representa el conjunto de acciones, que cuentan con duración, y que pueden requerir cualquiera de los recursos  $\mathcal{R}$  renovables.  $\mathcal{D}_{\max}$  representa la máxima duración del plan permitida por el usuario.

#### 3.1 La naturaleza de los recursos

En la literatura se ha hablado mucho del término recurso con significados distintos [9]. Aun a pesar de no existir una definición comúnmente aceptada, en el campo de la planificación en inteligencia artificial, un recurso es cualquier objeto, generalmente limitado, necesario para la ejecución de una acción y, que por tanto, restringe la estructura del plan. Aunque algunos autores trabajan en la identificación automática de recursos [9], esta cuestión todavía no está resuelta, por lo que asumimos que el usuario debe representar la tipología y disponibilidad de recursos y expresarla en el dominio.

En nuestro problema,  $\mathcal{R}$  representa el conjunto inicial de recursos unitarios, renovables del dominio. Estos recursos se pueden agrupar en tipos  $\mathcal{R}_{tipo}$ , como subclases de

$\mathcal{R}$  ( $\mathcal{R}_{tipo} \subseteq \mathcal{R}$ ). Por ejemplo, si en un problema disponemos de cuatro camiones y tres aviones, dispondremos de dos tipos de recursos,  $\mathcal{R}_{camion}$  y  $\mathcal{R}_{avion}/\mathcal{R} = \mathcal{R}_{camion} \cup \mathcal{R}_{avion}$ . Al igual que en [9], no todos los recursos del mismo tipo se consideran *intercambiables*, representados como  $\mathcal{R}_{intercambiable} \subseteq \mathcal{R}_{tipo}$ , ya que si la situación inicial no es la misma para todos ellos, no se podrán utilizar indistintamente. Si en el problema anterior hubiera camiones en Valencia y en Alicante, habría dos tipos de recursos intercambiables  $\mathcal{R}_{camion\_Valencia}$  y  $\mathcal{R}_{camion\_Alicante}$ , ambos pertenecientes a  $\mathcal{R}_{camion}$ . Esto es así porque en el proceso de generación de un plan no es lo mismo trabajar con los camiones que se encuentran en una ciudad que en otra, pues el plan implica acciones de movimiento distintas.

### 3.2 Nuevas características del lenguaje de modelización

El lenguaje de modelización utilizado es el conocido PDDL (*Planning Domain Definition Language*). Debido a la imposibilidad de manejar recursos en este lenguaje es necesario extender la especificación de las acciones. En la Figura 1 se muestra un ejemplo con tres acciones básicas de un típico dominio de transporte a las que se le han incorporado duraciones (sección `:duration`) y la utilización explícita de un recurso de tipo *camión* ( $\mathcal{R}_{camion}$ ) de capacidad unitaria. La gestión de los recursos disponibles viene dada por las secciones `:lock`, `:unlock` y `:use` (bloqueo, desbloqueo y utilización, respectivamente), por lo que el conjunto de acciones  $\mathcal{A}$  ya no tendrá precondiciones y efectos sobre la disponibilidad de recursos. En la sección `:lock` se indican todos los recursos de un determinado tipo que bloquea cada acción, de forma que ninguna otra acción podrá bloquear dicho recurso hasta que se ejecute una acción que lo desbloquee (sección `:unlock`). La sección `:use` indica que la acción utiliza un recurso de un tipo determinado. Una acción puede desbloquear recursos que no haya bloqueado y viceversa, e incluso bloquear y desbloquear el mismo recurso (suponiendo entonces que la acción no deja bloqueado el recurso más allá de la ejecución de la propia acción). Por ejemplo en el caso que nos ocupa, la acción `cargar` utiliza y deja bloqueado el recurso de tipo *camión* que utilice; el objeto se queda ocupando el recurso y no lo libera hasta que se ejecuta la acción `descargar`. Si la acción `cargar` no bloqueara el recurso sería posible ejecutar varias acciones `cargar` consecutivamente, lo que no se corresponde con la realidad al trabajar con un recurso de capacidad unitaria. La acción `mover` simplemente utiliza un recurso de tipo *camión*, independientemente de que en él haya, o no, un objeto cargado que lo bloquee. Para garantizar la completitud del planificador, en la situación inicial y final es necesario indicar cualquier restricción sobre los recursos. Por ejemplo, si existe algún objeto inicialmente cargado en un *camión* habrá que representar que dicho *camión* está inicialmente bloqueado.

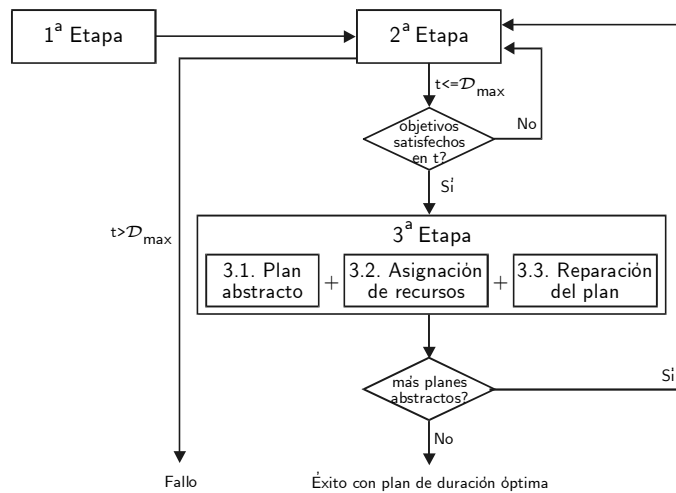
## 4 Arquitectura del sistema

El sistema funciona sobre las tres etapas de TPSYS (ver Figura 2). Tras la primera etapa de preproceso, la segunda y tercera etapa se intercalan hasta que se obtenga el plan de duración óptima (*‘Éxito’*) o la duración del plan supere a  $\mathcal{D}_{max}$  (*‘Fallo’*).

(:action cargar	(:action mover	(:action descargar
:parameters ()	:parameters ()	:parameters ()
:precondition ()	:precondition ()	:precondition ()
:effect ()	:effect ()	:effect ()
:duration 50	:duration 100	:duration 80
:lock ( $\mathcal{R}_{camion}$ )	:lock ()	:lock ()
:unlock ()	:unlock ()	:unlock ( $\mathcal{R}_{camion}$ )
:use ( $\mathcal{R}_{camion}$ )	:use ( $\mathcal{R}_{camion}$ )	:use ( $\mathcal{R}_{camion}$ )

Figure 1. Fragmento de un dominio de transporte.

Figure 2. Diagrama con las 3 etapas del sistema integrado



Con el fin de detallar las tres etapas del sistema resolveremos un sencillo problema de transporte de tres objetos (01, 02 y 03) desde Valencia a Alicante. Para ello utilizaremos las acciones definidas en la Figura 1, más tres acciones análogas de *embarcar*, *volar* y *desembarcar* que utilizan un recurso de tipo *avión* ( $\mathcal{R}_{avion}$ ), con duraciones 50, 40 y 80 respectivamente. En la situación inicial sólo disponemos de un *camión* ‘CM’ en Valencia y de un *avión* ‘AV’ en Madrid que para traerlo a Valencia implica ejecutar una acción *volar* de duración 100.

#### 4.1 Primera etapa. Abstracción de recursos

En esta etapa se asumen recursos ilimitados, por lo que se asignan *variables recurso* para la consecución de cada uno de los objetivos. Una *variable recurso*, que denominaremos ?recurso, representa a cualquiera de los recursos intercambiables que se pueden utilizar en la ejecución de la acción. Posteriormente, el proceso de asignación de re-

cursos asignará un recurso concreto a cada *variable recurso*. En el ejemplo actual, el objeto 01 sólo podrá ser transportado por la variable *camión* ?CM1 o por la variable *avión* ?AV1. Análogamente, los objetos 02 y 03 sólo podrán ser transportados por las variables ?CM2, ?AV2 y ?CM3, ?AV3, respectivamente. Por lo tanto,  $\{?CM1, ?CM2, ?CM3\} \in \mathcal{R}_{camion\_Valencia} \subseteq \mathcal{R}_{camion}$ , y  $\{?AV1, ?AV2, ?AV3\} \in \mathcal{R}_{avion\_Madrid} \subseteq \mathcal{R}_{avion}$ . En la Tabla I se muestran todas las acciones instanciadas de acuerdo a las *variables recurso* para el objeto 01. De forma análoga se instancian las acciones para los objetos 02 y 03.

Table I. Acciones con las *variables recurso* que se generan para el objeto 01.

01	<i>camión</i>	cargar(01, Val, ?CM1), cargar(01, Al, ?CM1), mover(Val, Al, ?CM1), mover(Al, Val, ?CM1), descargar(01, Val, ?CM1), descargar(01, Al, ?CM1)
	<i>avión</i>	embarcar(01, Val, ?AV1), embarcar(01, Al, ?AV1), volar(Val, Al, ?AV1), volar(Al, Val, ?AV1), volar(Ma, Val, ?AV1) desembarcar(01, Val, ?AV1), desembarcar(01, Al, ?AV1)

El sistema RealPlan [7, 9] también lleva a cabo una abstracción de recursos durante el proceso de planificación. Sin embargo, RealPlan realiza una abstracción de recursos distinta a la que se propone aquí al asumir un único recurso abstracto de capacidad ilimitada. La diferencia es obvia: nuestro sistema asigna una *variable recurso* distinta a cada acción que lo requiera, apareciendo tantas acciones *mover* y *volar* como *variables recurso* haya, mientras que RealPlan sólo maneja una acción *mover* y *volar* con el único recurso que contempla para transportar todos los objetos.

Tras instanciar todas las acciones, la primera etapa calcula todas las relaciones de exclusión estáticas entre acciones y entre acciones y proposiciones de la misma forma que lo hace TPSYS [4]. Estas relaciones de exclusión serán mínimas puesto que las acciones utilizan *variables recurso* distintas.

## 4.2 Segunda etapa. Expansión del grafo temporal

La segunda etapa expande un grafo temporal moviéndose incrementalmente por instantes de tiempo  $t$ , exactamente igual a como lo hace TPSYS [4]. El proceso de expansión alterna niveles de proposiciones y acciones,  $P_{[t]}$  y  $A_{[t]}$  respectivamente. Partiendo de las proposiciones de la situación inicial disponibles en el nivel de proposición  $P_{[0]}$ , se generan sucesivos niveles de acciones y proposiciones. En cada nivel de acción  $A_{[t]}$ , se generan todas las acciones que se pueden ejecutar en  $t$  porque sus precondiciones están presentes y no son excluyentes en  $P_{[t]}$ . Para cada acción generada, de duración  $dur$ , se añaden sólo sus efectos positivos (**add-effects**) al nivel de proposición correspondiente  $P_{[t+dur]}$ . La expansión termina cuando todas las proposiciones de la situación final  $\mathcal{F}_s$  están presentes y no son excluyentes en un nivel de proposición  $P_{[t]}$ . En el ejemplo de transporte, esto ocurre en el nivel  $P_{[230]}$  que es donde se consiguen los tres objetos en Alicante. Además,

si  $t > \mathcal{D}_{\max}$  el algoritmo finaliza con ‘Fallo’ (ver Figura 2) porque no existe ningún plan factible de duración inferior a  $\mathcal{D}_{\max}$ . Una propiedad interesante del grafo temporal es que su longitud se reduce en comparación con los planificadores basados en Graphplan. Al trabajar con acciones que manejan *variables recurso* distintas, el número de relaciones de exclusión entre ellas es mínimo y por tanto el número de niveles temporales que se generan también lo es.

### 4.3 Tercera etapa. Obtención de un plan óptimo

Esta etapa se divide en tres procesos, tal y como aparece en la Figura 2, que se ejecutan secuencialmente con el objetivo de obtener un plan óptimo.

#### 4.3.1 Obtención de un plan abstracto

El proceso de obtención de un plan coincide con la tercera etapa de TPSYS comentada en el apartado 2. Básicamente, el proceso busca una secuencia de acciones que satisfagan los objetivos del problema sin acciones excluyentes ejecutadas en paralelo. La diferencia estriba en que el plan obtenido será un plan abstracto en el sentido de que todas las acciones planificadas consideran la existencia de suficientes recursos para ellas. Por lo tanto, el primer plan abstracto obtenido cumplirá las siguientes propiedades:

- i) El plan obtenido es el plan con mayor grado de paralelismo posible.
- ii) Como consecuencia de la propiedad anterior, y bajo el enfoque de TPSYS, el plan obtenido es el más corto en cuanto a duración.
- iii) Si en el problema hubiese un número de recursos suficiente, el plan sería directamente ejecutable. Por tanto, el primer plan obtenido sería el óptimo.
- iv) El plan abstracto obtenido facilita enormemente la escalabilidad del sistema. Si el número de recursos disponibles se incrementa, sin variar el número de recursos intercambiables, el plan abstracto no se verá modificado y seguirá siendo válido.

#### 4.3.2 Asignación de recursos

La asignación de recursos la lleva a cabo un proceso de scheduling que instancia los recursos necesarios a cada acción del plan abstracto [3]. Aunque el proceso de scheduling se puede resolver mediante técnicas de satisfacción de restricciones, en este sistema se puede utilizar cualquier método de investigación operativa para la programación de proyectos con recursos limitados, tal y como aparece en [5]. Lógicamente, la optimalidad en la asignación de recursos va en función del proceso de scheduling utilizado. En el caso de que no se disponga de recursos suficientes para la ejecución en paralelo de las acciones, el método utilizado secuencia las acciones en el tiempo, obteniendo una nueva cota inferior de la duración del plan. Esto supone ejecutar todas las acciones que utilizan el mismo recurso secuencialmente, de forma que no utilicen simultáneamente el mismo recurso.

Como ya vimos en el apartado 3.2, para secuenciar acciones es necesario tener en cuenta los bloqueos de los recursos: una acción sólo puede bloquear un recurso si éste no está ya bloqueado. Al contrario que en planificadores basados en **Graphplan**, el comportamiento de este proceso es mejor cuanto mayor sea el número de recursos disponibles en cada tipo [9].

### 4.3.3 Reparación del plan

Una vez asignados los recursos al plan abstracto se dispone de un plan con los recursos totalmente instanciados. Si ha sido necesario secuenciar las acciones en dicho plan (por falta de recursos), éste puede no ser ejecutable por haberse eliminado alguno de los objetivos *intermedios*. En el ejemplo con el que estamos trabajando, cada vez que se utiliza el *camión* ‘CM’ o el *avión* ‘AV’ para realizar un transporte es necesario añadir acciones que dejen el recurso de nuevo disponible en la localización inicial.

La reparación del plan consiste en comprobar si al secuenciar las acciones del plan aparece algún conflicto de planificación que hace el plan no factible. En tal caso, este proceso añade las acciones necesarias para que el plan sea ejecutable. Para añadir estas acciones será necesario desplazarse a lo largo del grafo temporal generado en la segunda etapa, buscando un conjunto de acciones que resuelva los conflictos. Este proceso es idéntico al proceso de TPSYS de extensión del grafo, volviendo a la segunda etapa, cuando tras la etapa de extracción del plan no se cumplen todos los objetivos del problema [4].

Generalmente, los conflictos de planificación se resuelven añadiendo acciones que dejen disponibles los recursos para volver a ser utilizados. Lógicamente, si el proceso de asignación de recursos no ha realizado una secuenciación de las acciones del plan, el plan será directamente ejecutable. Tras este proceso se dispondrá de un plan ejecutable con una duración que proporcionará una cota para descartar planes de mayor duración.

### 4.3.4 Ilustración mediante un ejemplo

En este apartado ilustraremos el comportamiento de la tercera etapa sobre el ejemplo de transporte. Aunque existen muchas permutaciones para transportar los objetos, los 4 planes alternativos que proporciona el planificador (en los instantes de tiempo  $t = 230$  y  $t = 270$ ) aparecen en la Tabla II. El plan abstracto-1 implica la utilización de tres *camiones*, y el plan abstracto-2 de tres *aviones*. El plan abstracto-3 utiliza un *camión* y dos *aviones*, mientras que el plan abstracto-4 utiliza dos *camiones* y un *avión*. En  $t = 230$  se obtiene el plan abstracto-1 que sería el óptimo si existieran al menos tres *camiones* en el problema. Puesto que sólo se dispone de un *camión*, el proceso de asignación de recursos secuenciar las acciones, obteniendo una duración de  $t' = 3 \times (230) = 690$ , donde 230 es la suma de las duraciones de *cargar*, *mover* y *descargar*. Finalmente, el proceso de reparación del plan añade dos acciones *mover* para dejar el recurso *camión* disponible en *Valencia*, obteniendo un plan ejecutable de duración  $t'' = 890$ . Puesto que todavía existen planes abstractos alternativos con distintas asignaciones de recursos no se puede garantizar que dicho plan sea el óptimo. El plan abstracto-2 se encuentra en  $t = 270$ , pero se debe secuenciar hasta  $t' = 100 + 3 \times 170 = 610$  (se mueve el *avión* hasta *Valencia* y

luego se transportan los tres objetos). La reparación de este plan hace que su duración pase a ser  $t'' = 690$  (constituyendo el mejor plan actual). En  $t = 270$  todavía existen más planes abstractos, por lo que se repiten los mismos pasos, obteniendo el plan ejecutable-3 de duración  $t'' = 480$  que pasa a ser el mejor plan. El plan abstracto-4 se encuentra también en  $t = 270$ , pero hay que secuenciarlo y repararlo, haciendo que su duración  $t'' = 560$  supere a la del plan-3. En este punto, al no existir más planes alternativos con distinta asignación de recursos, el sistema finaliza devolviendo como plan óptimo el plan-3, que utiliza el *avión* 'AV' para O1 y O2, y el *camión* 'CM' para O3.

La obtención del plan óptimo consiste en obtener todos los planes que se van encontrado de acuerdo a las distintas asignaciones de recursos realizadas. En cada momento sólo se trabaja con un plan y su duración actúa como cota para descartar otros planes. Sin embargo, otra alternativa sería la de mantener varios planes alternativos simultáneamente, de forma que sólo se repararan aquéllos presumiblemente más ventajosos.

Table II. Planes que obtiene el sistema, con sus tiempos correspondientes.

	Plan abstracto				Asig. recursos		Reparac. del plan
	O1	O2	O3	$t$	Recursos	$t'$	$t''$
Plan-1	?CM1	?CM2	?CM3	230	$3 \times \text{CM} + 0 \times \text{AV}$	690	890
Plan-2	?AV1	?AV2	?AV3	270	$0 \times \text{CM} + 3 \times \text{AV}$	610	690
Plan-3	?AV1	?AV2	?CM3	270	$1 \times \text{CM} + 2 \times \text{AV}$	440	480
Plan-4	?AV1	?CM2	?CM3	270	$2 \times \text{CM} + 1 \times \text{AV}$	460	560

## 5 Conclusiones

La resolución de problemas reales de planificación con características temporales y de recursos compartidos dificulta el proceso de planificación, dando lugar a diferentes propuestas de resolución. Este artículo presenta un sistema que integra un proceso de planificación temporal con uno de asignación de recursos, haciendo más hincapié en la motivación y descripción del sistema que en experimentos concretos sobre una implementación en particular. Así, la contribución principal de este sistema ha sido la de eliminar del proceso específico de planificación la gestión de recursos.

El sistema propuesto funciona en tres etapas. En la primera se lleva a cabo una abstracción de recursos, que supone la existencia de recursos suficientes (ilimitados) para la obtención de los objetivos del problema. Por lo tanto, las acciones del problema se instancian de forma que el número de relaciones de exclusión es mínimo, pues cada objetivo se resuelve con recursos distintos. La segunda etapa expande un grafo temporal de forma sencilla en el que se representan los instantes de tiempo en el que las acciones se pueden ejecutar. Finalmente, la tercera etapa se divide en tres procesos distintos con el objetivo de encontrar un plan óptimo ejecutable: i) obtención de un plan abstracto que satisfaga todos los objetivos de la forma más paralela posible, ii) asignación de recursos

concretos a cada una de las acciones del plan abstracto, y iii) reparación del plan, resolviendo cualquier conflicto que pueda aparecer al asignar los recursos, en caso de que no existan recursos suficientes en el problema.

El sistema propuesto presenta la ventaja de la escalabilidad en problemas de planificación con elevado número de recursos. Un incremento en el número de recursos disponibles no implica coste adicional en el proceso de planificación, sino una reducción en la complejidad global del sistema al poderse ejecutar simultáneamente más acciones, mejorando sensiblemente el rendimiento de los enfoques proposicionales clásicos basados en Graphplan. Adicionalmente, como futura línea de trabajo está la de manejar recursos multicapacidad [5]. De esta forma, el scheduler permitirá ejecutar múltiples acciones simultáneamente, reduciendo la duración del plan y aprovechando mejor las características potenciales de los recursos.

## References

- [1] R. Barták. Towards mixed planning and scheduling. In *Proc. CPDC2000 Workshop*, 2000.
- [2] A.L. Blum and M.L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281–300, 1997.
- [3] A. Garrido and F. Barber. Integrating planning and scheduling. *Applied Artificial Intelligence*, 15(5):471–491, 2001.
- [4] A. Garrido, E. Onaindía, and F. Barber. Time-optimal planning in temporal problems. In *Proc. European Conference on Planning (ECP-01)*, 2001.
- [5] W. Herroelen, B. De Reyck, and E. Demeulemeester. Resource-constrained project scheduling: A survey of recent developments. *Computers and Operations Research*, 25(4):279–302, 1998.
- [6] D.E Smith and D.S. Weld. Temporal planning with mutual exclusion reasoning. In *Proc. 16th Int. Joint Conf. on AI (IJCAI-99)*, pages 326–337, 1999.
- [7] B. Srivastava. RealPlan: Decoupling causal and resource reasoning in planning. In *Proc. AAAI-2000*, 2000.
- [8] B. Srivastava and S. Kambhampati. Efficient planning through separate resource scheduling. In *Proc. AAAI Spring Symp. on Search Strategy under Uncertain and Incomplete Information*, 1999.
- [9] B. Srivastava, S. Kambhampati, and B.M. Do. Planning the project management way: Efficient planning by effective integration of causal and resource reasoning in RealPlan. Technical report, Arizona State University, ASU CSE, 2000.
- [10] D.S. Weld. Recent advances in AI planning. *AI Magazine*, 20(2):93–123, 1999.