Razonamiento y planificación automática

Alejandro Cervantes

Laboratorio: Planificación con PDDL

Presentación Laboratorio

Entorno

- Se recomienda mucho el plugin para Visual Studio Code
- Podéis usar directamente Editor.planning.domains, pero tendréis problemas para depurar.
- Podéis cambiar el planificador (pero a veces es difícil de ejecutar, suele requerir compilación)

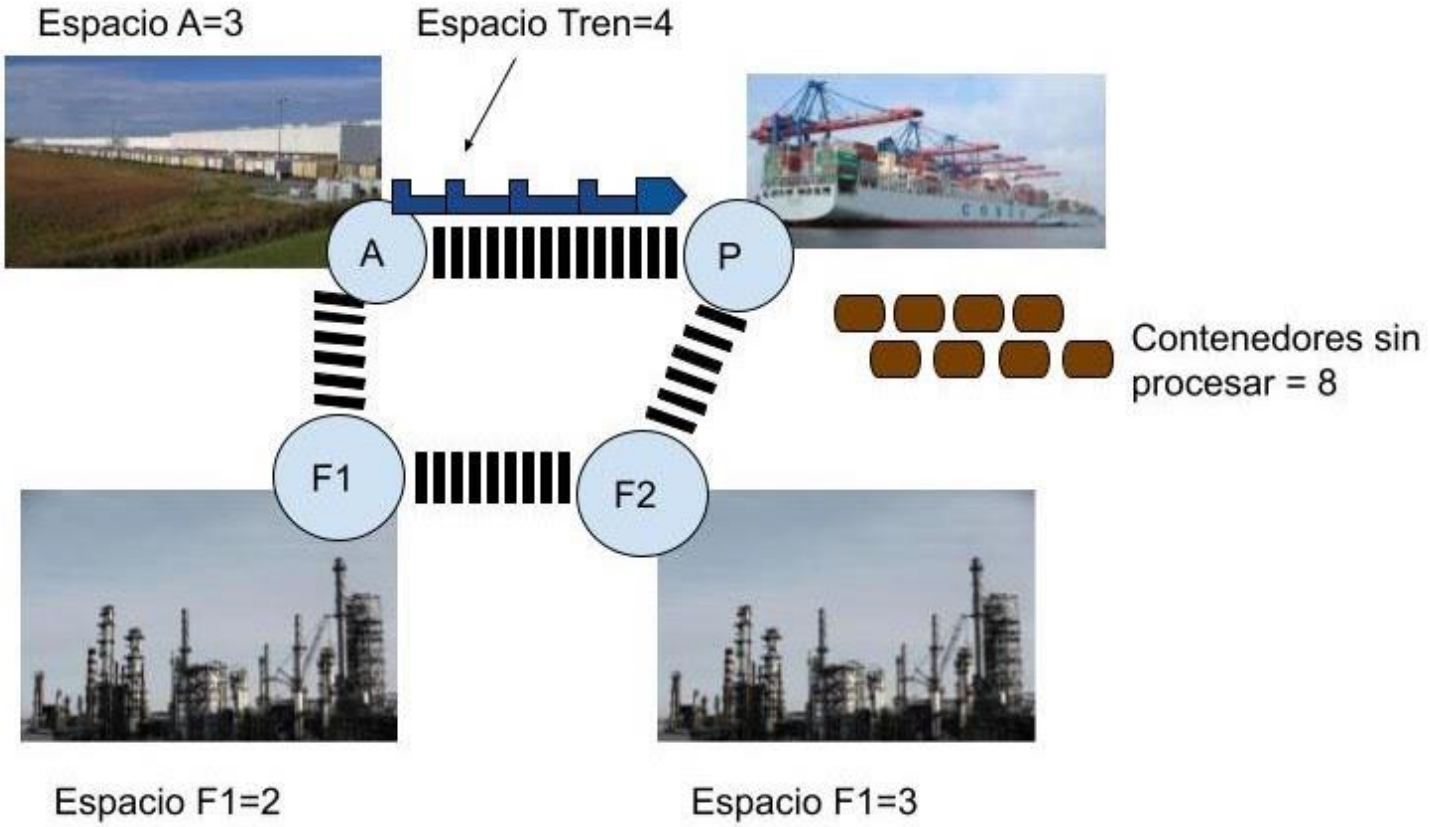
Parte 1: Caso base

- Se indica la distribución concreta de contenedores, fábricas y depósito destino

Parte 2: Modificaciones

- Se trata de hacer modificaciones interesantes del problema del caso base, manteniendo el dominio

Dominio y caso base



Operadores

• Quedarse **parado** (no necesaria, pero implícita si procesar es una acción)

• **Mover** a una localización **directamente conectada** sobre la vía (es decir, de A a P, o de F1 a A, etc.). **Puede moverse en direcciones opuestas en acciones sucesivas** (es decir, puede darse la vuelta ya que tiene máquina en ambos extremos).

• **Cargar** un contenedor en cualquier localización donde se encuentre, **siempre y cuando no se exceda el espacio disponible**.

• **Descargar** un contenedor en cualquier localización donde se encuentre, a excepción del puerto. Recuerde que al descargar no se puede exceder la capacidad de la localización.

LIMITACIONES y CONSEJOS IMPORTANTES

• Los planificadores **no tienen por qué encontrar el óptimo**. El que se usa en planning.domains en particular no lo intenta

• **No uséis fluents numéricos** para contar capacidades, el planificador de planning.domains no los implementa

• Usad **tipos**

• Codificad siempre de forma **incremental**. Guardad siempre versiones que funcionen. Haced pruebas de cada regla por separado.

• Podéis usar el dominio "transport" de IPC 2008 como ejemplo para gestionar capacidades numéricas, o usar un enfoque orientado a objetos como "storage" de IPC 2006

Estándar de PDDL: PDDL 1.2, sin valores numéricos. Podéis usar requirements como :adl.

El planificador de Planning Domains

• Planning.domains utiliza el planificador **siw-then-bfs**. (http://www.lapkt.org) Tiene varios modos de funcionamiento y encadena varios algoritmos.

La salida del planificador indica que usa:

- Primera opción: usa un algoritmo llamado IW, como Breadth-First-Search pero descarta algunos nodos. La versión serializada (SIW) intenta encontrar cada subobjetivo de forma independiente, de modo que no se deshagan los subobjetivos anteriores. También se puede ejecutar sobre un problema relajado.

- Si falla, se combina con un algoritmo de búsqueda heurística avara (Best-first Search) cuya función de evaluación depende de la novedad del nodo y otros criterios (otros planificadores usan la primera pasada para calcular H de la segunda)

- Referencia: Christian Muise, Nir Lipovetzky, Miquel Ramirez (2015). MAP-LAPKT: Omnipotent Multi-Agent Planning via Compilation to Classical Planning. *Proceedings of the Competition of Distributed and Multiagent Planners (CoDMAP)*.

Instalación de planificadores adicionales

Los planificadores de la competición ICAPS (https://www.icaps-conference.org/) se distribuyen como imagen que se puede ejecutar usando **Singularity**.

Instrucciones Windows:

https://www.blopig.com/blog/2021/09/using-singularity-on-windows-with-wsl2/

www.unir.net