Planificación Automática

Grupo PLG

Universidad Carlos III de Madrid

IA. 2008-09

Indice

Indice

Indice

Indice

Indice

Indice

Indice

Indice

Grafos de planes. GRAPHPLAN

- A mediados de los 90, se produce una revolución en planificación: GRAPHPLAN [Blum and Furst, 1995]
- Se basa en grafos de planificación, que son estructuras que representan de forma completamente instanciada la resolución del problema
- Dos fases:
  - Generación de alcanzabilidad: niveles alternados de proposición y acción
  - Búsqueda de plan válido: búsqueda hacia atrás de un posible plan paralelo

Otros planificadores

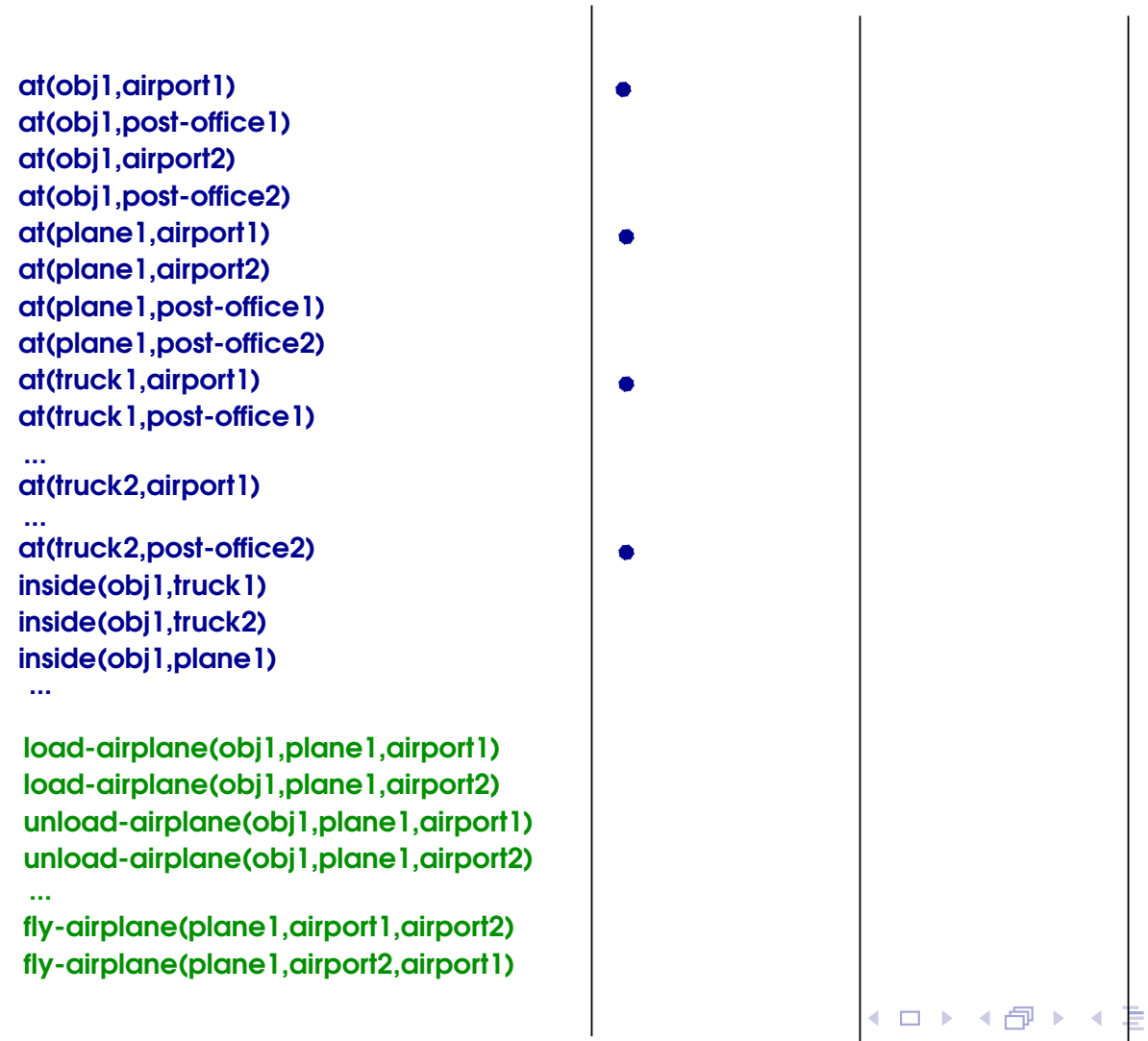
- IPP: [Koehler *et al.*, 1997]
- STAN: [Long and Fox, 1999]
- LPG: [Gerevini *et al.*, 2003]
- TGP [Smith and Weld, 1999]
- Como un CSP: [Kambhampati, 2000]

Fases del algoritmo

Dos fases

- Expansión del grafo: extiende un grafo del plan hasta que las condiciones necesarias para existencia de plan se cumplen
  - alterna niveles de proposición y acción
  - se calculan restricciones binarias de “mutex”
  - termina cuando todas las metas aparecen como no mutex en el mismo nivel de proposición
- Extracción de la solución: búsqueda hacia atrás
  - Se comienza la búsqueda de qué acciones concretas asignar a cada meta / submeta
  - Si no se logra una asignación válida, se generan más niveles de proposiciones / acciones y se repite el proceso

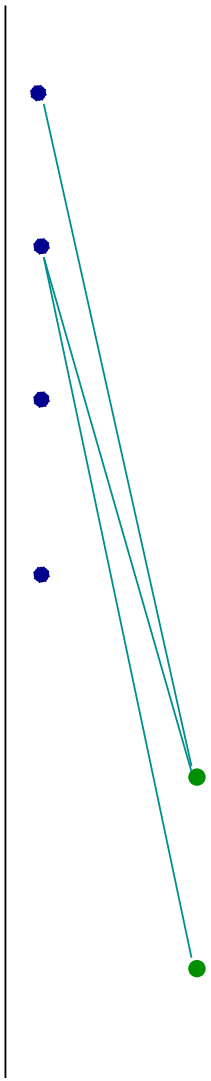
Grafo de planificación



Grafo de planificación

at(obj1,airport1)
at(obj1,post-office1)
at(obj1,airport2)
at(obj1,post-office2)
at(plane1,airport1)
at(plane1,airport2)
at(plane1,post-office1)
at(plane1,post-office2)
at(truck1,airport1)
at(truck1,post-office1)
...
at(truck2,airport1)
...
at(truck2,post-office2)
inside(obj1,truck1)
inside(obj1,truck2)
inside(obj1,plane1)
...

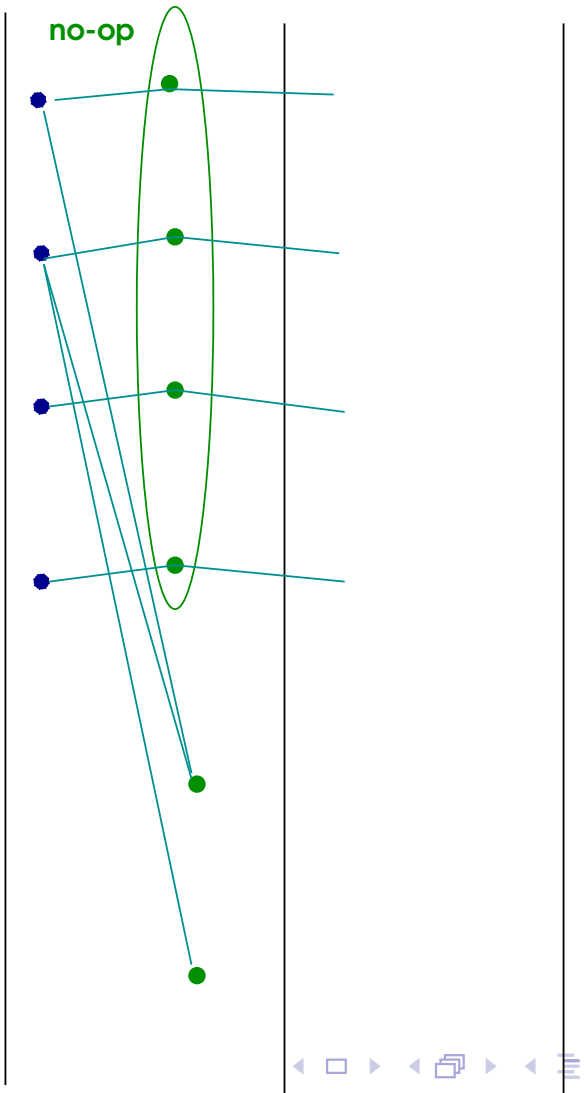
load-airplane(obj1,plane1,airport1)
load-airplane(obj1,plane1,airport2)
unload-airplane(obj1,plane1,airport1)
unload-airplane(obj1,plane1,airport2)
...
fly-airplane(plane1,airport1,airport2)
fly-airplane(plane1,airport2,airport1)



Grafo de planificación

at(obj1,airport1)
at(obj1,post-office1)
at(obj1,airport2)
at(obj1,post-office2)
at(plane1,airport1)
at(plane1,airport2)
at(plane1,post-office1)
at(plane1,post-office2)
at(truck1,airport1)
at(truck1,post-office1)
...
at(truck2,airport1)
...
at(truck2,post-office2)
inside(obj1,truck1)
inside(obj1,truck2)
inside(obj1,plane1)
...

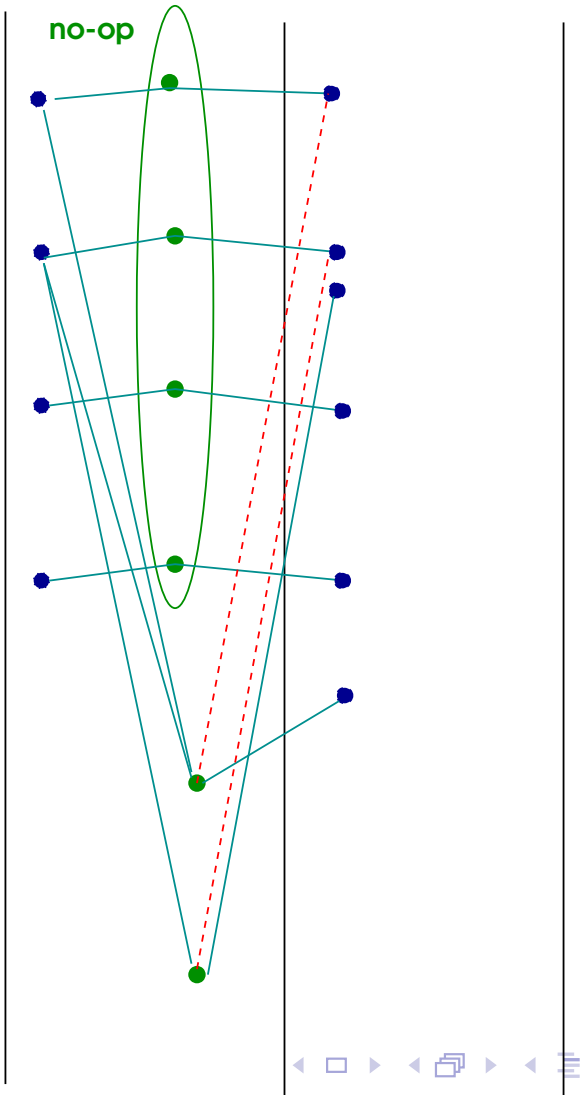
load-airplane(obj1,plane1,airport1)
load-airplane(obj1,plane1,airport2)
unload-airplane(obj1,plane1,airport1)
unload-airplane(obj1,plane1,airport2)
...
fly-airplane(plane1,airport1,airport2)
fly-airplane(plane1,airport2,airport1)



Grafo de planificación

at(obj1,airport1)
at(obj1,post-office1)
at(obj1,airport2)
at(obj1,post-office2)
at(plane1,airport1)
at(plane1,airport2)
at(plane1,post-office1)
at(plane1,post-office2)
at(truck1,airport1)
at(truck1,post-office1)
...
at(truck2,airport1)
...
at(truck2,post-office2)
inside(obj1,truck1)
inside(obj1,truck2)
inside(obj1,plane1)
...

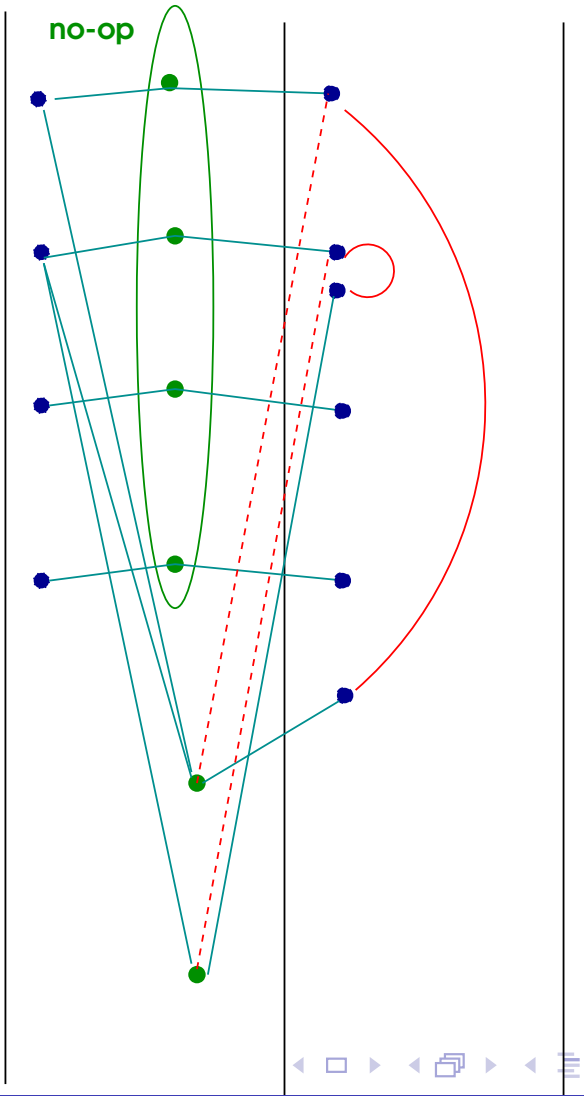
load-airplane(obj1,plane1,airport1)
load-airplane(obj1,plane1,airport2)
unload-airplane(obj1,plane1,airport1)
unload-airplane(obj1,plane1,airport2)
...
fly-airplane(plane1,airport1,airport2)
fly-airplane(plane1,airport2,airport1)



Grafo de planificación

at(obj1,airport1)
at(obj1,post-office1)
at(obj1,airport2)
at(obj1,post-office2)
at(plane1,airport1)
at(plane1,airport2)
at(plane1,post-office1)
at(plane1,post-office2)
at(truck1,airport1)
at(truck1,post-office1)
...
at(truck2,airport1)
...
at(truck2,post-office2)
inside(obj1,truck1)
inside(obj1,truck2)
inside(obj1,plane1)
...

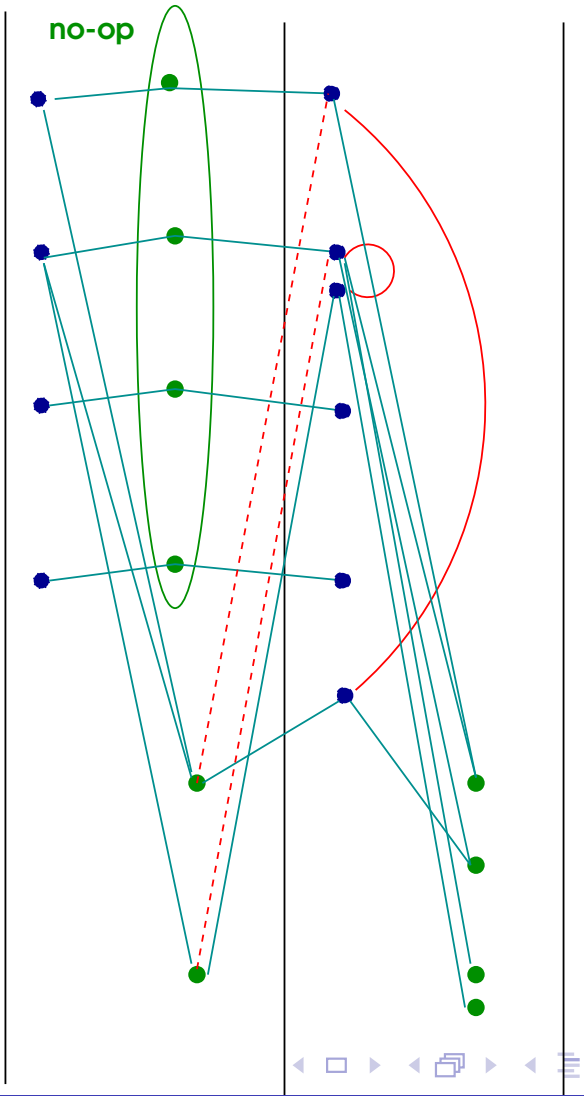
load-airplane(obj1,plane1,airport1)
load-airplane(obj1,plane1,airport2)
unload-airplane(obj1,plane1,airport1)
unload-airplane(obj1,plane1,airport2)
...
fly-airplane(plane1,airport1,airport2)
fly-airplane(plane1,airport2,airport1)



Grafo de planificación

at(obj1,airport1)
at(obj1,post-office1)
at(obj1,airport2)
at(obj1,post-office2)
at(plane1,airport1)
at(plane1,airport2)
at(plane1,post-office1)
at(plane1,post-office2)
at(truck1,airport1)
at(truck1,post-office1)
...
at(truck2,airport1)
...
at(truck2,post-office2)
inside(obj1,truck1)
inside(obj1,truck2)
inside(obj1,plane1)
...

load-airplane(obj1,plane1,airport1)
load-airplane(obj1,plane1,airport2)
unload-airplane(obj1,plane1,airport1)
unload-airplane(obj1,plane1,airport2)
...
fly-airplane(plane1,airport1,airport2)
fly-airplane(plane1,airport2,airport1)



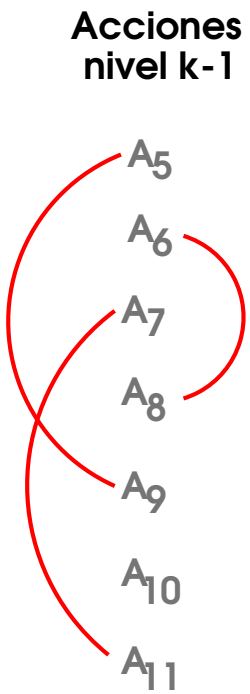
Mutex

- Dos acciones son mutex si:
  - una borra una precondición o efecto positivo de la otra

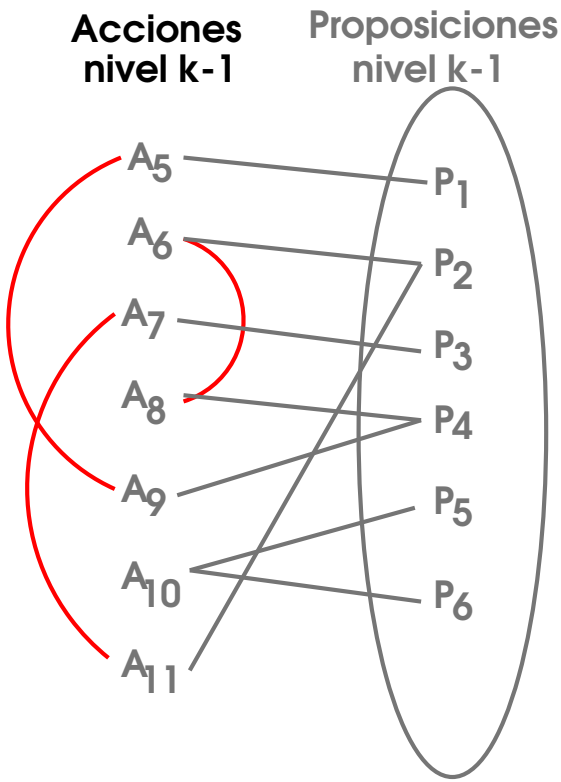
    levantar(A) y quitar(C,B)

  - una precondición de una acción es mutex con otra precondición de la otra acción
- Dos proposiciones son mutex si:
  - cada acción que consiga la primera proposición es mutex con alguna acción que consiga la segunda

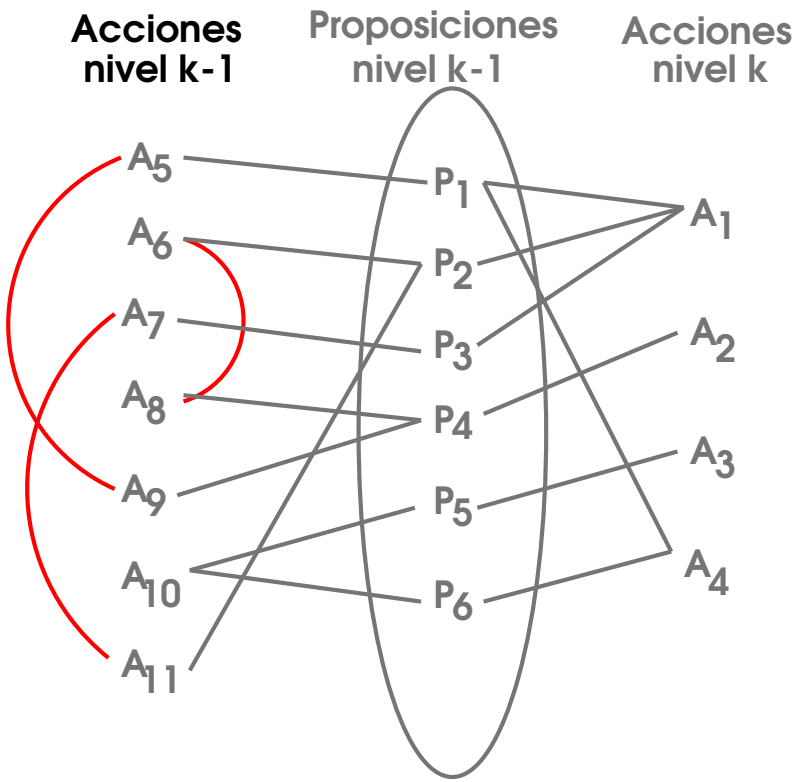
Ejemplo de búsqueda



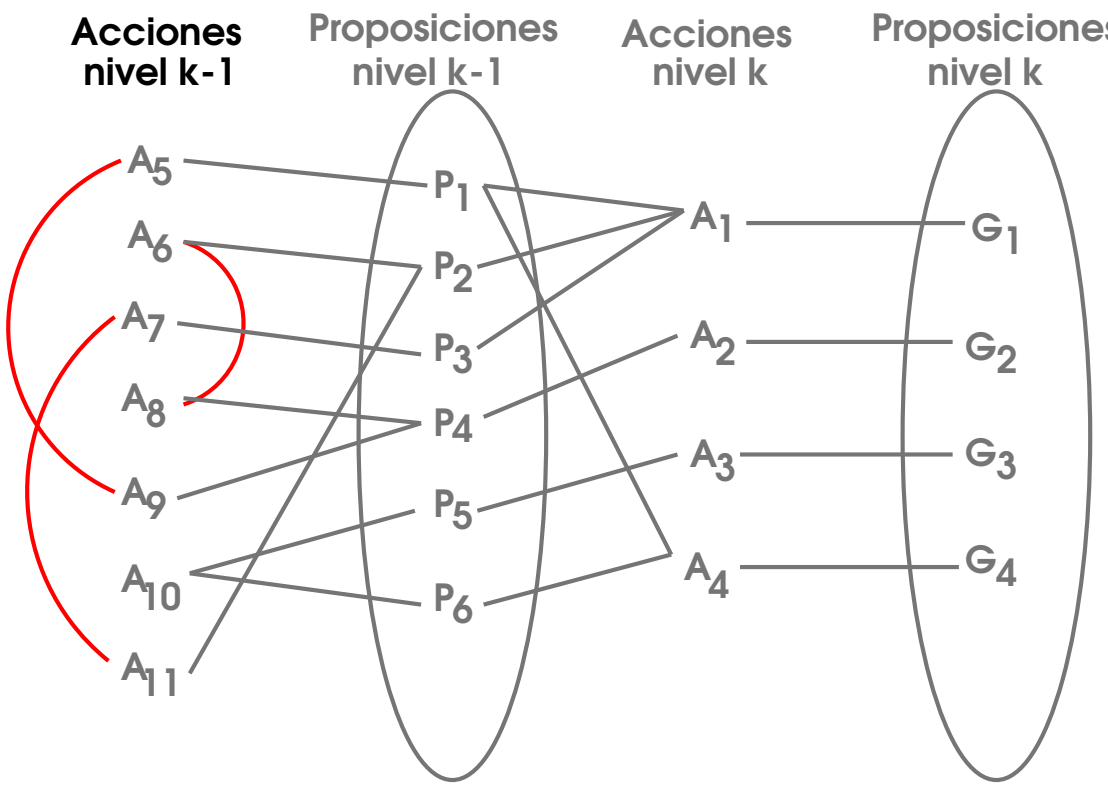
Ejemplo de búsqueda



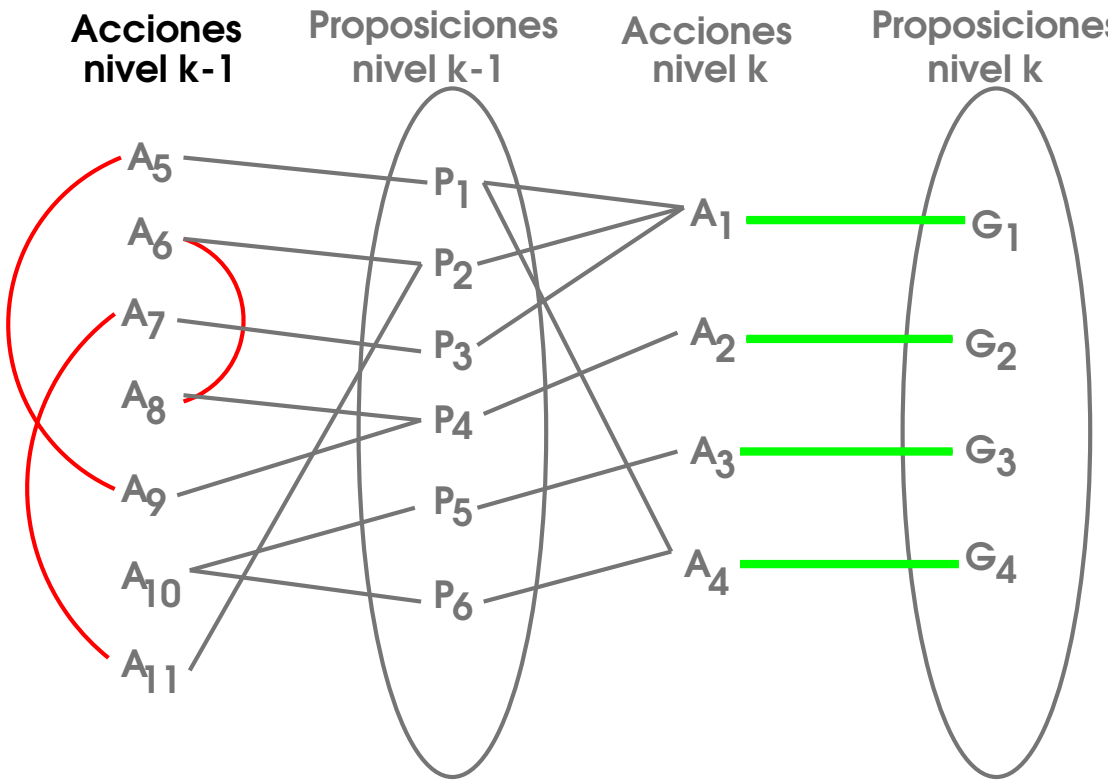
Ejemplo de búsqueda



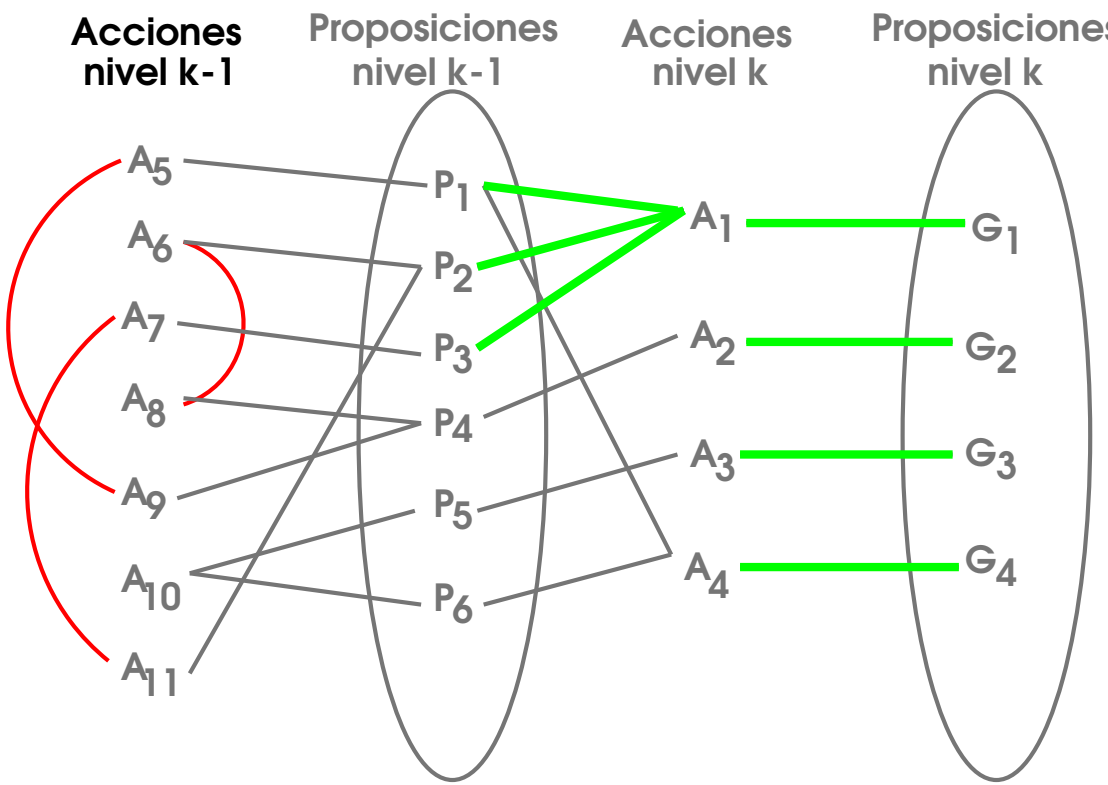
Ejemplo de búsqueda



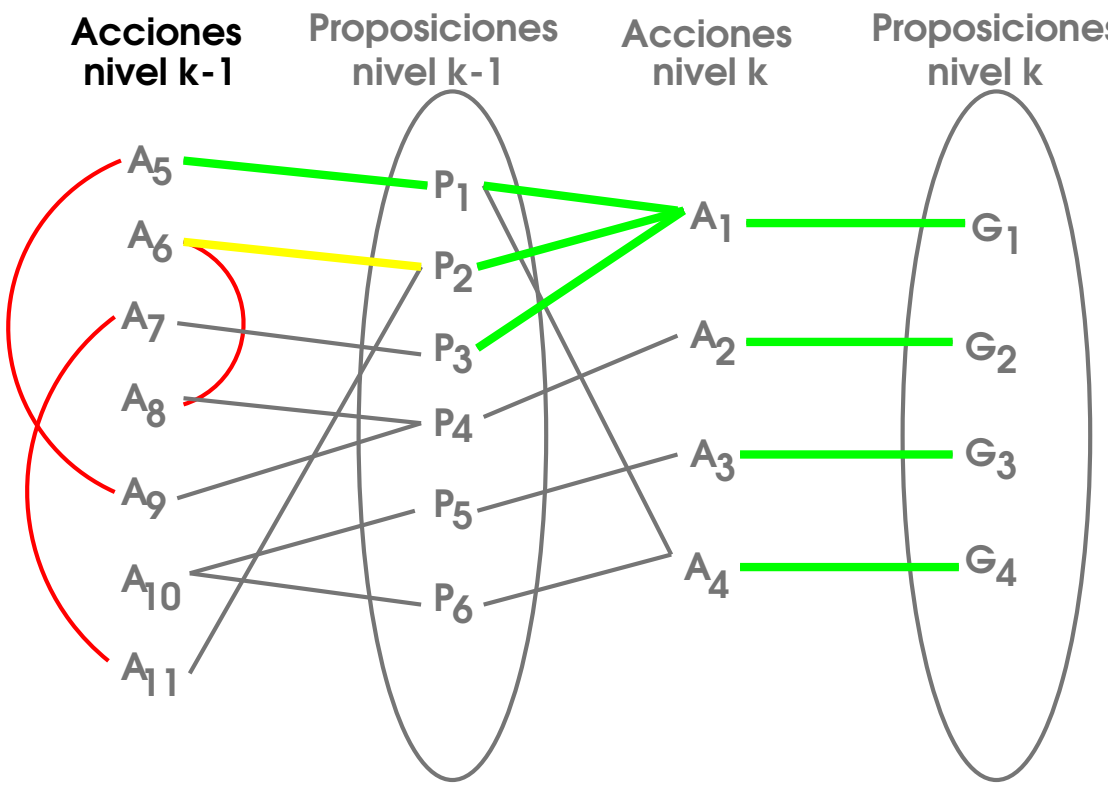
Ejemplo de búsqueda



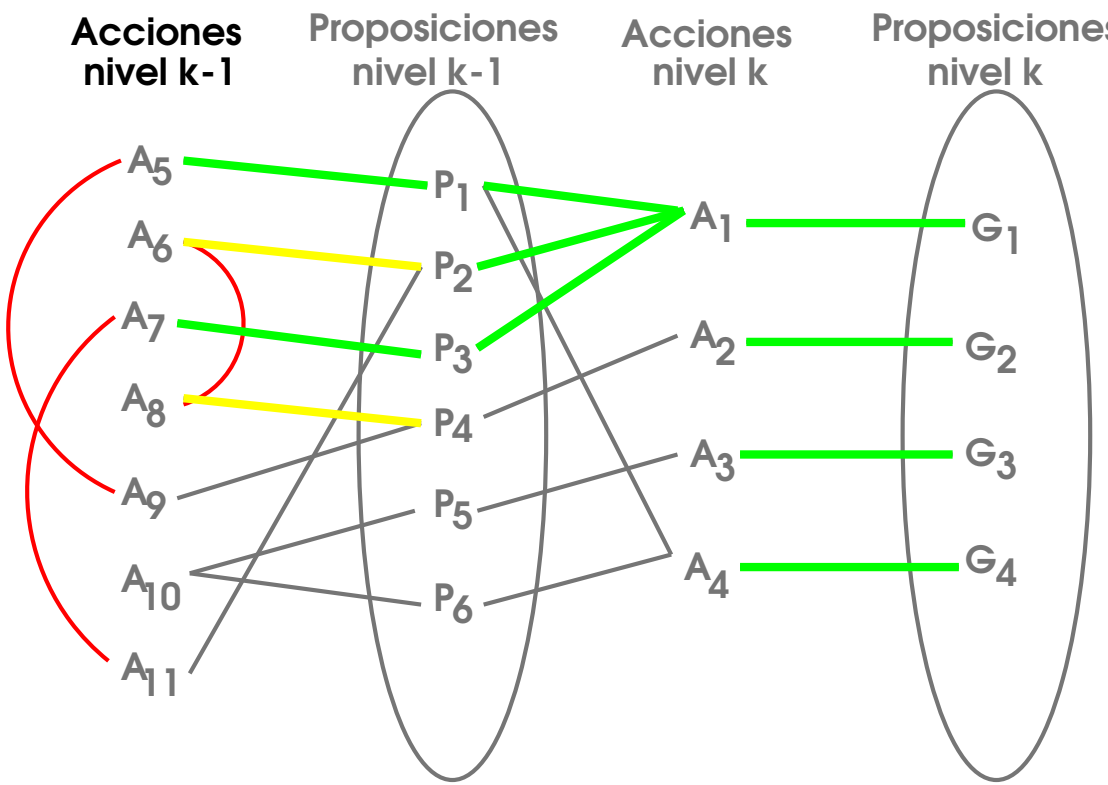
Ejemplo de búsqueda



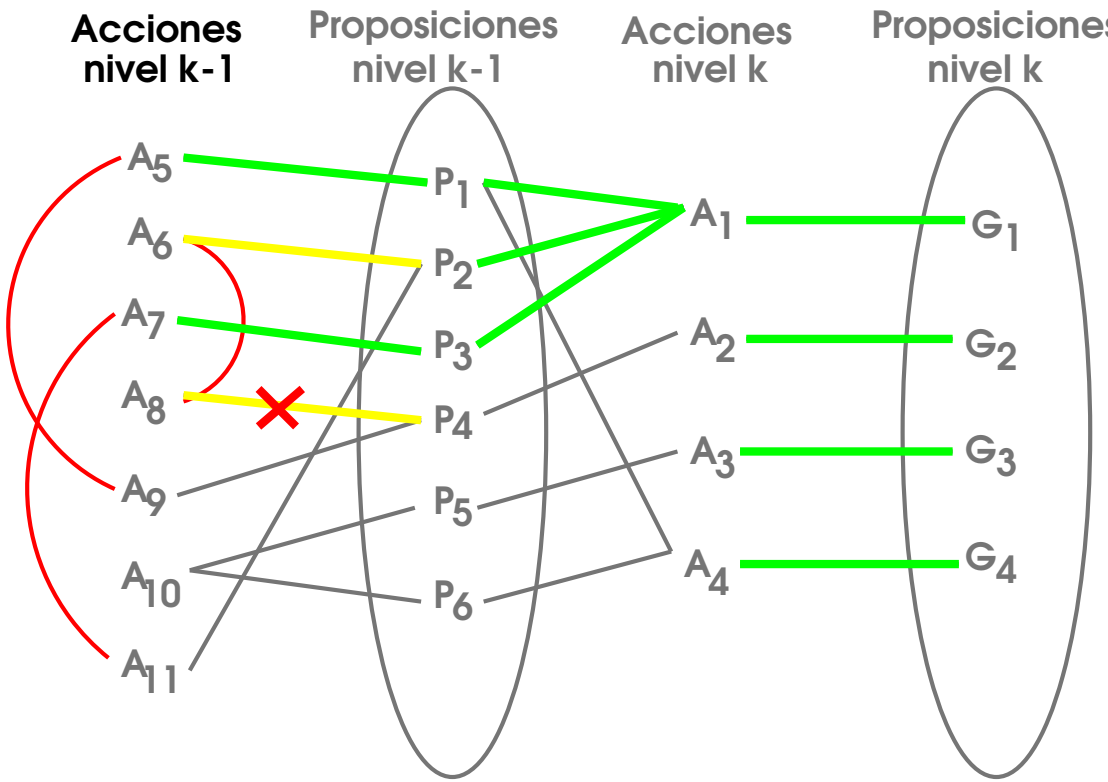
Ejemplo de búsqueda



Ejemplo de búsqueda



Ejemplo de búsqueda



Otras cuestiones

- Es completo y *sound* (el plan generado es válido)
- Obtiene el plan paralelo óptimo en términos de número de pasos de ejecución paralelo
- No asegura encontrar el óptimo en términos de número de operadores
- Memos: se guarda la información de que ha fallado en ese nivel para cuando vuelva
- Tiempo: se puede mejorar el algoritmo para tener en cuenta la duración de las acciones

Eficiencia

- Se pueden implementar mecanismos que mejoran la eficiencia:
  - un vector de bits para cada estado
  - cuatro vectores de bits para cada acción instanciada (dos para precondiciones y dos para efectos)
  - se puede guardar para cada proposición y acción la primera vez que apareció en el grafo
  - se puede guardar para cada mutex el primer nivel donde desapareció
  - se pueden eliminar las proposiciones que no cambian su valor

Más información en

- Libro: [Ghallab *et al.*, 2004]
- Otros sistemas: STAN [Long and Fox, 1999], IPP [Koehler *et al.*, 1997], LPG [Gerevini *et al.*, 2003]
- Visto como CSP dinámico: [Kambhampati, 2000]
- Mejorándolo con aprendizaje: [Kambhampati, 1999]

SATPLAN

- El que la planificación sea PSpace-Complete es cierto cuando la longitud de los planes puede ser exponencialmente larga
- Sin embargo, si se trata de encontrar un plan que tenga menos de k pasos en la solución, la dificultad se queda en NP-completo
- El problema se convierte en conocer “a priori” la longitud
- Una solución: búsqueda binaria
- Se convierte la planificación a un problema de satisfacibilidad de una fórmula lógica (CNF) con un horizonte n (longitud del plan)
- Se resuelve con una máquina SAT (WALKSAT, CHAFF, ...)
- Si se encuentra la solución, se para. Si no, se aumenta n y se repite el proceso

Una posible codificación

- A cada predicado instanciado se le añade el nivel en el que está

  `en(obj1,aeropuerto1,1),en(obj1,aeropuerto1,2), ...`

- El estado inicial se representa como la conjunción de todas las proposiciones que son ciertas y todas las que son falsas

  `en(obj1,aeropuerto1,0),en(avión1,aeropuerto1,0), ...`

- Las metas se representan como la conjunción de todas las proposiciones que deben ser ciertas y todas las que deben ser falsas en el instante *n*

  `en(obj1,oficina2,3),en(obj2,oficina1,3), ...`

Codificación de operadores

- A cada acción instanciada se le añade el nivel en el que está

  `cargar(obj1,avión1,aeropuerto1,1),`
  `cargar(obj1,avión1,aeropuerto1,2)`

- Cada acción en el instante *i* necesita las precondiciones en el instante *i* y consigue los efectos en el instante $i + 1$

  `cargar(obj1,avión1,aeropuerto1,1)` $\rightarrow$
  `en(obj1,aeropuerto1,1),en(avión1,aeropuerto1,1),`
  `dentro(obj1,avión1,2),~en(obj1,aeropuerto1,2)`

- Solo una acción se permite en cada paso

  `~cargar(obj1,avión1,aeropuerto1,1)`$\vee$
  `~descargar(obj1,avión1,aeropuerto1,1)`

Más en codificación de operadores

- Explicación del axioma de marco: si una proposición aparece en un nivel $i+1$ y no estaba antes, alguna acción que la consigue debe aparecer en i

$$\texttt{en(obj1,aeropuerto1,2)},\sim\texttt{en(obj1,aeropuerto1,1)} \rightarrow \texttt{cargar(obj1,avión1,aeropuerto1,1)}$$

- y si una acción desaparece de un nivel $i+1$ y antes estaba, debe aparecer una acción que la elimina en el nivel i

$$\sim\texttt{en(obj1,aeropuerto1,2)},\texttt{en(obj1,aeropuerto1,1)} \rightarrow \texttt{descargar(obj1,avión1,aeropuerto1,1)}$$

Planificación como satisfacibilidad

- Dos tipos de algoritmos:
  - basados en Davis-Putnam: completos y válidos
  - estocásticos (como WALKSAT): válidos, pero no completos

Problemas SAT

- SAT: problemas de satisfacibilidad lógica
- Dada: una fórmula arbitraria en lógica proposicional (en FNC)
- Determinar: una asignación de valor a las proposiciones que haga cierta la fórmula
- Algoritmo:
  1. Estado-inicial es conjunto de asignaciones nula
  2. Seleccionar una proposición y generar dos sucesores del nodo actual: asignar verdadero a la proposición y asignar falso
  3. Seleccionar por cuál de los dos nodos seguir
  4. Sustituir el valor elegido por la proposición en la fórmula lógica y simplificar
  5. Si la fórmula queda como verdadera, se termina
  6. Si la fórmula se hace falsa, se hace retroceso
  7. Si no, se vuelve a 2

Más información en

- Idea inicial: [Kautz and Selman, 1992]
- Técnicas: SATPLAN [Kautz and Selman, 1996], BLACKBOX [Kautz and Selman, 1999]

Indice

Indice

Referencias

Avrim L. Blum and Merrick L. Furst.

Fast planning through planning graph analysis.

In Chris S. Mellish, editor, *Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI-95*, volume 2, pages 1636–1642, Montréal, Canada, August 1995. Morgan Kaufmann.

Alfonso Gerevini, Alessandro Saetti, and Ivan Serina.

Planning through stochastic local search and temporal action graphs.

Journal of Artificial Intelligence Research, 20:239–290, 2003.

Malik Ghallab, Dana Nau, and Paolo Traverso.

Automated Task Planning. Theory & Practice.

Morgan Kaufmann, 2004.

Subbarao Kambhampati.

Improving graphplan's search with ebl & ddb techniques.

In Thomas Dean, editor, *Proceedings of the IJCAI'99*, pages 982–987, Stockholm, Sweden, July-August 1999. Morgan Kaufmann Publishers.

Subbarao Kambhampati.

Planning graph as a (dynamic) CSP: Exploiting EBL, DDB and other CSP search techniques in Graphplan.

Journal of Artificial Intelligence Research, 12:1–34, 2000.

Henry Kautz and Bart Selman.

Planning as satisfiability.

In *Proceedings of ECAI'92*, 1992.

Henry Kautz and Bart Selman.

Pushing the envelope: Planning, propositional logic, and stochastic search.

In *Proceedings of AAAI-96*, pages 1194–1201, 1996.

Henry Kautz and Bart Selman.

Unifying sat-based and graph-based planning.

In *Proceedings of IJCAI-99*, Stockholm (Sweden), 1999.

Jana Koehler, Bernhard Nebel, Jörg Hoffmann, and Yannis Dimopoulos.

Extending planning graphs to an ADL subset.

In S. Steel and R. Alami, editors, *Proceedings of the 4th European Conference on Planning, ECP'97*, volume 1348 of *Lecture Notes in Computer Science*, pages 273–285. Springer-Verlag, 1997.

Derek Long and Maria Fox.

Efficient implementation of the plan graph in STAN.

Journal of Artificial Intelligence Research, 10:87–115, 1999.

David E. Smith and Daniel S. Weld.

Temporal planning with mutual exclusion reasoning.

In *Proceedings of IJCA'99*, pages 326–337, 1999.