# TP6

September 18, 2025

## 1 Temas Tratados en el Trabajo Práctico 6

- Modelado de problemas en espacios de estado.
- Algoritmos de planificación hacia adelante y hacia atrás.
- Representación y solución de problemas descritos en lenguaje STRIPS.
- Algoritmo GRAPHPLAN.
- Planificación con restricciones de tiempo y recursos.
- Caminos críticos y tiempos de relajación.

## 2 Ejercicios Teóricos

### 2.1 ¿En qué tipo de algoritmos se basa un planificador para encontrar el mejor camino a un estado solución?

**Los tipos principales de algoritmos que usa son:**

- Búsqueda hacia adelante (progression planning): Explora a partir del estado inicial aplicando acciones hasta alcanzar el objetivo. Se apoya en algoritmos como A* para encontrar la mejor secuencia de acciones.
- Búsqueda hacia atrás (regression planning): Empieza desde el estado objetivo y busca qué acciones lo pueden generar, retrocediendo hasta llegar al estado inicial.
- STRIPS y Graphplan: STRIPS da la representación formal de estados, acciones y objetivos. Graphplan construye un grafo de planificación y utiliza búsqueda en ese grafo para encontrar un plan válido y, en algunos casos, óptimo.
- Planificación con optimización: Cuando se consideran tiempos y recursos, se aplican algoritmos de búsqueda con heurísticas de optimización (por ejemplo, caminos críticos o relajación temporal).

### 2.2 ¿Qué tres elementos se encuentran dentro de una acción formulada en lenguaje STRIPS? Describa brevemente qué función cumple cada uno.

**En el lenguaje STRIPS (Stanford Research Institute Problem Solver), cada acción se describe con tres elementos principales:**

- Nombre y parámetros de la acción: Identifican qué acción es y sobre qué objetos actúa. Ejemplo: Volar(avión1, Mendoza, BsAs). Función: dar un identificador claro y los argumentos necesarios para aplicar la acción.
- Precondiciones (PRECOND): Son los literales que deben cumplirse en el estado actual para que la acción pueda ejecutarse. Ejemplo: En(avión1, Mendoza) $\wedge$ Aeropuerto(Mendoza) $\wedge$ Aeropuerto(BsAs). Función: asegurar que la acción solo se puede ejecutar si el mundo está en un estado adecuado.
- Efectos (EFECTO): Son los cambios que produce la acción en el mundo, agregando hechos nuevos y eliminando los que dejan de ser ciertos. Ejemplo: $\neg$En(avión1, Mendoza) $\wedge$ En(avión1, BsAs). Función: actualizar el estado para reflejar la nueva situación tras ejecutar la acción.

### 2.3 Describa las ventajas y desventajas de desarrollar un algoritmo de planificación hacia adelante y hacia atrás en el espacio de estados.

**La planificación hacia adelante comienza desde el estado inicial y aplica acciones sucesivas hasta alcanzar el objetivo.**

- Ventajas: es intuitiva, sigue el mismo camino que recorrería el agente en la realidad y siempre garantiza que las acciones ejecutadas son aplicables.
- Desventajas: el espacio de búsqueda puede ser muy grande y se pueden explorar muchos caminos irrelevantes que no acercan al objetivo.

**La planificación hacia atrás parte del objetivo y busca qué acciones podrían producirlo, retrocediendo hasta llegar al estado inicial.**

- Ventajas: reduce la búsqueda porque se concentra en acciones directamente relacionadas con el objetivo y evita caminos innecesarios.
- Desventajas: puede ser difícil determinar las precondiciones exactas y verificar si son alcanzables desde el estado inicial, además de que no siempre es intuitiva de implementar.

### 2.4 Considere el problema de ponerse uno mismo zapatos y medias. Aplique GRAPHPLAN a este problema y muestre la solución obtenida. Muestre el plan de orden parcial que es solución e indique cuántas linealizaciones diferentes existen para el plan de orden parcial.

**PROBLEMA: Ponerse medias y zapatos** ESTADO INICIAL: - Ninguna media puesta - Ningún zapato puesto

OBJETIVO: - Tener ambas medias puestas - Tener ambos zapatos puestos

ACCIONES DISPONIBLES (STRIPS-like)

- Acción: PonerMedia(izquierda)

  Precondiciones: $\neg$Media izquierda

  Efectos: Media izquierda puesta

- Acción: PonerMedia(derecha)

  Precondiciones: ¬Media derecha

  Efectos: Media derecha puesta

- Acción: PonerZapato(izquierdo)

  Precondiciones: Media izquierda puesta $\wedge$ ¬Zapato izquierdo

  Efectos: ZapatoIzquierdo puesto

- Acción: PonerZapato(derecho)

  Precondiciones: Media derecha puesta $\wedge$ ¬Zapato derecho

  Efectos: Zapato derecho puesto

GRAPHPLAN: Expansión por niveles

Nivel S0 (estado inicial): - Pie izquierdo desnudo (¬Media izquierda $\wedge$ ¬Zapato izquierdo) - Pie derecho desnudo (¬Media derecha $\wedge$ ¬Zapato derecho)

Nivel A0 (acciones aplicables desde S0): - PonerMedia(izquierda) - PonerMedia(derecha)

Nivel S1 (nuevos estados alcanzables): - Media izquierda puesta - Media derecha puesta

Nivel A1 (acciones aplicables desde S1): - PonerZapato(izquierdo) - PonerZapato(derecho)

Nivel S2 (objetivo alcanzado): - Zapato izquierdo puesto - Zapato derecho puesto

PLAN DE ORDEN PARCIAL

PonerMedia(izquierda) → PonerZapato(izquierdo) → PonerMedia(derecha) → PonerZapato(derecho)

// Restricciones de precedencia:

// - Cada zapato depende de su respectiva media

// - No hay orden entre pie izquierdo y derecho

LINEALIZACIONES POSIBLES

Orden 1: MediaIzq → ZapatoIzq → MediaDer → ZapatoDer

Orden 2: MediaIzq → MediaDer → ZapatoIzq → ZapatoDer

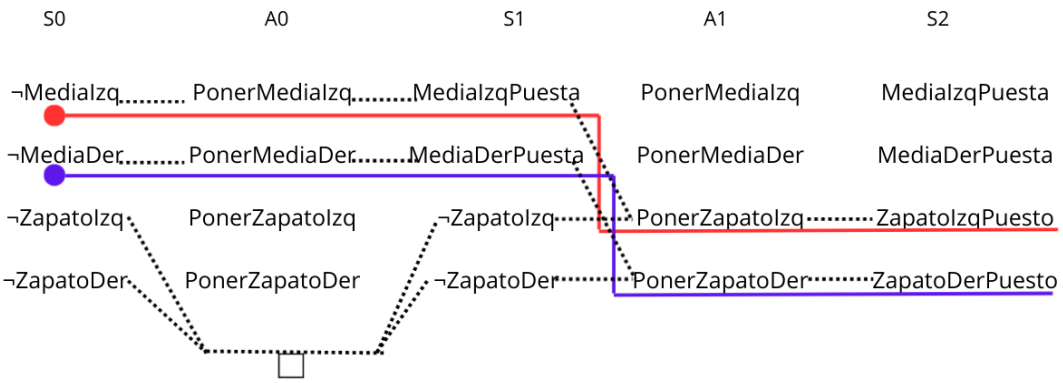
Orden 3: MediaDer → ZapatoDer → MediaIzq → ZapatoIzq

Orden 4: MediaDer → MediaIzq → ZapatoIzq → ZapatoDer

Orden 5: MediaIzq → MediaDer → ZapatoDer → ZapatoIzq

Orden 6: MediaDer → MediaIzq → ZapatoDer → ZapatoIzq

TOTAL: 6 linealizaciones válidas.




**2.5 Se requiere ensamblar una máquina cuyas piezas están identificadas con las letras A, B, C, D y E. El tiempo que se tarda en ensamblar cada pieza es:**

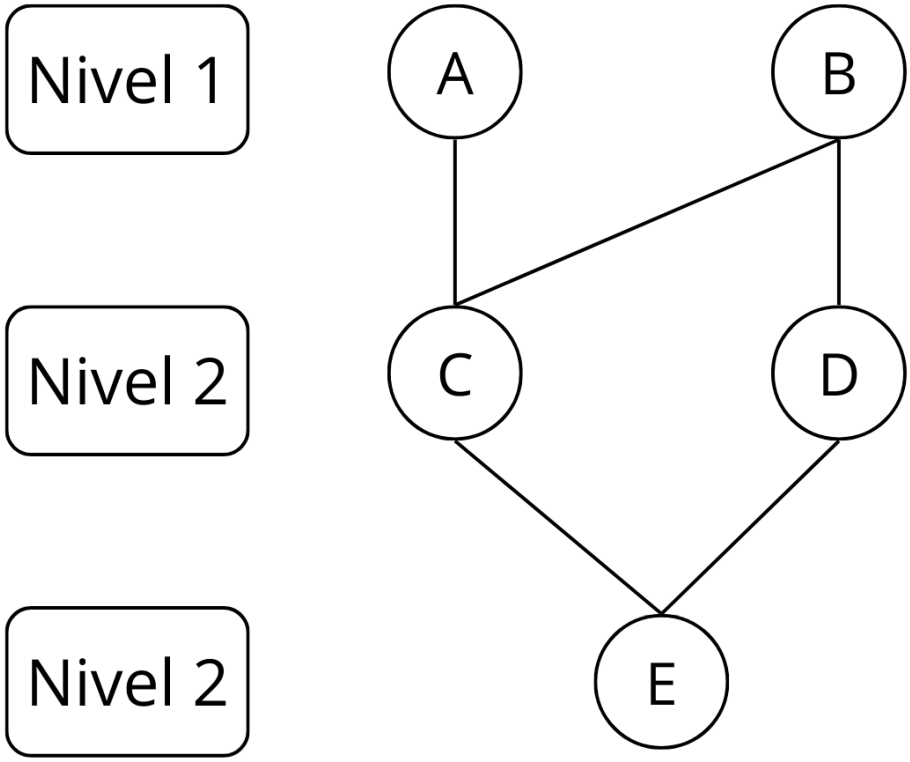
- A: 2 semanas
- B: 1 semana
- C: 4 semanas
- D: 3 semanas
- E: 5 semanas

El orden de ensamblaje de cada pieza requiere que:

- A esté realizado antes que C
- B esté realizado antes que C
- B esté realizado antes que D
- C esté realizado antes que E
- D esté realizado antes que E

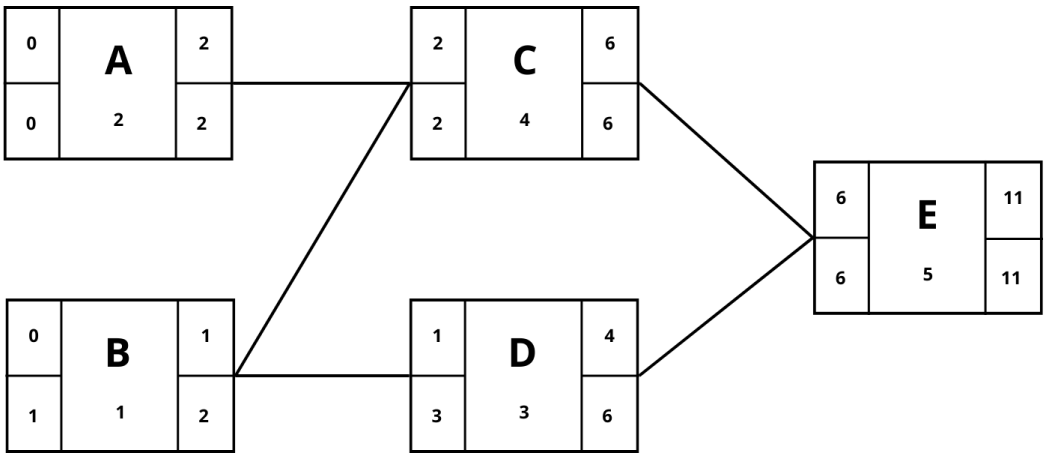
Con esta información:

5.1 Plan de Orden Parcial:




Las tareas iniciales son A y B; C y D son intermedias; E es la final.

5.2 Camino Crítico



Primero calculamos los intervalo de tiempo previo al inicio para obtener la duración total del proyecto:

A: ES $= 0 \rightarrow$ EF $= 0 + 2 = 2$

B: ES $= 0 \rightarrow$ EF $= 0 + 1 = 1$

C: ES $= 2 \rightarrow$ EF $= 2 + 4 = 6$

D: ES $= 1 \rightarrow$ EF $= 1 + 3 = 4$

E: ES $= 6 \rightarrow$ EF $= 6 + 5 = 11$

La duración del proyecto = 11 semanas (EF de E).

Backward pass empezando LF(E) = 11:

E: LF $= 11 \rightarrow$ LS $= 11 - 5 = 6$

C: LF $= 6 \rightarrow$ LS $= 6 - 4 = 2$

D: LF $= 6 \rightarrow$ LS $= 6 - 3 = 3$

A: LF $= 2 \rightarrow$ LS $= 2 - 2 = 0$

B: LF $= 2 \rightarrow$ LS $= 2 - 1 = 1$

El camino crítico es A $\rightarrow$ C $\rightarrow$ E.

5.3 Tiempos de relajación Los tiempos de relajación son (slack = LS − ES = LF − EF):

A: ES=0, EF=2, LS=0, LF=2 $\rightarrow$ slack $= 0$

B: ES=0, EF=1, LS=1, LF=2 $\rightarrow$ slack $= 1$

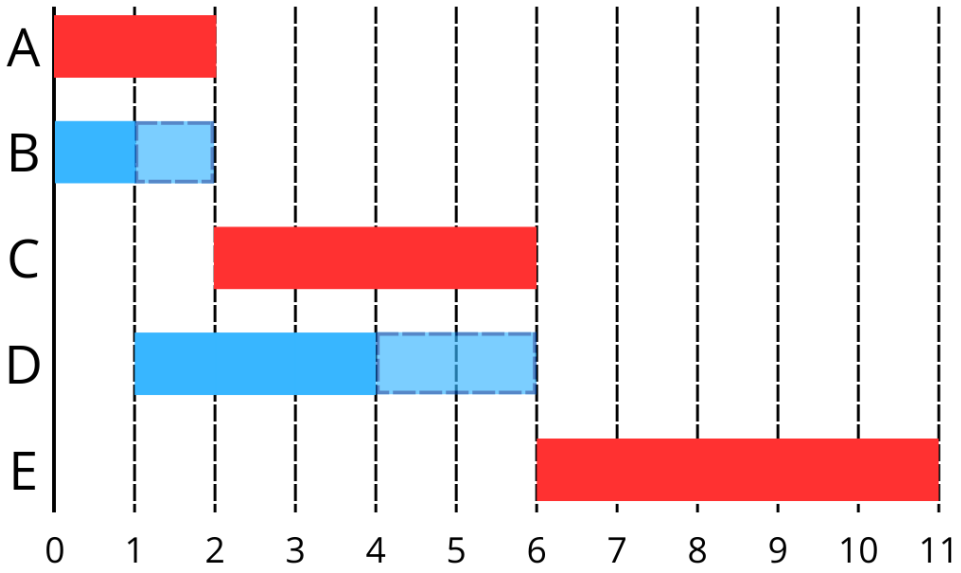
C: ES=2, EF=6, LS=2, LF=6 $\rightarrow$ slack $= 0$

D: ES=1, EF=4, LS=3, LF=6 $\rightarrow$ slack $= 2$

E: ES=6, EF=11, LS=6, LF=11 $\rightarrow$ slack $= 0$

B puede retrasarse hasta 1 semana sin afectar la fecha final; D hasta 2 semanas; A, C y E no pueden retrasarse sin retrasar el proyecto.

5.4 Diagrama temporal




## 3 Ejercicios de Implementación

**3.1 Suponga que tiene un robot de oficina capaz de moverse y tomar y depositar objetos. El robot solo puede tener un objeto a la vez, pero puede conseguir una *caja* en la que depositar varios objetos. Suponga que programa al robot para *ir a la tienda* a comprarle un *café* y en el camino de vuelta tome una *carta* del *buzón* de la oficina para para que se la traiga junto con el café. Describa en lenguaje STRIPS:**

---

**El dominio del robot (nombre, predicados y acciones que puede hacer el robot).** Dominio: MundoRobot

**Predicados:**

- En(objeto, lugar)
- Sosteniendo(objeto)
- SosteniendoCaja()
- Tiene(objeto, contenedor)
- Vacio()

**Acciones:** * Ir(lugar1, lugar2): * Precondiciones: En(objeto, lugar1) * Efectos: ~En(objeto, lugar1) En(objeto, lugar2)

- Tomar(objeto, lugar):
  - Precondiciones: En(robot, lugar) En(objeto, lugar) Vacio()
  - Efectos: ~En(objeto, lugar) ~Vacio() Sosteniendo(objeto)
- Dejar(objeto, lugar)
  - Precondiciones: En(robot, lugar) Sosteniendo(objeto)
  - Efectos: ~Sosteniendo(objeto) En(objeto, lugar) Vacio()
- TomarCaja(lugar)
  - Precondiciones: En(robot, lugar) En(caja, lugar) Vacio()
  - Efectos: ~En(caja, lugar) ~Vacio() SosteniendoCaja()
- PonerEnCaja(objeto, lugar)
  - Precondiciones: En(robot, lugar) En(objeto, lugar) SosteniendoCaja()
  - Efectos: ~En(objeto, lugar) Tiene(objeto, caja)

---

**El problema que se quiere resolver (estado inicial, estado objetivo y objetos del mundo representados).**

- Objetos del Mundo:
  - Lugares: oficina, tienda, buzon
  - Objetos: robot, cafe, carta
  - Contenedores: caja
- Estado Inicial: En(robot, oficina) En(caja, oficina) En(cafe, tienda) En(carta, buzon) Vacio()
- Estado Objetivo: En(robot, oficina) SosteniendoCaja() Tiene(cafe, caja) Tiene(carta, caja)

---

**Introduzca el código desarrollado en los puntos anteriores en el planificador online y obtenga el plan de acción que tomará el robot para cumplir lo solicitado.**

**Descripción del Dominio PDDL: "MundoRobot"**

```
(define (domain MundoRobot)
  (:requirements :strips :typing)

  (:types
    lugar contenedor - object
  )

  (:predicates
    (en ?o - object ?l - lugar)
    (sosteniendo ?o - object)
    (sosteniendo-caja)
    (tiene ?o - object ?c - contenedor)
    (vacio)
  )
```

```
(:action Ir
  :parameters (?lugar1 - lugar ?lugar2 - lugar)
  :precondition (en robot ?lugar1)
  :effect (and
    (not (en robot ?lugar1))
    (en robot ?lugar2)
  )
)

(:action Tomar
  :parameters (?o - object ?l - lugar)
  :precondition (and
    (en robot ?l)
    (en ?o ?l)
    (vacio)
  )
  :effect (and
    (not (en ?o ?l))
    (not (vacio))
    (sosteniendo ?o)
  )
)

(:action Dejar
  :parameters (?o - object ?l - lugar)
  :precondition (and
    (en robot ?l)
    (sosteniendo ?o)
  )
  :effect (and
    (not (sosteniendo ?o))
    (en ?o ?l)
    (vacio)
  )
)

(:action TomarCaja
  :parameters (?l - lugar)
  :precondition (and
    (en robot ?l)
    (en caja ?l)
    (vacio)
  )
  :effect (and
    (not (en caja ?l))
    (not (vacio))
    (sosteniendo-caja)
```

```
    )
  )

  (:action PonerEnCaja
    :parameters (?o - object ?l - lugar)
    :precondition (and
      (en robot ?l)
      (en ?o ?l)
      (sosteniendo-caja)
    )
    :effect (and
      (not (en ?o ?l))
      (tiene ?o caja)
    )
  )
)
```

**Descripción del Problema PDDL: "ProblemaEncargo"**

```
(define (problem ProblemaEncargo)
  (:domain MundoRobot)

  (:objects
    robot cafe carta - object
    caja - contenedor
    oficina tienda buzon - lugar
  )

  (:init
    (en robot oficina)
    (en caja oficina)
    (en cafe tienda)
    (en carta buzon)
    (vacio)
  )

  (:goal
    (and
      (en robot oficina)
      (tiene cafe caja)
      (tiene carta caja)
      (sosteniendo-caja)
    )
  )
)
```

**Plan de Accion tomado por el Robot** (TomarCaja oficina)

(Ir oficina buzon)

(PonerEnaja carta buzon)

(Ir buzon tienda)

(PonerEnCaja cafe tienda)

(Ir tienda oficina)

; cost = 6 (unit cost)

---

## 4 Bibliografía

Russell, S. & Norvig, P. (2004) *Inteligencia Artificial: Un Enfoque Moderno.* Pearson Educación S.A. (2a Ed.) Madrid, España

Poole, D. & Mackworth, A. (2023) *Artificial Intelligence: Foundations of Computational Agents.* Cambridge University Press (3a Ed.) Vancouver, Canada