Inteligencia Artificial II

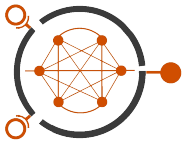
TP 1 - Guía de resolución

Optimización con A*

Martin Marchetta
martin.marchetta@ingenieria.uncuyo.edu.ar

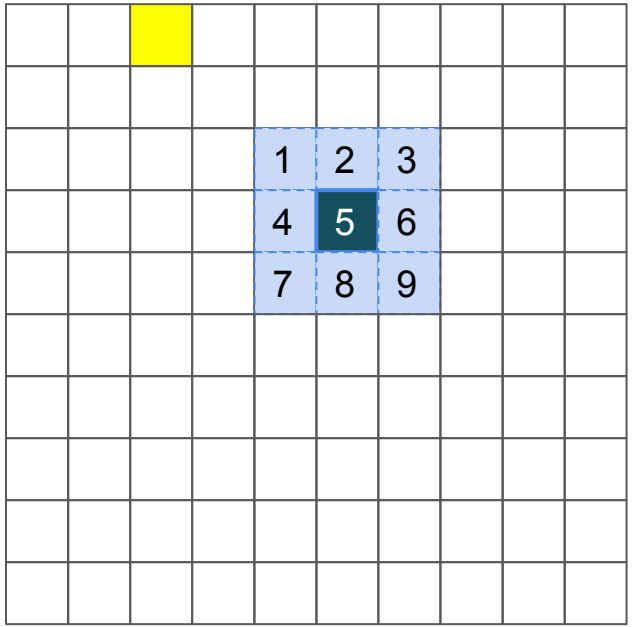
UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

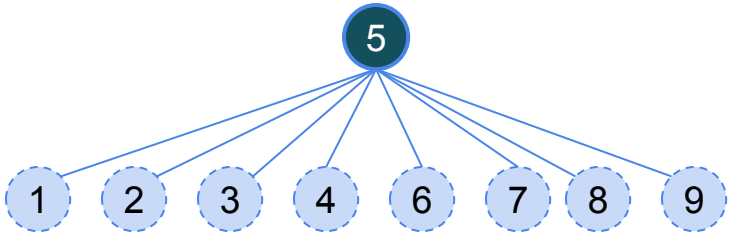
FACULTAD
DE INGENIERÍA



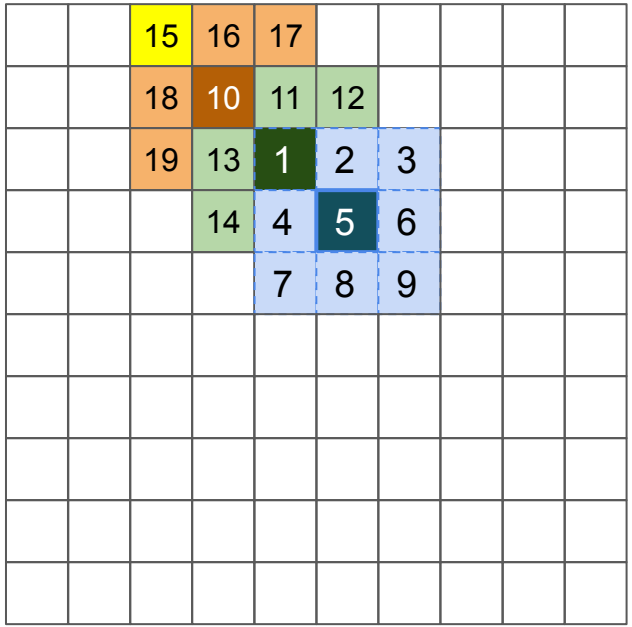
LABSIN

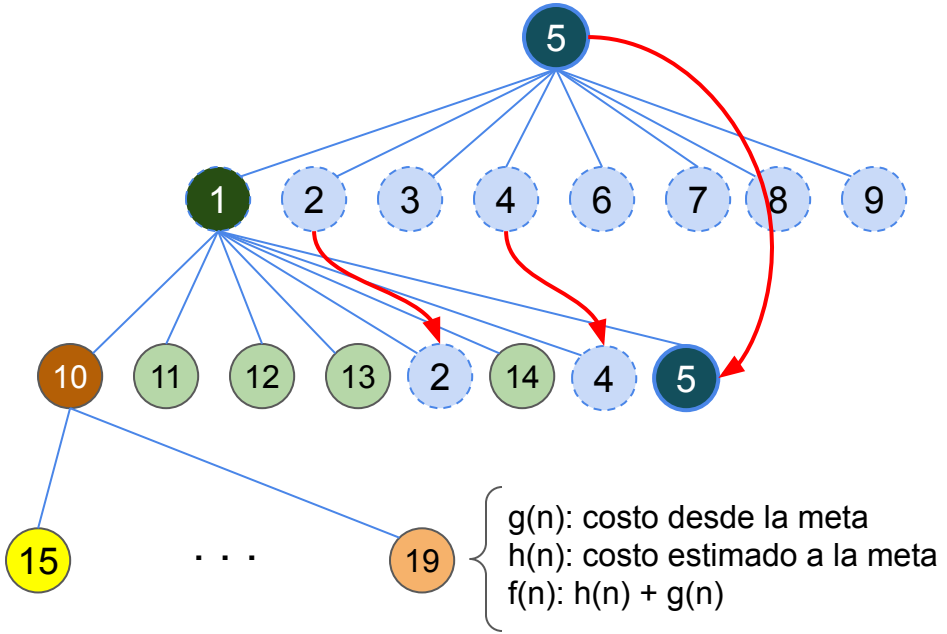
Caminos con A*





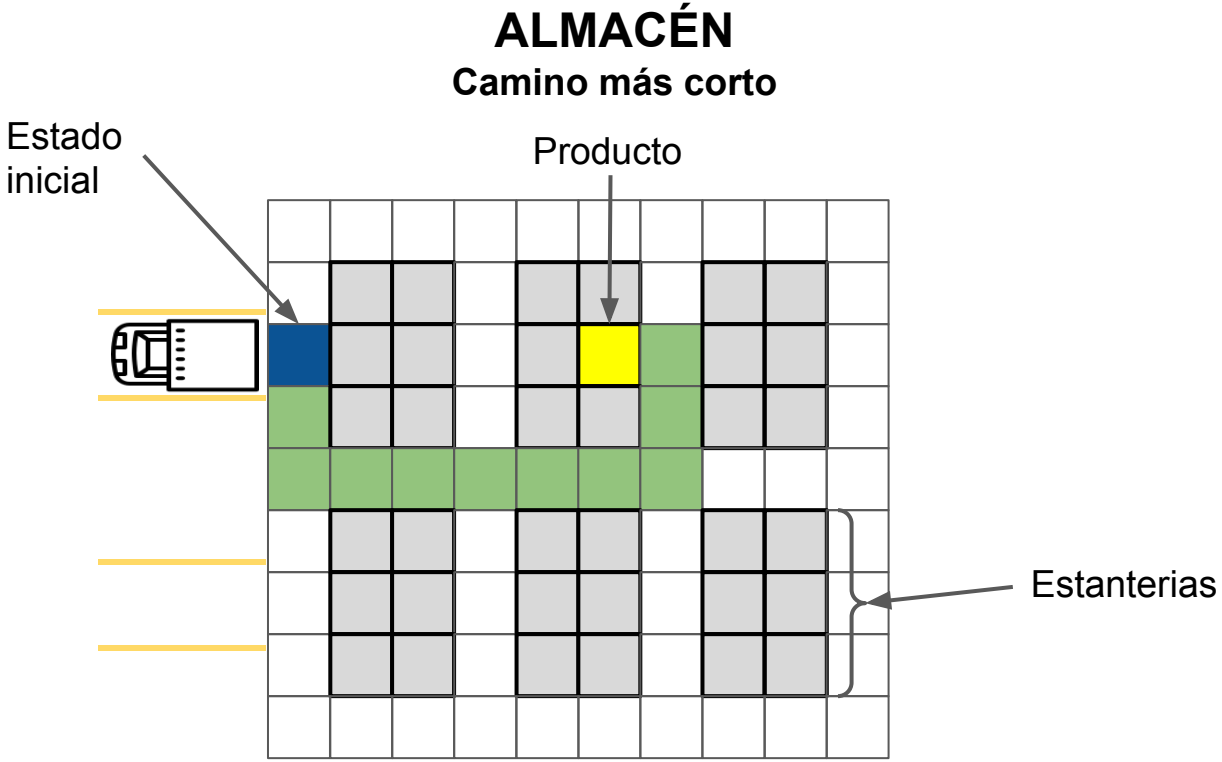
Caminos con A*





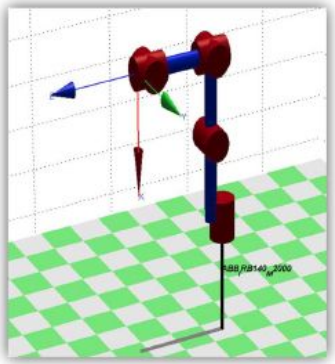
En cada iteración se expande el nodo pendiente con menor valor de f(n)

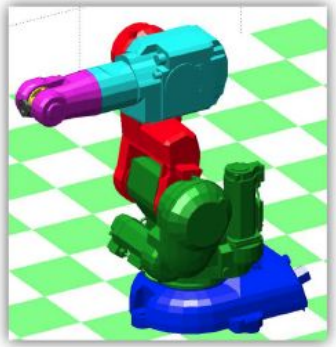
Caminos con A*



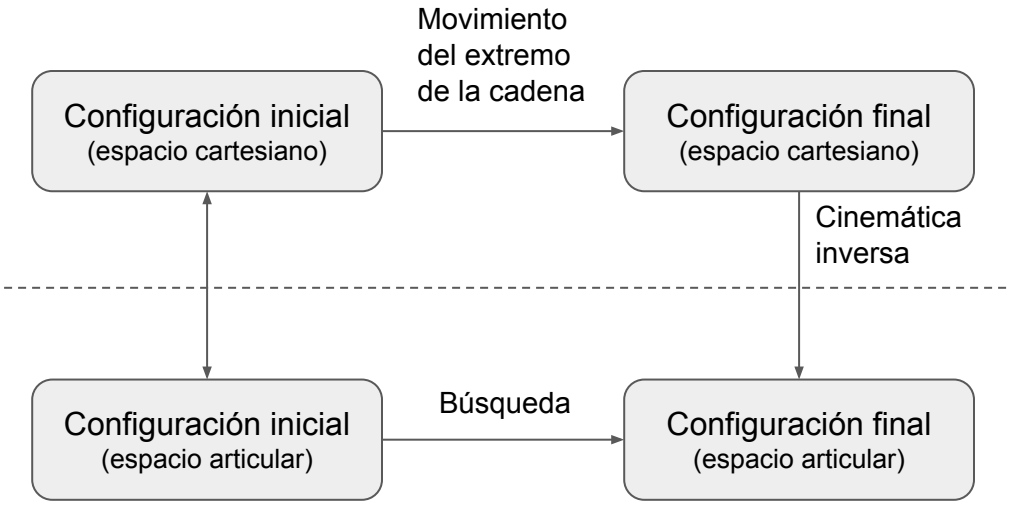
Caminos con A*

Cadena cinemática



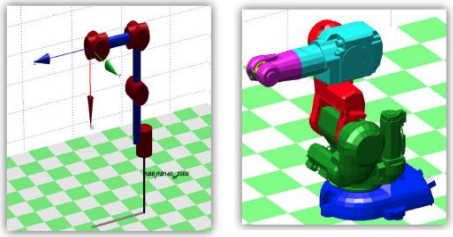


CONSIGNA



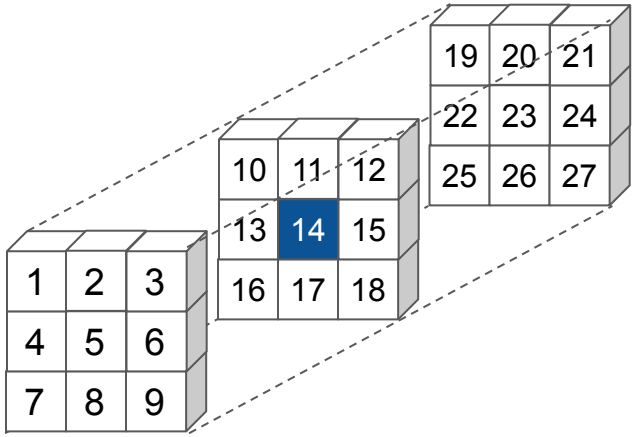
CONTROL

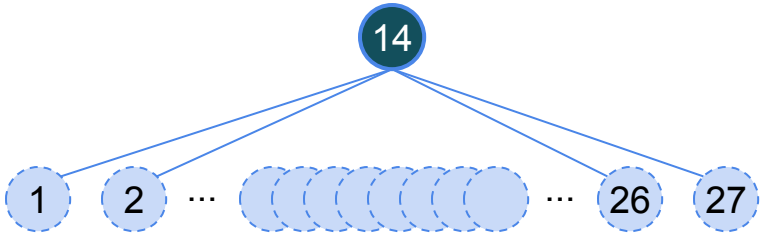
Caminos con A*



Configuración inicial (espacio articular) → Búsqueda (espacio articular) → Configuración final (espacio articular)

Ejemplo con 3 articulaciones → Búsqueda 3D en el espacio articular





Con N articulaciones → Búsqueda N-Dimensional

Caminos con A*

- Consejos de implementación
  - Utilizar bucles para generar todos los estados sucesores de un nodo
    - Generalmente un bucle por cada dimensión
      - Ej: para optimizar un camino en el plano, se utilizarían 2 bucles: i y j, para 3 articulaciones se utilizarían 3 bucles, etc)
      - Para el plano el primer bucle va de i-1 a i+1, y el segundo de j-1 a j+1, donde el nodo actual es (i, j)
      - Tratar de evitar repetir el código por cada dimensión a explorar
  - Un algoritmo se basa en una lista abierta y una lista cerrada
    - Lista abierta: nodos que aún están “en la frontera” para ser explorados
    - Lista cerrada: nodos que ya fueron explorados
    - La lista cerrada sirve para no explorar subcaminos más de una vez

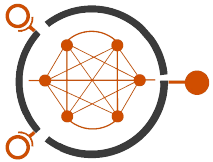
Inteligencia Artificial II

TP 1 - Guía de resolución

Martin Marchetta
martin.marchetta@ingenieria.uncuyo.edu.ar

UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

FACULTAD
DE INGENIERÍA



LABSIN

Inteligencia Artificial II

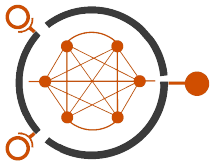
TP 1 - Guía de resolución

Optimización híbrida con Temple Simulado y A*

Martin Marchetta
martin.marchetta@ingenieria.uncuyo.edu.ar

UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

FACULTAD
DE INGENIERÍA



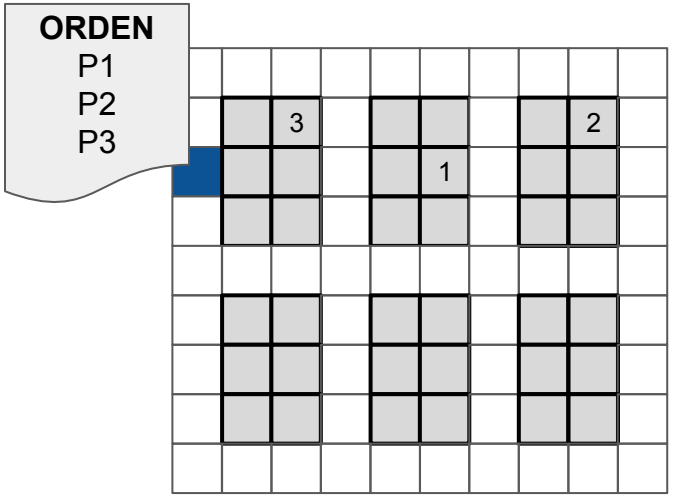
LABSIN

Optimización híbrida: Temple Simulado + A*

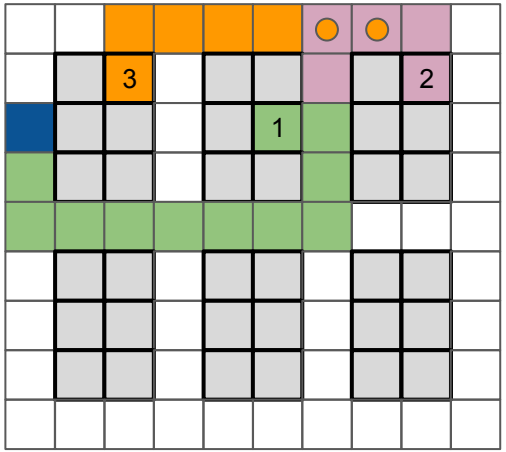
UNCUYO UNIVERSIDAD NACIONAL DE CUYO · FACULTAD DE INGENIERÍA · LABSIN

ALMACÉN

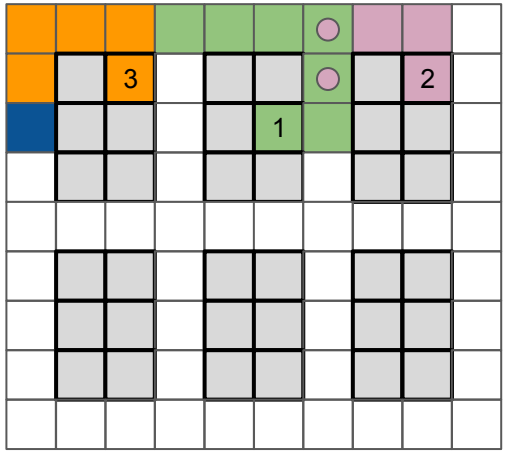
Orden de pedido: camino más corto multi-producto



Ubicación de los productos



(a) Picking [P1, P2, P3]
Costo: 20



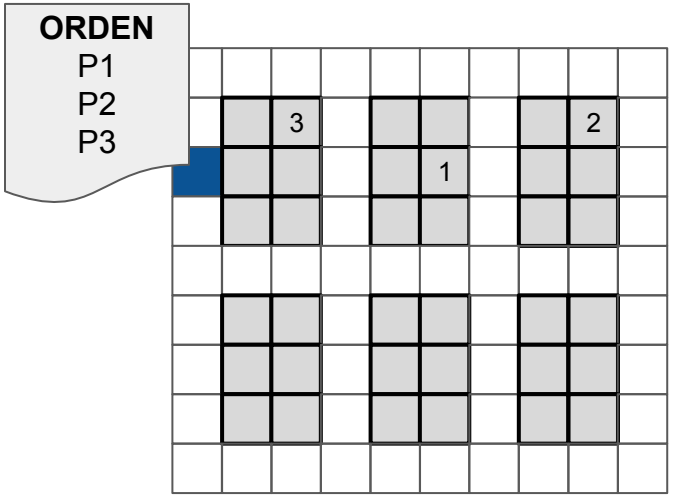
(b) Picking [P3, P1, P2]
Costo: 14

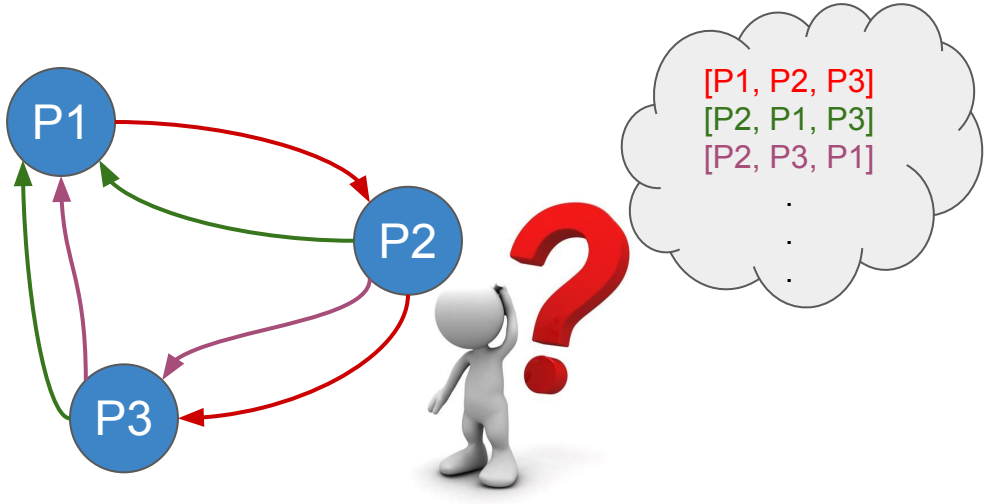
Optimización híbrida: Temple Simulado + A*

UNCUYO UNIVERSIDAD NACIONAL DE CUYO · FACULTAD DE INGENIERÍA · LABSIN

ALMACÉN

Orden de pedido: camino más corto multi-producto





Optimización en 2 niveles:
(a) orden de productos; (b) camino entre productos

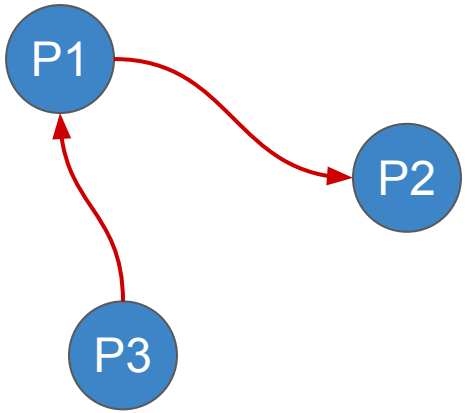
Optimización híbrida: Temple Simulado + A*

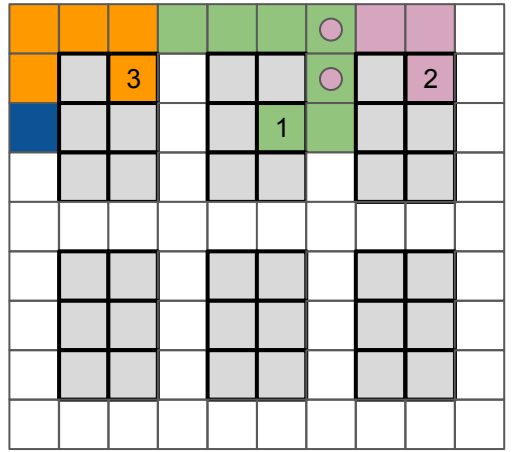
ALMACÉN

Orden de pedido: camino más corto multi-producto

ORDEN
P1
P2
P3

Temple simulado con permutaciones (orden de picking de productos) + A* (cálculo de E) (camino entre productos)



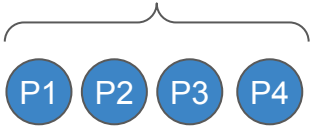


Optimización híbrida: Temple Simulado + A*

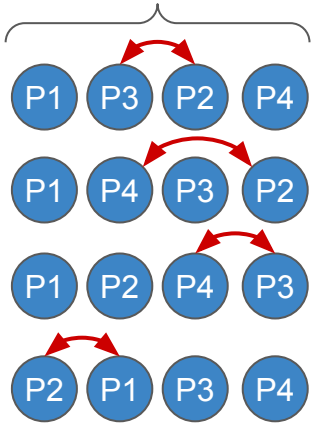
Modelo de estado

Temple Simulado vs. A*

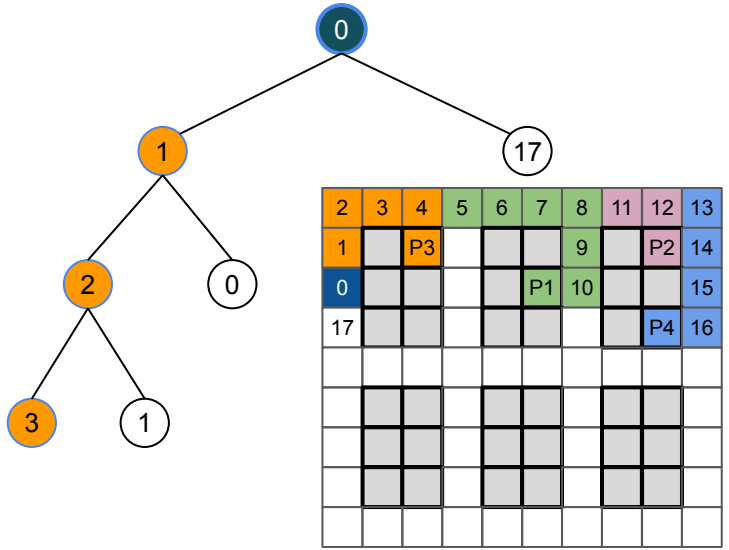
Estado actual



Estados vecinos
(1 sola permutación)



ORDEN
P1
P2
P3
P4

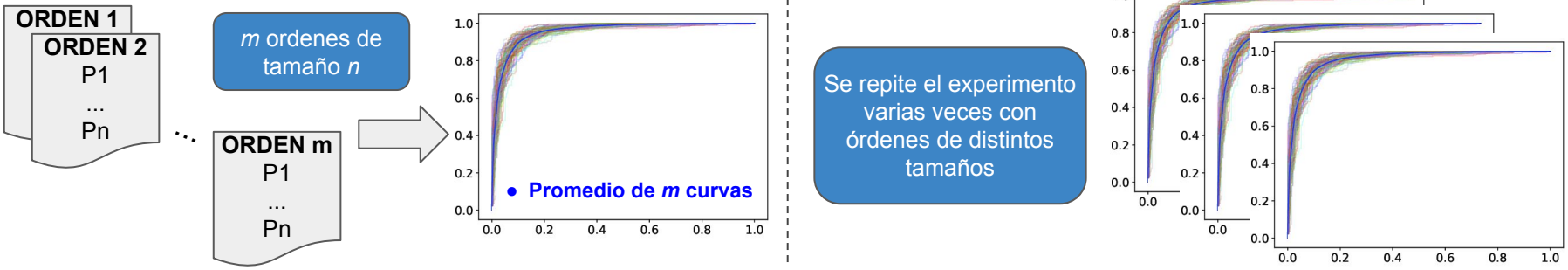


TEMPLE SIMULADO: Cada estado es una ruta completa

A*: Cada estado es un paso en una ruta (construye rutas)

Optimización híbrida: Temple Simulado + A*

- Afinamiento del Temple Simulado
  - Función de enfriamiento
  - Valor inicial de *t*
- Dada una configuración (un valor de t y una función de enfriamiento)
  - Correr varias veces el algoritmo con órdenes del mismo tamaño y promediar resultados
  - Repetir el experimento varias veces, utilizando cada vez órdenes de distinto tamaño
  - Se busca eliminar el factor “suerte” dada la aleatoriedad del algoritmo



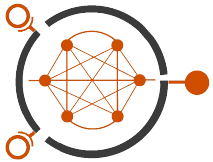
Inteligencia Artificial II

TP 1 - Guía de resolución

Martin Marchetta
martin.marchetta@ingenieria.uncuyo.edu.ar

UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

FACULTAD
DE INGENIERÍA



LABSIN

Inteligencia Artificial II

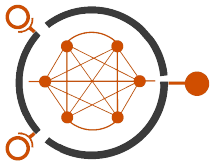
TP 1 - Guía de resolución

Optimización híbrida: Algoritmos Genéticos, Temple Simulado y A*

Martin Marchetta
martin.marchetta@ingenieria.uncuyo.edu.ar

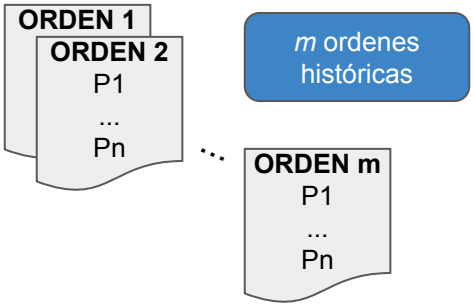
UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

FACULTAD
DE INGENIERÍA



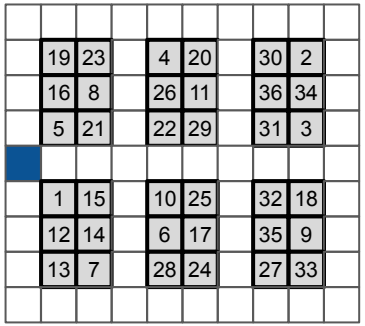
LABSIN

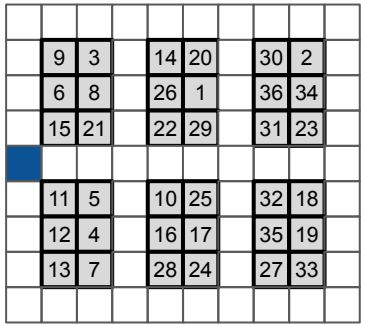
Optimización del almacén: Algoritmos Genéticos

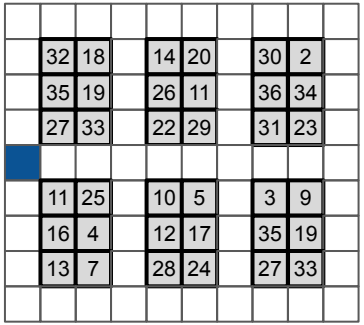


m ordenes históricas

Donde pongo los productos?







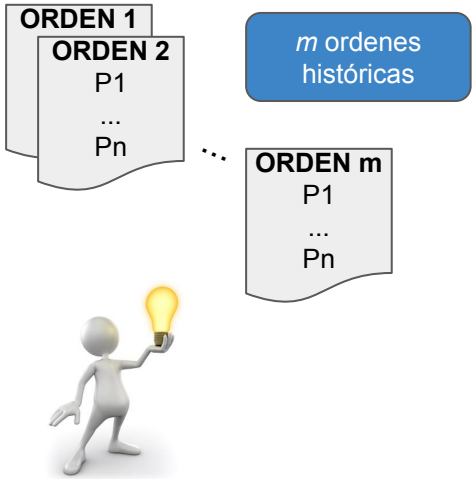
Optimización híbrida: Temple Simulado + A*

UNCUYO UNIVERSIDAD NACIONAL DE CUYO · FACULTAD DE INGENIERÍA · LABSIN

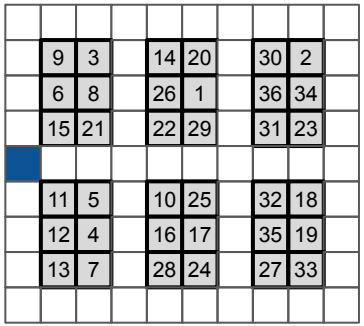
ALMACÉN

Optimización de ubicación de productos

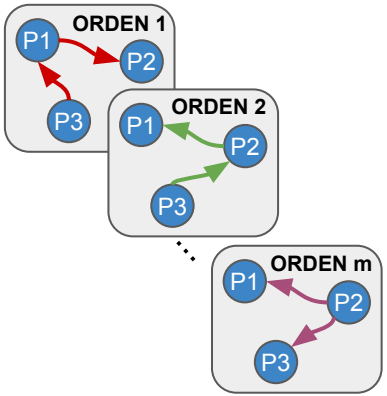
Algoritmo Genético (ubicación de productos) + **Temple simulado (permutaciones)** (orden de picking de productos) + **A* (cálculo de E)** (camino entre productos)



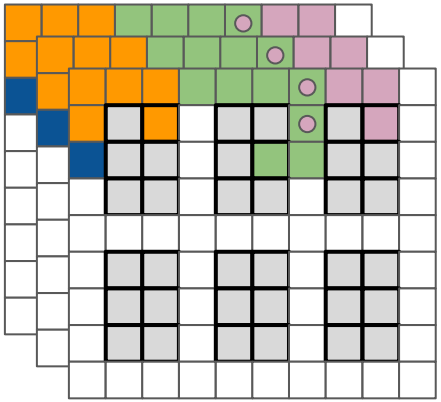
Algoritmo Genético



Temple simulado

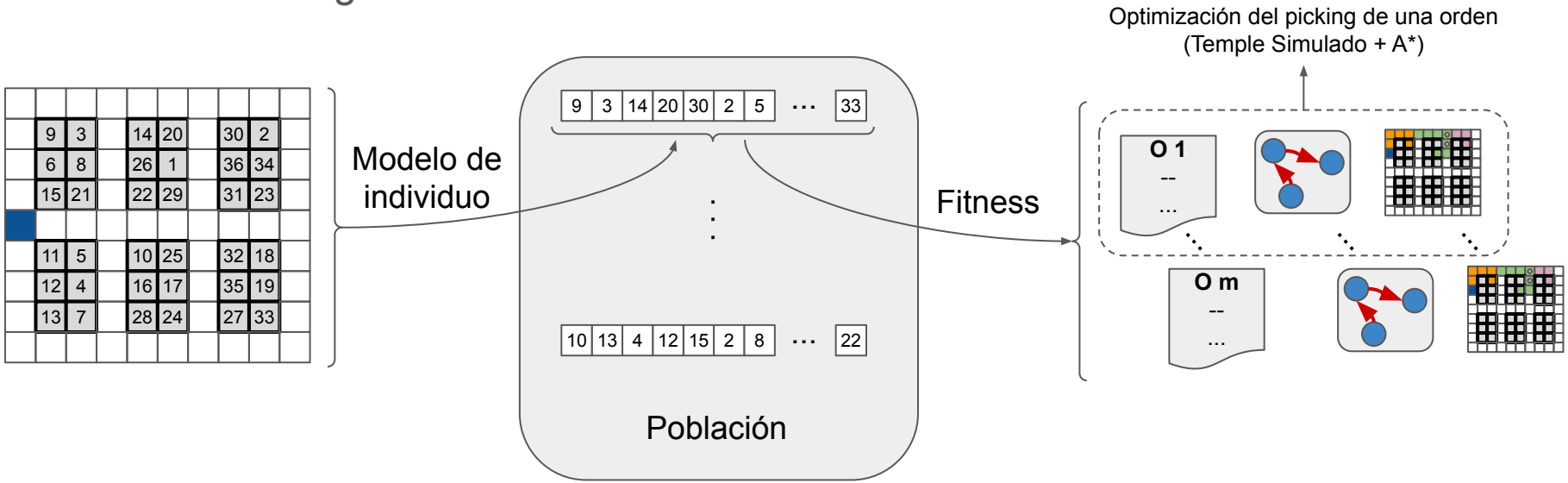


A*



Optimización híbrida: Temple Simulado + A*

- Diseño del Algoritmo Genético



- Mecanismos evolutivos con permutaciones

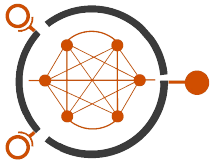
Inteligencia Artificial II

TP 1 - Guía de resolución

Martin Marchetta
martin.marchetta@ingenieria.uncuyo.edu.ar

UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

FACULTAD
DE INGENIERÍA



LABSIN

Inteligencia Artificial II

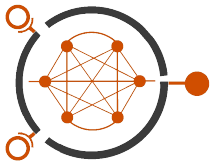
TP 1 - Guía de resolución

Scheduling con Constraint Satisfaction

Martin Marchetta
martin.marchetta@ingenieria.uncuyo.edu.ar

UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

FACULTAD
DE INGENIERÍA



LABSIN

Scheduling con Constraint Satisfaction

- Datos de entrada
  - Tareas a realizar
    - Identificador
    - Duración
    - Requerimientos de máquina (tipo)
  - Máquinas disponibles
    - Identificador
    - Tipo

- Suposiciones
  - Cada tarea requiere una sola máquina
  - Cada máquina sólo puede hacer una tarea en un momento determinado

- Objetivo
  - Determinar el período de inicio de cada tarea de manera de no exceder la capacidad de máquina

Scheduling con Constraint Satisfaction

- Diseño
  - Tipos de variables
    - TS_i: Período en que se inicia la tarea *i*
    - TM_i: Máquina utilizada en la tarea *i*
  - Dominio de cada variable
    - TS_i
      - Valores enteros que representan una discretización en períodos (que pueden representar horas por ej)
      - Ej: si la suma total de las duraciones de todas las tareas es 30 períodos, podemos suponer que el dominio de TS_i será el conjunto [1, 2, …, 30]
    - TM_i
      - Lista de máquinas del tipo requerido por la tarea *i*

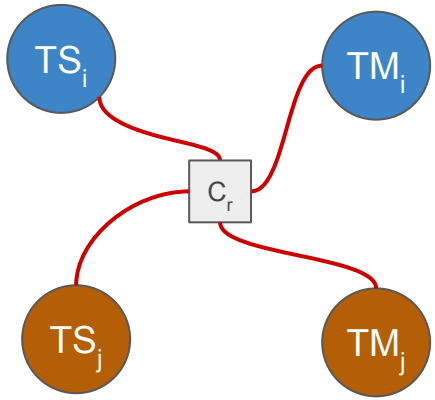
Scheduling con Constraint Satisfaction

- Diseño (cont.)
  - Restricciones
    - Restricciones globales: involucran más de 2 variables (4 variables en esta formulación propuesta)
    - Cada restricción C_r involucra
      - TS_i
      - TM_i
      - TS_j
      - TM_j
    - *i* y *j* son un par de tareas que requieren el mismo tipo de máquina
    - Opcionalmente se puede poner una deadline para cada tarea:

$$TS_i <= deadline_i - D_i$$

donde $deadline_i$ y D_i (duración de la tarea i) no son variables, sino constantes que vienen dados por los datos de entrada)

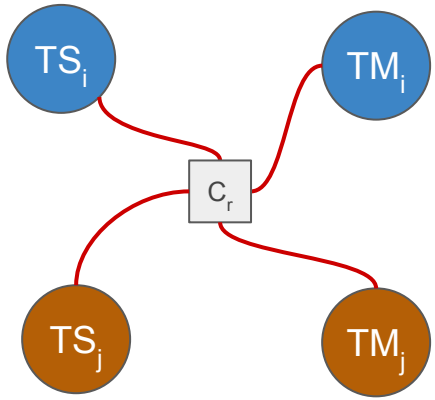
Grafo de restricciones modelo (el real tendría instancias de cada variable TS1, TM1, TS2, TM2, etc.)



Scheduling con Constraint Satisfaction

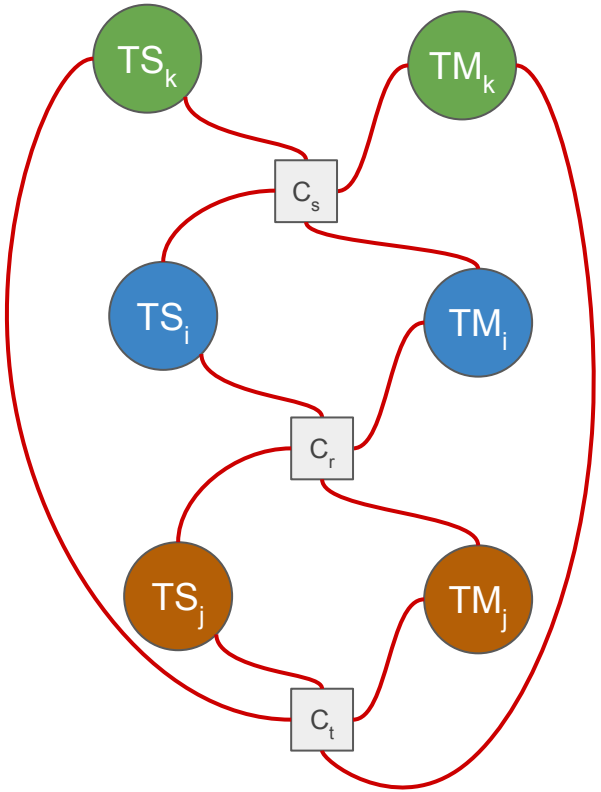
- Propagación de restricciones
  - Dado que estas restricciones involucran 4 variables, sólo cuando se han asignado 3 de ellas se puede propagar la restricción a la 4º)
  - Ej:
    - asignando TS_i=1, TM_i=M1, TS_j=1, se debe eliminar la máquina M1 del dominio de TM_j (si TM_j tuviera solo a M1 como valor en su dominio, se procede inmediatamente a realizar el backtracking
    - Alternativamente se puede asignar TM_j primero, y propagar la restricción sobre TS_j (el orden de elección de variables dependerá de la heurística utilizada)

Grafo de restricciones modelo (el real tendría instancias de cada variable TS1, TM1, TS2, TM2, etc.)



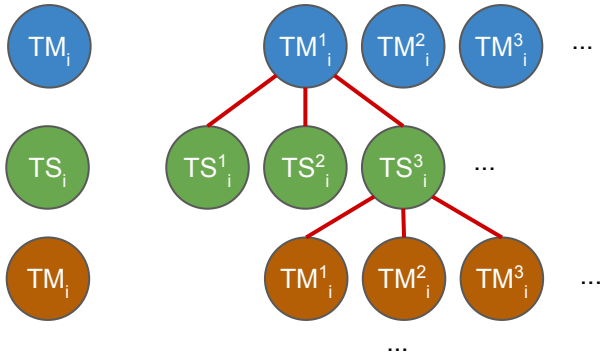
Scheduling con Constraint Satisfaction

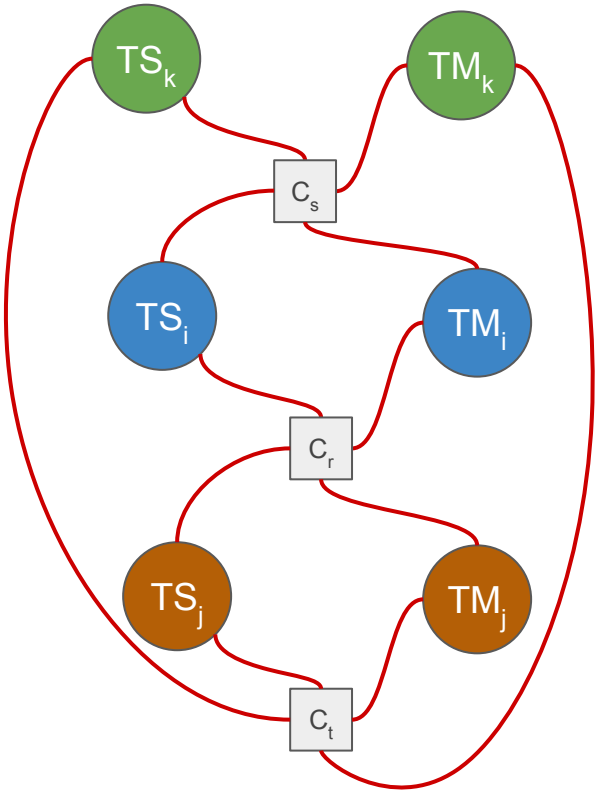
- Propagación de restricciones
  - Es posible que algunas variables intervengan en varias restricciones
    - Asignación se propaga en varias restricciones



Scheduling con Constraint Satisfaction

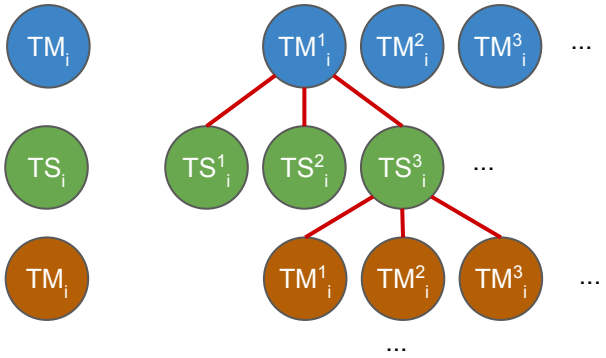
- Estrategias de búsqueda
  - Búsqueda global
    - Árbol de búsqueda
    - En cada nivel se asigna valor a una variable
    - Nodos en cada nivel: valores alternativos para la variable elegida
    - Al hacer backtracking, es necesario revertir los valores de los dominios de las variables que fueron alteradas, o bien cada nodo debe tener un estado completo del grafo (requiere más memoria)

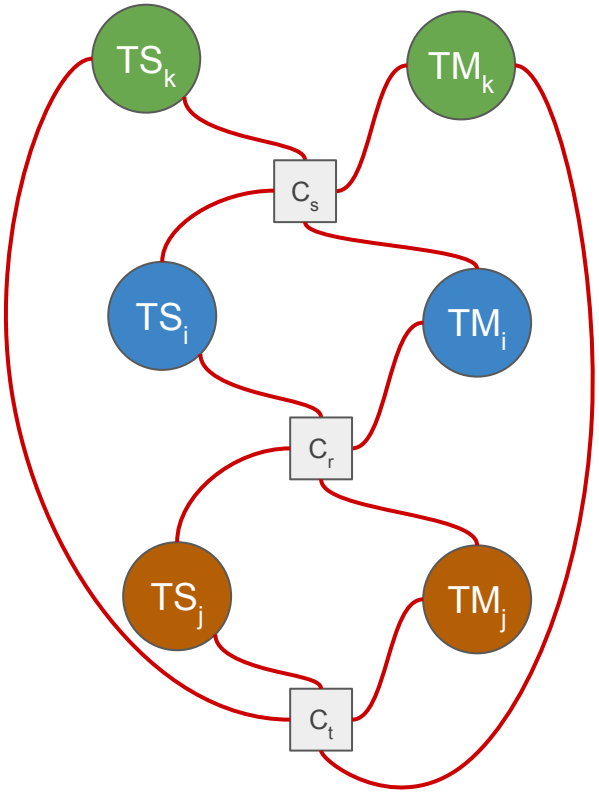




Scheduling con Constraint Satisfaction

- Estrategias de búsqueda
  - Búsqueda global con optimización
    - Opcionalmente se define un criterio de optimización (ej: tiempo total o *time span,* algún modelo de costo de producción, etc)
    - La búsqueda es similar al caso anterior pero se puede usar A*
    - Complejidad adicional: cada nodo a expandir debe mantener el estado de todas las variables (no sólo se revierten los valores en backtracking, sino también en “cambio del nodo a expandir”)





Scheduling con Constraint Satisfaction

- Estrategias de búsqueda
  - Búsqueda local
    - Se parte de una asignación *completa* de variables (aunque seguramente *inconsistente*)
    - Se aplica algún algoritmo de búsqueda local para comenzar a cambiar el valor de cada variable (hill climbing, temple simulado, algoritmo genético, etc)
    - Cambia la heurística a utilizar (ej: variable más conflictiva)

    - Ventaja: puede ser mucho más eficiente que la búsqueda global
    - Desventaja: es difícil incluso plantear un criterio de optimización (además de que los algoritmos son inherentemente subóptimos)