Sistemas Inteligentes I

Tema 7. Planificación

José A. Montenegro Montes
monte@lcc.uma.es

Resumen

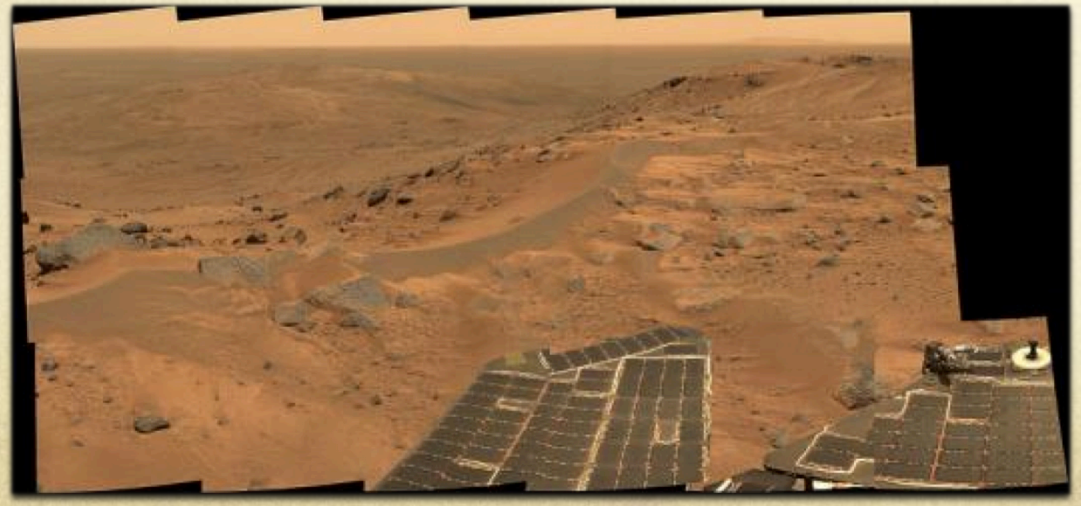
- Introducción
- Modelos de Planificación
- Lenguajes de Planificación
- Algoritmos de Planificación
- Conclusiones
- Bibliografía

Planificación

Introducción

Motivación

- En enero de 2004, dos vehículos robóticos de la NASA, llamados *Spirit* y *Opportunity*, llegaron a Marte con éxito
- Cada día había que generar un plan nuevo para cada vehículo
  - Cada plan tenía que satisfacer reglas de seguridad complejas, al mismo tiempo que conseguir tantos resultados científicos como fuera posible
  - Se supuso que los vehículos no sobrevivirían mucho más de 90 días



Panorámica "tipo postal" tomada por el *Spirit*: una meseta azotada por el viento, con rocas esparcidas, pequeños afloramientos de minerales, y dunas de arena

Motivación

- Los edificios modernos tienen varios ascensores
  - Los rascacielos tienen muchos grupos de ascensores, y se tarda varios minutos en llegar a la última planta
- Sin una planificación adecuada, el viaje en momentos de máximo tráfico podría ser una pesadilla

Generalidades

- La planificación, o sea, el desarrollo de un plan de acción para conseguir ciertos objetivos, es una parte crítica de la IA
- Aquí estudiaremos cómo un agente puede usar la estructura de un problema para construir planes de acción complejos
- Introduciremos una representación para los problemas de planificación que es capaz de manejar problemas grandes

Modelos de Planificación

Modelos de Planificación

- La Planificación Clásica se centra en la selección de acciones en entornos que son deterministas y cuyos estados iniciales son conocidos.

- El modelo puede ser descrito como un espacio de estado con los siguientes elementos:
  - Un conjunto finito y discreto de estados *S*,
  - Un **estado inicial** conocido
  - Un conjunto de **estados objetivos**
  - **Acciones** aplicable en cada estado
  - Un **función de transición de estado** determinista *f(a,s)* para y
  - **Costes de las acciones** *c(a,s)* que podría depender de las acciones y los estados.

Modelos de Planificación

- Una **solución** o **plan** es una secuencia de acciones a_0, ..., a_1 que genera una secuencia de estados s_0,s_1, ..., s_{n+1}
  - tal que a_i es aplicable en el estado si y
  - finaliza en el estado $s_{i+1} = f(a_i.s_i)$, que es un estado objetivo.
- El coste de un plan es la suma del coste de las acciones y un plan será óptimo si tienen un coste mínimo.
  - Cuando los costes de las acciones son todos 1, el coste del plan se reduce a la longitud del plan y el plan optimo se reduce al plan más corto.
- El cómputo de un plan puede ser establecido como un problema de búsqueda del camino en un grafo dirigido, cuyos nodos son los estados, y cuyo nodos origen y destino son el estado inicial s_0 y los estado objetivos S_G.

Lenguajes de Planificación

Strips

Lenguajes de Planificación

- Uno de los lenguajes más comunes para representar los problemas de planificación es Strips (inicios de 70's).
  - Lenguaje basado en variables booleanas

- Un problema en Strips es una tupla P = {F, O, I, G} donde
  - F es el conjunto de variables o fluentes,
  - O es el conjunto de operadores o acciones
  - $I \subseteq F$ es la situación inicial, y
  - $G \subseteq F$ representa el objetivo

Lenguajes de Planificación

- En Strips, las acciones o $\in$ O están representadas por tres conjuntos de átomos denominados Add, Delete, y Precondition lists, denotadas como *Add(o)*, *Del(o)*, *Pre(o)*.
  - *Add(o)* describe los átomos que hacen la acción o verdad,
  - *Del(o)* los átomos que hacen o falso
  - *Pre(o)* los átomos que deben ser verdad para que la acción pueda ser aplicada.

Lenguajes de Planificación

- Como problema tenemos un dominio que incluye tres localizaciones l_1 , l_2 , y l_3 , y tres tareas t_1 , t_2 , y t_3 , donde ti solamente puede ser realizado en l_i

  - **Conjunto F de fluentes**
    - $at(l_i)$
    - $done(t_i)$,

  - **Conjunto O de acciones** ($i,j=1, \ldots 3$, listas *pre*, *add*, y *delete*)
    - $go(l_i, l_j)$

$$Pre(a) = \{at(l_i)\} , Add(a) = \{at(l_j)\} , Del(a) = \{at(l_i)\}$$

    - $do(t_i)$

$$Pre(a) = \{at(l_i)\} , Add(a) = \{done(t_i)\} , Del(a) = \{ \}$$

Lenguajes de Planificación

- El problema de realizar las tareas t_1 y t_2 comenzando en l_3 puede ser modelado por la tupla P = {F, I, O, G} donde

$$I = \{at(l_3)\} \text{ y } G = \{done(t_1), done(t_2)\} .$$

- Una solución a P es una secuencia de acciones aplicables que mapea el estado s_0=I a un estado donde los objetivos en G son todos verdad.

$$\pi = \{go(l_3, l_1), do(t_1), go(l_1, l_2), do(t_2)\}$$

Lenguajes de Planificación

PDDL

Lenguajes de Planificación

- PDDL (Planning Domain Definition Language) es un intento de estandarizar los lenguajes de planificación de IA.
  - Fue desarrollado inicialmente en 1998 y está inspirado, entre otros, en STRIPS

- PDDL describe los cuatro componentes que necesitamos para definir un problema de búsqueda:
  - el estado inicial,
  - las acciones que están disponibles en un estado,
  - el resultado de aplicar una acción,
  - y la comprobación del objetivo

Lenguajes de Planificación

- Cada estado se representa como una conjunción de fluentes, que son átomos planos, sin funciones
  - La suposición del mundo cerrado significa que cualquier fluente que no sea mencionado es falso

- Las **acciones** se describen mediante un conjunto de esquemas de acción, que consiste en un nombre de acción, una lista de todas las variables usadas, una precondición y un efecto

Action(*Fly*(*p*,*from*,*to*),

PRECOND: *At*(*p*,*from*) ∧ *Plane*(*p*) ∧ *Airport*(*from*) ∧ *Airport*(*to*)

EFFECT: ¬*At*(*p*,*from*) ∧ *At*(*p*,*to*))

Lenguajes de Planificación

- La precondición y el efecto de una acción son conjunciones de literales
- Una acción *a* es aplicable en el estado *s* si *s* satisface las precondiciones
- El resultado de ejecutar la acción *a* en el estado *s* es el estado *s'*.
  - *s' es* obtenido:
    - eliminando de *s* los fluentes que aparecen como literales negativos en los efectos de la acción (lista de borrado),
    - añadiendo los fluentes que son literales positivos en los efectos de la acción (lista de inserción)

Ejemplo de problema de planificación

- Consideremos un problema de transporte aéreo de mercancías que consiste en cargar y descargar mercancías y llevarlas de un lugar a otro
- **Acciones**: Load, Unload, y Fly
- Las acciones afectan a dos predicados:
  - *In*(*c*,*p*) significa que la mercancía *c* está en el avión *p*
  - *At*(*x*,*a*) significa que el objeto *a* está en el aeropuerto *a*

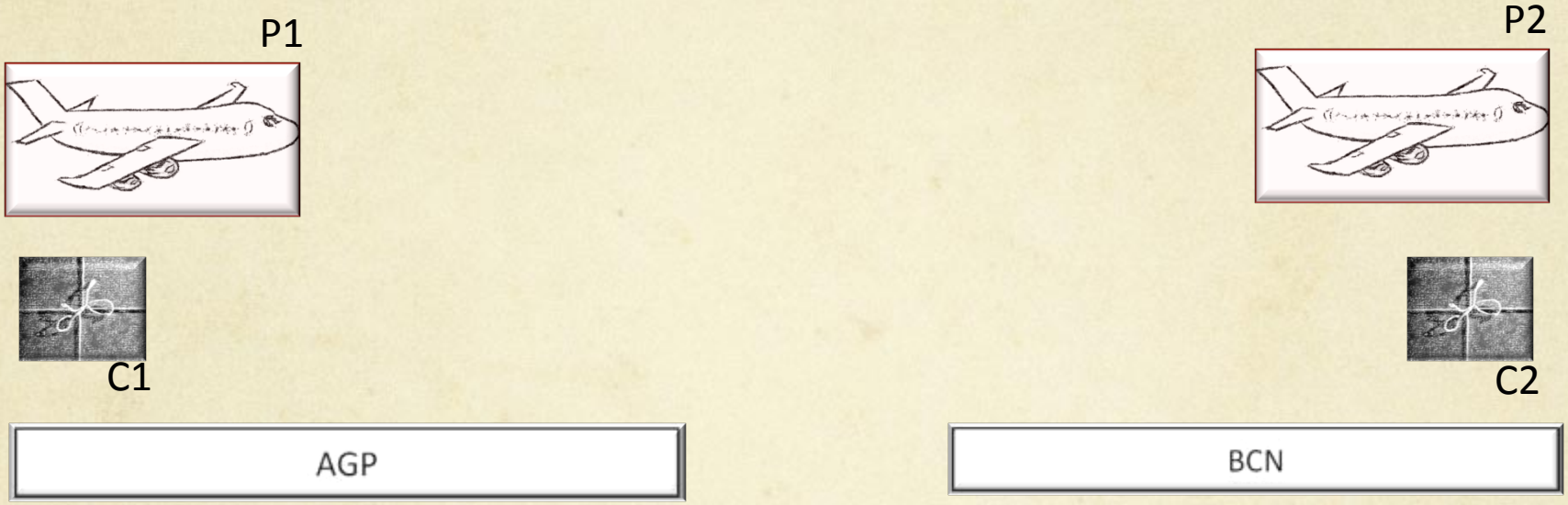
Ejemplo de problema de planificación (II)

- **Action(Load(c,p,a),**
  PRECOND: $At(c,a) \wedge At(p,a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$
  EFFECT: $\neg At(c,a) \wedge In(c,p)$)

- **Action(UnLoad(c,p,a),**
  PRECOND: $In(c,p) \wedge At(p,a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$
  EFFECT: $At(c,a) \wedge \neg In(c,p)$)

- **Action(Fly(p,from,to),**
  PRECOND: $At(p,from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$
  EFFECT: $\neg At(p,from) \wedge At(p,to)$)

Ejemplo de problema de planificación (III)



$$
\begin{aligned}
Init(\quad & At(C_1, AGP) \land At(C_2, BCN) \land \\
& At(P_1, AGP) \land At(P_2, BCN) \land \\
& Cargo(C_1) \land Cargo(C_2) \land Plane(P_1) \land Plane(P_2) \land \\
& Airport(AGP) \land Airport(BCN))
\end{aligned}
$$

Ejemplo de problema de planificación (III)

$$Goal(\ At(C_1,BCN) \wedge At(C_2,AGP)\ )$$

El siguiente plan es una posible solución:

[$Load(C_1,P_1,AGP)$,
$UnLoad(C_1,P_1,BCN)$,
$Fly(P_2,AGP,BCN)$,
$Fly(P_1,AGP,BCN)$,
$Load(C_2,P_2,AGP)$, F
$UnLoad(C_2,P_2,AGP)$]

Algoritmos de Planificación

Algoritmos de Planificación

- **GPS**, primer Planificador IA en 1963
- **UCPOP** 1992
- **Graphplan** 1995
- **SAT** 1996
- **Heurísticos** 1996, problema de planificación es resuelto mediante algoritmos de búsqueda donde las funciones heurísticas son extraídas automáticamente del problema.

Planificación como problemas de búsqueda

- Un problema Strips P = {F,O,I,G} determina un modelo de estado S(P) donde
  - Los estados $s \in S$ son colecciones de átomos de F
  - El estado inicial s_0 es I
  - Los estados objetivos s cumplen $G \subseteq s$
  - Las acciones a en A(s) son ops en O
  - El próximo estado s' = s − Del(a) + Add(a)
  - Los costos de las acciones c(a, s) son todos 1
- En la planificación como búsqueda heurística, el problema P es resuelto mediante algoritmos de búsqueda en el grafo asociado al modelo S(P)

Planificación como problemas de búsqueda

- P(s) es un problema de planificación definido como P={F,I,O,G} y π(s) es la solución encontrada para P+(s). La heurística h(s) que estima el coste del problema P(s) es definida como:

$$h(s) = cost(\pi(s)) = \sum_{a \in \pi(s)} cost(a)$$

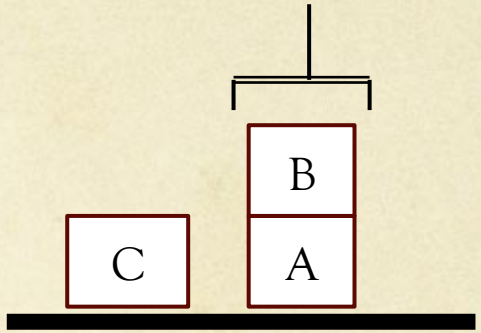
  - P+(s) es una relajación de P(s) eliminando las listas Del(a)

- La solución al problema viene determinado por la selección del siguiente estado que esté más cerca al objetivo (menor valor de la heurística)

Heurísticas en el mundo de bloques

- **Descripción de un estado:**

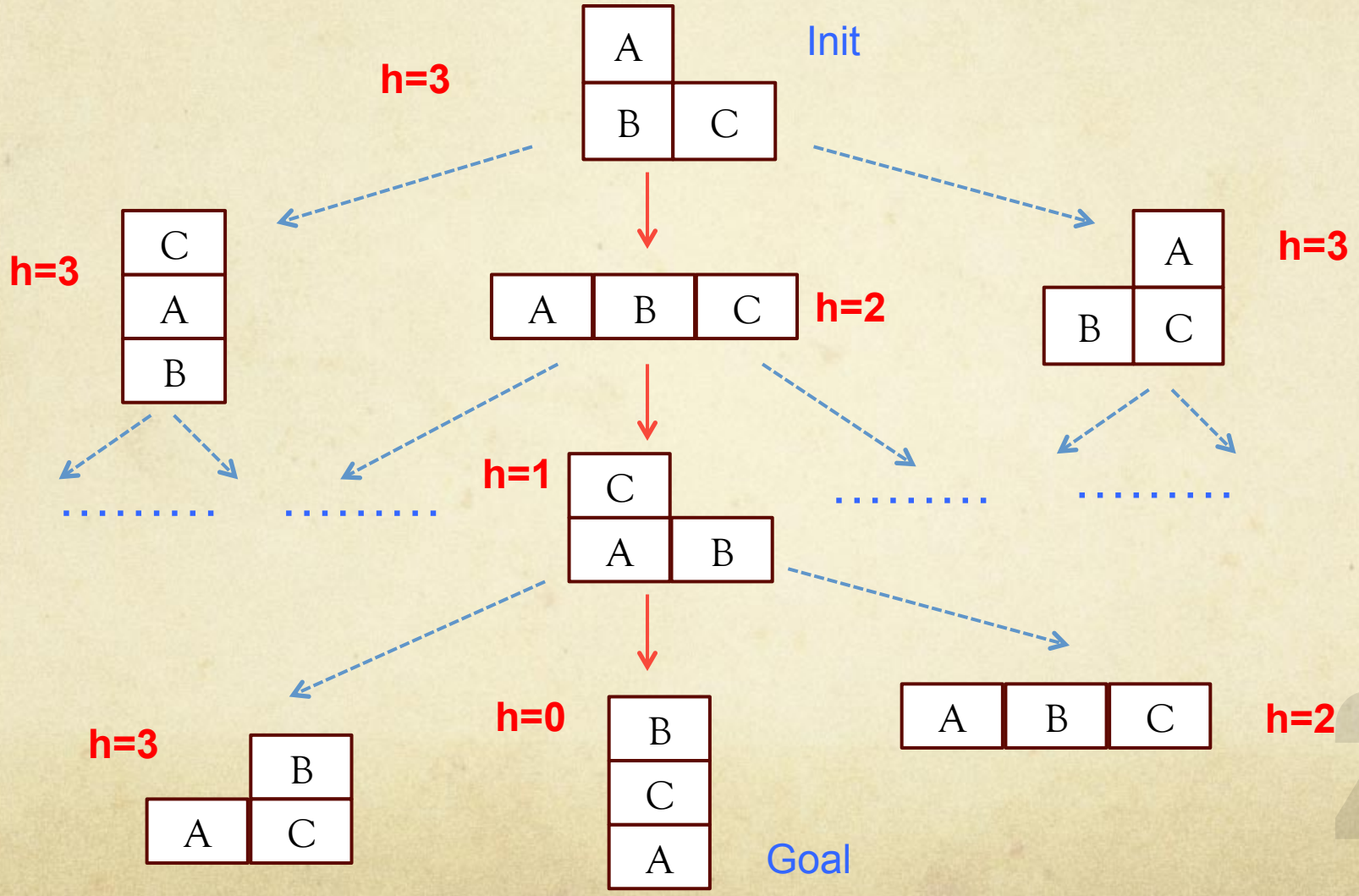
DESPEJADO(B), DESPEJADO(C),
SOBRE(B,A), SOBRELAMESA(C),
SOBRELAMESA(A), BRAZOLIBRE



- **Predicados en el mundo de bloques:**

DESPEJADO(x), el bloque x está despejado
BRAZOLIBRE, brazo libre
SOBRELAMESA(x), el bloque x está sobre la mesa
SOBRE(x,y), el bloque x está sobre el y
AGARRADO(x), el bloque X está sujeto por el brazo

Heurísticas en el mundo de bloques



Planificación como problemas de búsqueda

```
Solve(Nodes)
    if Empty Nodes -> Fail
    else Let Node = Select-Node Nodes
        Let Rest = Nodes - Node
        if Node is Goal -> Return Solution
        else Let Children = Expand-Node Node
            Let New-Nodes = Add-Nodes Children Rest
            Solve(New-Nodes)
```

- Podemos implementar distintos algoritmos dependiendo de instanciación de:
  - **Select-Node** Nodes
  - **Add-Nodes** New-Nodes Old-Nodes

Planificación como problemas de búsqueda

- **Profundidad** expande los n nodos más profundos primero
  - *Select-Node Nodes*: Selecciona Primer Nodo en Nodes
  - *Add-Nodes New Old*: Establece New antes Old
  - *Implementation:* Nodes es una Pila (LIFO)
- **Anchura** expande los n nodos menos profundos primero
  - *Select-Node Nodes:* Selecciona Primer Nodo en Nodes
  - *Add-Nodes New Old*: Establece New después Old
  - *Implementation*: Nodes es una Cola (FIFO)

Planificación como problemas de búsqueda

- **Mejores Primero** expande los mejores n nodos primero; min f(n)

  - *Select-Node Nodes*: Devuelve n en Nodes con min f(n)
  - *Add-Nodes New Old*: Realiza una mezcla ordenada
  - *Implementation*: Nodes es un Heap
  - Special cases
    - **Uniform cost/Dijkstra**: f(n) = g(n)
    - **A***: f(n) = g(n) + h(n)
    - **Greedy Best First**: f(n) = h(n)

Conclusión

Sumario

- Los sistemas de planificación son algoritmos de resolución de problemas que operan sobre representaciones de estados y acciones
- El lenguaje PDDL describe los estados inicial y objetivo como conjunciones de literales, y las acciones en términos de sus precondiciones y efectos
- Varios algoritmos de resolución propios para resolver planes y actualmente algoritmos de búsquedas basados en heurísticas.

Epílogo

- La NASA desarrolló un programa de planificación automática para los vehículos robóticos de Marte
  - Cuando se empleó en un entorno intenso y con plazos reducidos, consiguió un incremento del 10%-40% en los resultados científicos, en comparación con la operación sin ayuda de la IA
  - *Spirit* funcionó hasta que quedó atascado en una trampa de arena en 2009, y *Opportunity* todavía está operando (30 veces su vida estimada)
- Los fabricantes de ascensores incorporan planificadores automáticos en sus productos
  - Disminuyen la congestión y el tiempo de viaje, a la vez que ahorran energía

Bibliografía

Bibliografía

- Heuristics, Planning, and Cognition. In Heuristics, Probability and Causality: A Tribute to Judea Pearl. R. Dechter, H. Geffner, J. Halpern (Eds), College Publications, 2010.
- Hector Geffner, Course on Automated Planning.
- AIMA 3 Edición

Sistemas Inteligentes

José A. Montenegro Montes

monte@lcc.uma.es